НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматики та управляння в технічних системах

«Система для бронювання квестів через мережу Інтернет»

Курсова робота

З дисципліни «Об’єктно-орієнтоване програмування»

|  |  |
| --- | --- |
| Керівник  Хмелюк В.С  «Допущений до захисту»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Особистий підпис керівника)  « » \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 р.  Захищений з оцінкою  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оцінка)  Члени комісії:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Особистий підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Особистий підпис) | Виконавець  ст. Овсеєнко В.К.  зал. Книжка № ЗПІ-зп6123  гр. ЗПІ-зп 61  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Особистий підпис виконавця)  « » \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 р.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­­\_\_\_\_\_\_\_\_\_  (Розшифровка підпису)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Розшифровка підпису) |

Київ - 2017

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра АВТОМАТИКИ ТА УПРАВЛЯННЯ В ТЕХНІЧНИХ СИСТЕМАХ

Дисципліна «Об’єктно-орієнтоване програмування»

Курс 2 Група ЗПІ-зп61 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Овсеєнко В.К. |
| (прізвище, ім’я, по батькові) |

|  |  |
| --- | --- |
| 1. Тема роботи | «Система для бронювання квестів через мережу Інтернет» |
|  | |
|  | |
|  | |

|  |  |
| --- | --- |
| 2. Строк здачі студентом закінченої роботи | 06 06 2017 |

|  |
| --- |
| 3. Вихідні дані до роботи: |
| Реалізація програми в програмі Visual Studio 2017 з використанням фреймворку ASP.Net Core |
|  |
|  |

|  |
| --- |
| 4. Зміст розрахунково-пояснювальної записку (перелік питань, які підлягають розробці) |
| *1.Вступ. 2. Постановка задачі. 3. Огляд існуючих рішень та опис моделі.* |
| *4. Режим роботи та опис діалогу. 5. Опис програми. 6. Інструкція програміста.* |
| *7. Керівництво користувача 8. Висновки.* |
| *9. Список використаних джерел.* |

|  |
| --- |
| 5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) |
| Діаграма класів, діаграма зв’язки структури |
|  |
|  |
|  |

|  |  |
| --- | --- |
| 6. Дата видачі завдання | 14 02 2017 |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Назва етапів виконання курсової роботи** | **Строк виконання етапів роботи** | **Підписи або примітки** |
| 1. | Отримання та узгодження теми курсової роботи | 14.02.2017 |  |
| 2. | Огляд існуючих рішень з тематики роботи | до 03.03.2017 |  |
| 3. | Розробка моделі та структури програми | до 12.03.2017 |  |
| 4. | Вибір алгоритмів розв’язання задачі | до 18.03.2017 |  |
| 5. | Розробка діалогового інтерфейсу програми | до 28.03.2017 |  |
| 6. | Кодування програми | до 06.04.2017 |  |
| 7. | Налагодження та перевірка програми | до 18.04.2017 |  |
| 8. | Оформлення пояснювальної записки | до 22.05.2017 |  |
| 9. | Представлення курсової роботи до захисту | 06.06.2017 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Студент** |  |  | Овсеєнко В.К. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Керівник** |  |  | Хмелюк В.С. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2017 р.

ЗМІСТ

[ВСТУП 5](#_Toc484344131)

[МАТЕМАТИЧНА МОДЕЛЬ ТА СТУКТУРА ПРОГРАМИ 7](#_Toc484344132)

[ПЕРЕЛІК ТА ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТУКТУРА ДІАЛОГУ 10](#_Toc484344133)

[СТРУКТУРА ДАНИХ ТА РЕСУРСІВ ПРОГРАМИ 12](#_Toc484344134)

[ОПИС ПРОГРАМИ 16](#_Toc484344135)

[ІНСТРУКЦІЯ ПРОГРАМІСТА 19](#_Toc484344136)

[КЕРІВНИЦТВО КОРИСТУВАЧА 25](#_Toc484344137)

[ВИСНОВКИ 28](#_Toc484344138)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 29](#_Toc484344139)

# ВСТУП

Дана курсова робота описує структуру та методи програми, яка є частиною веб-додатку. Веб-додаток являє собою веб-сайт з переліком популярних на сьогоднішній день квест-кімнат. На сайті користувачі після обов’язкової реєстрації мають змогу забронювати обраний час з переліку доступних для бронювання. Серед основних функцій програми – реалізація реєстрації користувачів та бронювання обраного часу. Програма написана за допомогою сучасного фреймворку ASP.NET Core на мові програмування C#. Даний фреймворк дуже популярний та широко використовується у світі для створення веб-додатків та мобільних додатків. Для написання та компіляції програми ми використовували програмне забезпечення від компанії Microsoft Visual Studio 2017.

Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов програмування, хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET. Символ # у назві мови можна інтерпретувати і як дві пари плюсів ++, що натякають на новий крок у розвитку мови порівняно з C++ (подібно до кроку від C до C++), і як музичний символ дієз, разом з буквою C, що становить в англійській мові назву ноти до-дієз. Останнє й дало назву мові. Попри те, що символ # (октоторп) насправді є символом для позначення номера на більшості клавіатур і відрізняється від символу дієз ♯ (Unicode U+266F), Microsoft, як автор мови, неодноразово зверталася до своїх клієнтів з проханням прийняти таку стилізацію. C# розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. (Проте ця закономірність буде порушена з виходом C# 3.0, що є розширеннями мови, що не спираються на розширення платформи .NET.) CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо.

Титульним компілятором C# є Microsoft Visual C#. Існують інші компілятори C#, часто вони включають реалізації Common Language Infrastructure і бібліотеки класів .NET.

# МАТЕМАТИЧНА МОДЕЛЬ ТА СТУКТУРА ПРОГРАМИ

Проект, який ми розглядаємо у цій роботі, являє собою веб-додаток, веб-сайт, на якому представлена певна кількість квест-кімнат та доступного для їх бронювання часу. Користувачу пропонується зареєструватись або увійти в систему (якщо користувач уже зареєструвався раніше), вибрати той чи інший квест та забронювати його.

Проект у Visual Studio 2017 має робочу назву VO\_Quests та складається з кількох основних частин. Загальну схему проекту наведено нижче:

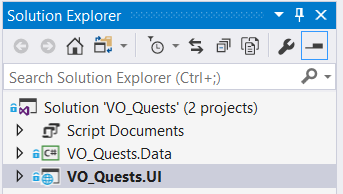


Рисунок 1 – Загальна схема проекту у Visual Studio

У папці Script Documents знаходяться файли зовнішніх бібліотек, необхідних для проекту. VO\_Quests.Data, як очевидно з назви, відповідає за зв'язок з базами даних та містить посилання на них. Структура цієї частини проекту наступна:

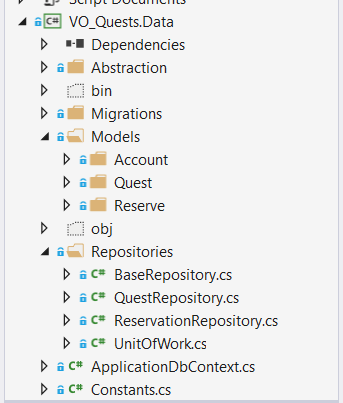


Рисунок 2 – Структура VO\_Quests.Data

Файл Constants.cs містить клас Constants з основними змінними, які використовуються у проекті. Наприклад, змінні початкової та кінцевої годин доступних для бронювань:

public const int QuestsStartHour = 10;

public const int QuestsEndHour = 20;

У класі ApplicationDbContext знаходяться посилання на основні репозиторії Quest та Reservation. Нагадаємо, що репозиторії – це паттерн у мові C#, який дуже часто використовується у якості проміжної ланки між класами, які безпосередньо взаємодіють з даними та іншою частиною програми. Це дає змогу зробити програму більш гнучкою та менш вразливою до зміни баз даних.

Структуру таблиць баз даних та зв'язок між ними можна графічно зобразити наступним чином:

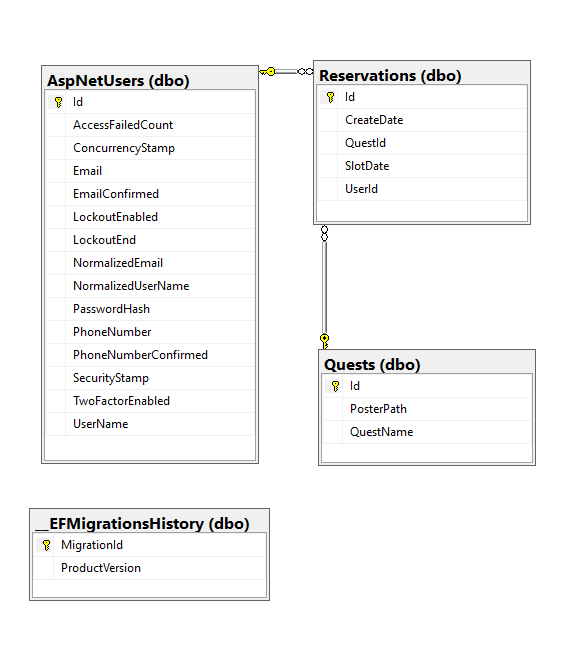


Рисунок 3 – Схема бази даних

У свою чергу, VO\_Quests.UI – це та частина проекту, у якій знаходиться основна логіка проекту, ті файли, які відповідають за її відображення та інтеракцію з користувачем, а також посилання на VO\_Quests.Data для зв’язку з базою даних. Така структура допомагає зробити проект більш гнучким на менш залежним від ймовірних наслідків при зміні бази даних.

Оскільки для реалізації проекту ми використовувати популярний фреймворк ASP.NET Core, який є MVC-фреймворком, тобто реалізує архітектуру Модель-Представлення-Контроллер, то основними частинами нашого проекту являються контроллери (Controllers), моделі (ViewModels) та представлення (View). Нагадаємо, що Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку. Представлення – це те, що ми отримуємо на виході виконання програми, тобто те, з чим безпосередньо взаємодіє користувач. А Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду. Структура папки VO\_Quests.UI наступна:

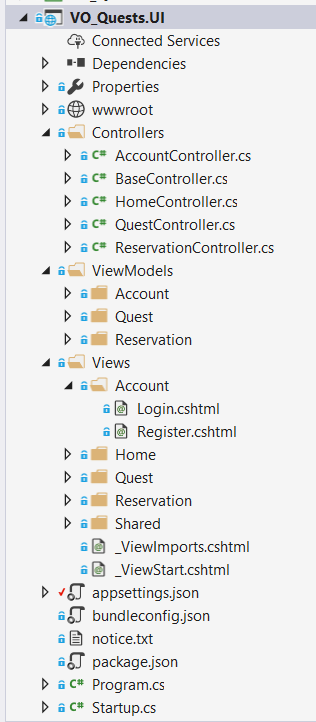


Рисунок 4 – Структура VO\_Quests.UI

# ПЕРЕЛІК ТА ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТУКТУРА ДІАЛОГУ

Як уже зазначалось у попередньому розділі, проект під робочою назвою “VO\_Quests” складається з двох основних частин, одна з яких відповідає за зв'язок з базою даних, а інша за інтерфейсну частину, тобто візуальне відображення проекту. Для того, щоб зібрати проект та запустити локальний сервер, необхідно відкрити проект у програмі Microsoft Visual Studio 2017 та вибрати необхідні параметри, зокрема - який саме проект необхідно зібрати на запустити. У нашому випадку це – VO\_Quests.UI. Для цього необхідно на верхній панелі знайти та натиснути кнопку Run у вигляді трикутника зеленого кольору:

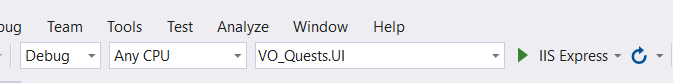


Рисунок 5 – Верхня панель програми Microsoft Visual Studio 2017

Після цього, якщо не виникне жодних помилок, в терміналі буде виведене наступне повідомлення та запущений локальний сервер, початкова сторінка проекту буде відкрита у обраному браузері.

IIS (Internet Information Services, до версії 5.1 — Internet Information Server) — це набір серверів для декількох служб Інтернету від компанії Майкрософт. IIS поширюється з операційними системами родини Windows NT.

Основний компонент IIS — веб-сервер, який дозволяє розміщувати в Інтернеті сайти. IIS підтримує протоколи HTTP, HTTPS, FTP, POP3, SMTP, NNTP. IIS другий за популярністю веб-сервер за кількістю сайтів, після Apache HTTP Server. За даними компанії Netcraft на 11.10.2007, понад 37.13% сайтів обслуговується веб-сервером IIS.

У нашому проекті у якості сервера ми використовуємо IIS Express. Він являє собою легкий, автономний варіант IIS, оптимізований для розробників. IIS Express дозволяє легко використовувати найостаннішу версію IIS для розробки і тестування веб-сайтів. Він має всі основні можливості IIS 7 і вище, а також додаткові функції, призначені для полегшення розробки веб-сайтую.

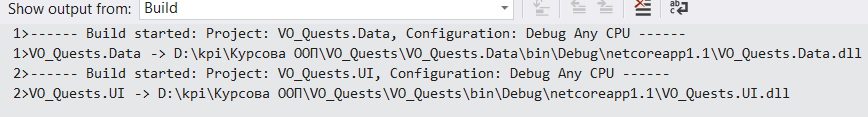


Рисунок 6 – Вивід у терміналі після успішного запуску програми

У браузері відобразиться початкова сторінка проекту. Інтерфейс у браузері та його поведінку, а також керівництво користувача описано нами у розділі «Керівництво користувача».

Для того, щоб зупинити сервер, необхідно натиснути кнопку у вигляді червоного квадрата на верхній панелі програми Microsoft Visual Studio 2017.

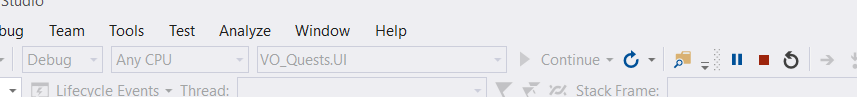


Рисунок 7 – Верхня панель Microsoft Visual Studio 2017 під час роботи програми

# СТРУКТУРА ДАНИХ ТА РЕСУРСІВ ПРОГРАМИ

Перед тим, як розглянути детально кожен клас, його ідентифікатори та функції, необхідно спочатку навести загальну структуру проекту та взаємозв’язок класів. Структуру даних та взаємозв’язок класів програми можна представити у графічному вигляді:

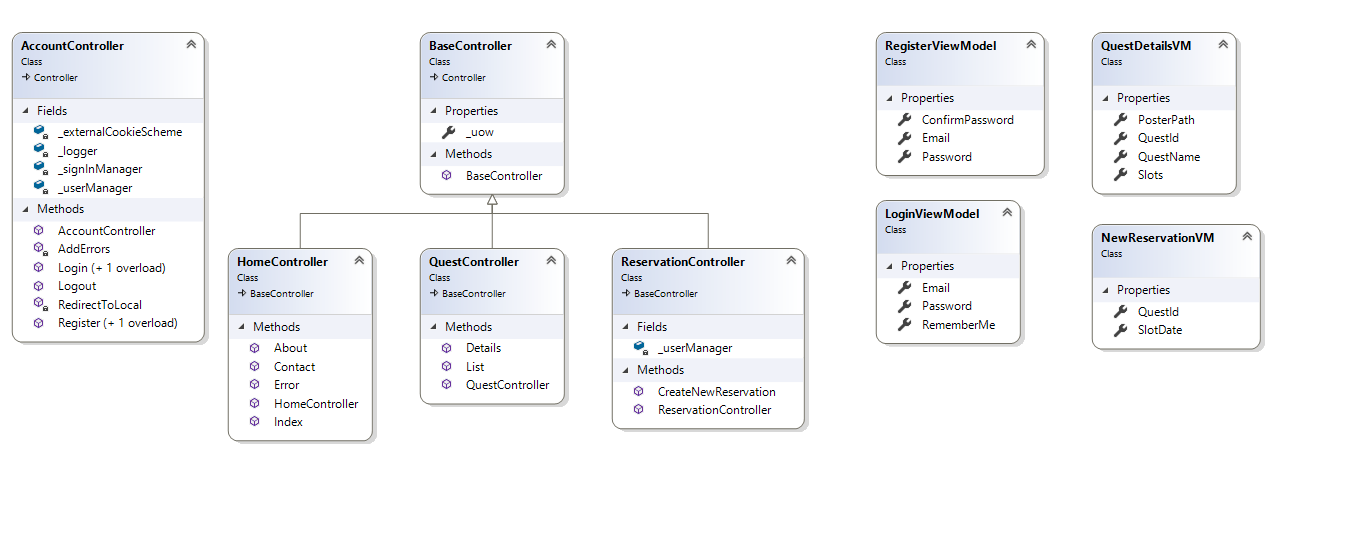


Рисунок 8 – Структура проекту

Як бачимо з наведеної схеми, можна виділити кілька основних класів – AccountController, BaseController, HomeController, QuestController, ReservationController, RegisterViewModel, QuestDetailsVM, LoginViewModel, NewReservationVM. Далі ми розглянемо окремо кожен з наведених класів та способи їх взаємодії між собою.

Клас AccountController відповідає за реєстрацію користувачів та занесення до бази даних інформації про них. Цей клас користується методами класів фреймворка ASP.Net Corе, такими як: UserManager та SignInManager, які у свою чергу належать до класу Identity. Оскільки це стандартні класи фреймворка, поля, в яких їх обявлено мають модифікатор readonly.

AccountController містить також конструктор, який має таку ж назву як і клас. Його синтаксис подібний до методу, проте різниця у тому, що конструктор не повертає значення, а завдання його полягає у ініціалізації самого класу та наданню значень початковим змінним екземпляру.

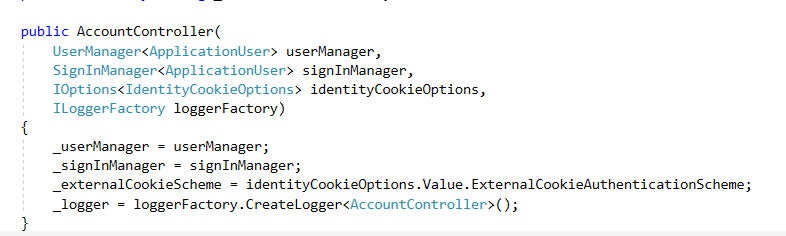


Рисунок 9 – AccountController

Також у класі AccountController зберігаються методи які відповідають за занесення до бази даних інформації про зареєстрованих користувачів. До бази даних заноситься інформація про ім’я користувача, його імейл, пароль, а також кожному користувачеві присвоюється свій унікальний id.

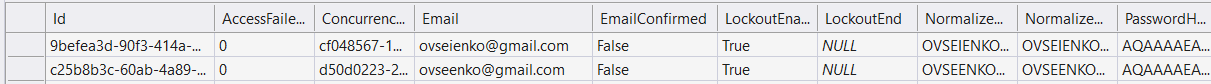
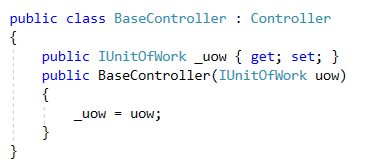


Рисунок 10 – Поля у базі даних

У проекті ми використовуємо паттерн UnitOfWork, основна ідея якого полягає у тому, що всі транзакції всіх репозиторіїв з базою даних відбуваються через один спільний клас BaseController, що гаранту спільний контекст для кількох потрібних нам репозиторіїв. Тобто можна сказати що клас BaseController – це колекція репозиторіїв, до якої входять QuestController, HomeController та ReservationController, які ми розглянемо більш детально нижче.



Рисунко 11 – BaseController

Клас HomeController містить метод Index(), який використовує інтерфейс IActionResult, назва якого за домовленістю починається з літери І. Інтерфейси дозволяють визначити певний функціонал, який не має конкретної реалізації. Цей функціонал буде реалізовано класами, які цей інтерфейс використовуватимуть. IActionResult, який знаходиться у просторі імен Microsoft.AspNetCore.Mvc призначено для генерації результату дії. У нашому випадку метод Index відповідає за відображення згенерованої головної html-сторінки.

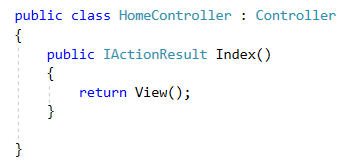


Рисунок 12 – HomeController

Основна задача класу QuestController полягає у відображенні списку квестів та їх деталей а також отримання інформації з бази даних про бронювання того чи іншого квесту тим чи іншим користувачем, використовуючи описаний вище паттерн UnitOfWork. Клас містить методи List() (відображення повного списку квестів) та Details(int id) (отримання інформації про окремий квест за унікальним id).

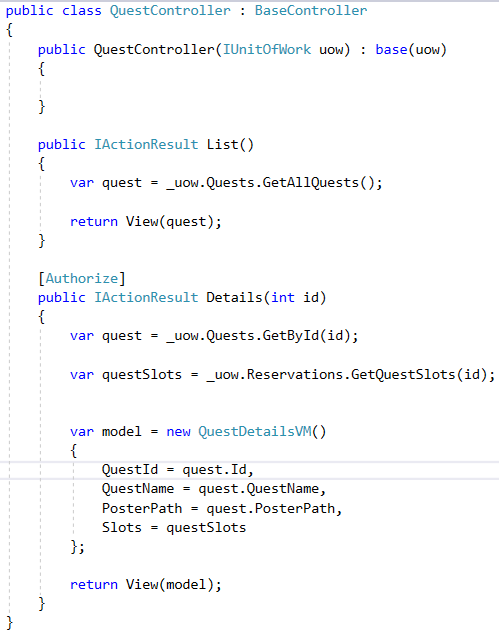


Рисунок 13 – QuestController

Що ж стосується класу ReservationController, то основним призначенням його методу CreateNewReservation є запис до бази даних інформації про створення нового бронювання. Дані про нове бронювання передаються через модель NewReservationVM, яка містить інформацію про квест, користувача, який зробив бронювання час, на який було заброньовано квест та час, коли це було зроблено.



Рисунок 14 – ReservationController

# ОПИС ПРОГРАМИ

Якщо зобразити у вигляді схеми структуру проекту, то схема мала б такий вигляд:

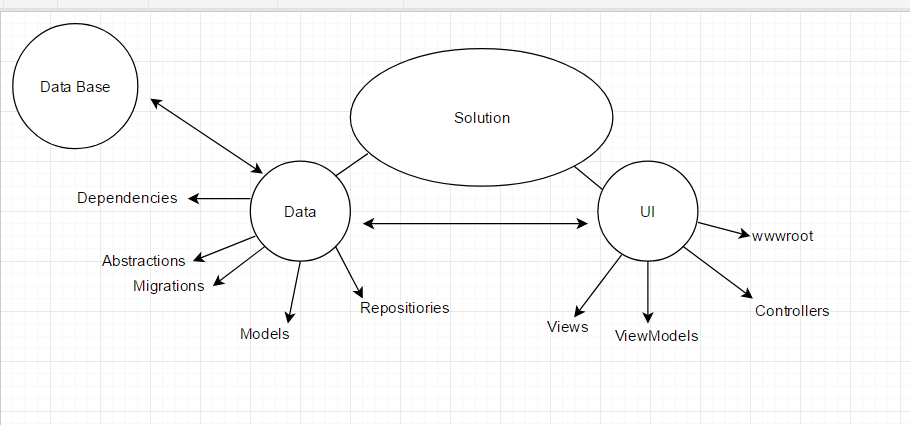


Рисунок 15 – Загальна схема програми

З наведеної схеми видно, що проект, що має назву VO\_Quests складається з двох основних частин – VO\_Quests.Data та VO\_Quests.UI. Отримання інформації з бази даних та запис до неї здійснюється через Data. Це зроблено з тією метою, щоб зробити проект більш гнучким на менш залежним від ймовірних наслідків при зміні бази даних.

До моделей VO\_Quests.Data належать Account, Quest та Reserve. Моделі вважаються однією з найголовніших частин проекту, оскільки саме вони закладають основну логіку та мають зв'язок з контроллерами. За допомогою моделей в контролері створюються нові екземпляри, а також викликаються необхідні методи. Так, наприклад у класі моделі Quest встановлюються основні поля відомостей про окремий квест, які передаються до бази даний у випадку бронювання.

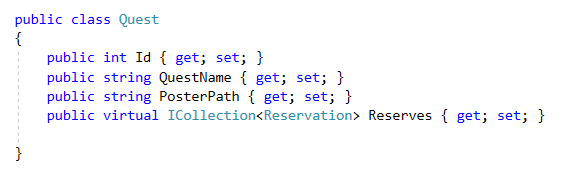


Рисунок 16 – клас Quest

До репозиторіїв VO\_Quests.Data належать BaseRepository, ReservationRepository, QuestRepository та UnitOfWork. Як уже зазначалось у попередньому розділі, ми використовуємо паттерн UnitOfWork, який дозволяє бути впевненими у тому, що всі репозиторії мають спільний контекст звернення до бази даних.

VO\_Quests.UI складається з трьох основних для mvc-фреймворку частин – Views, ViewModels, Controllers та директорії wwwroot, де знаходяться сторонні бібліотеки та скрипти, картинки та файли зі стилями.

До контролерів належать AccountController, BaseController, HomeController, QuestController, ReservationController. До моделей – Account, Quest та Reservation. У директорії View знаходяться файли .cshtml які після обробки відображаються у браузері.

Для того, щоб зібрати та запустити проект необхідно натиснути кнопку Run.

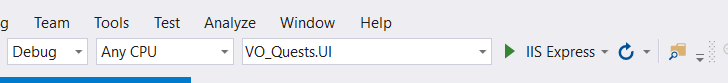


Рисунок 17 – Верхня панель Microsoft Visual Studio

Після чого буде запущено сервер, а після того як проект буде зібрано без помилок з’явиться повідомлення у терміналі.

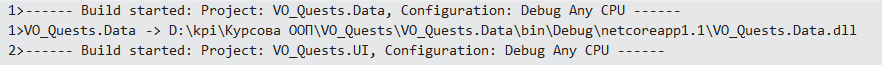


Рисунок 18 – Вивід у терміналі

Після цього відкриється браузер з головною сторінкою нашого веб-додатку.

У випадку ж якщо локальний сервер буде зупинено, у терміналі з’явиться наступне повідомлення:

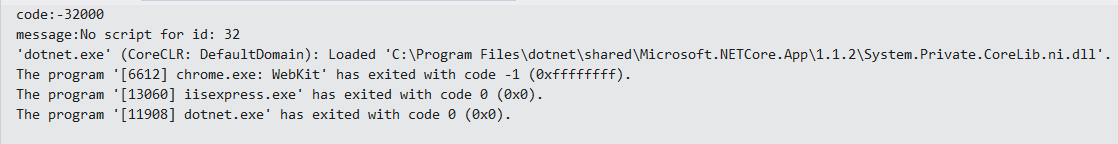


Рисунок 19 – Вивід у терміналі після зупинки локального сервера

# 

# ІНСТРУКЦІЯ ПРОГРАМІСТА

Програма під робочою назвою “VO\_Quests” написана на мові C# у програмі Visual Studio 2017. Це програмне забезпечення має все необхідне для розробки, збирання та запуску проекту. Visual Studio 2017 може бути встановлена на таких операційних системах, як Windows та Mac OS. Вимоги до операціних систем для встановлення Visual Studio 2017:

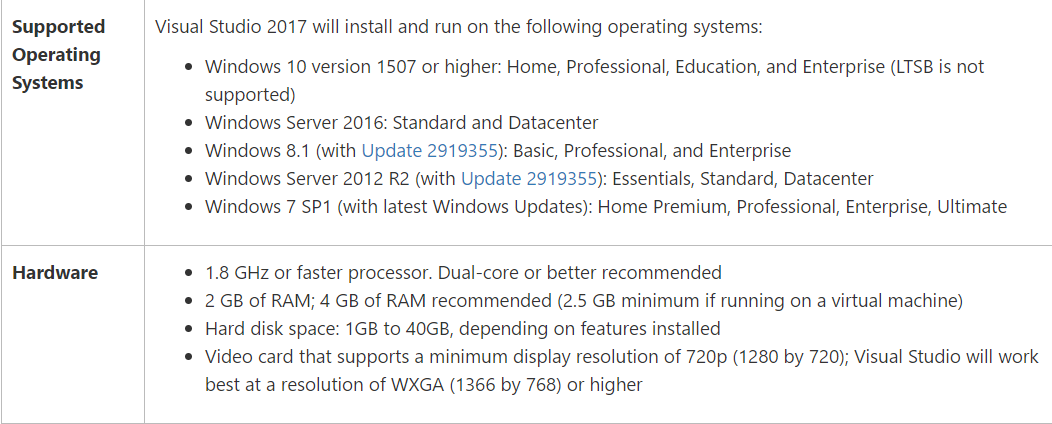


Рисунок 20 – Вимоги щодо ОС

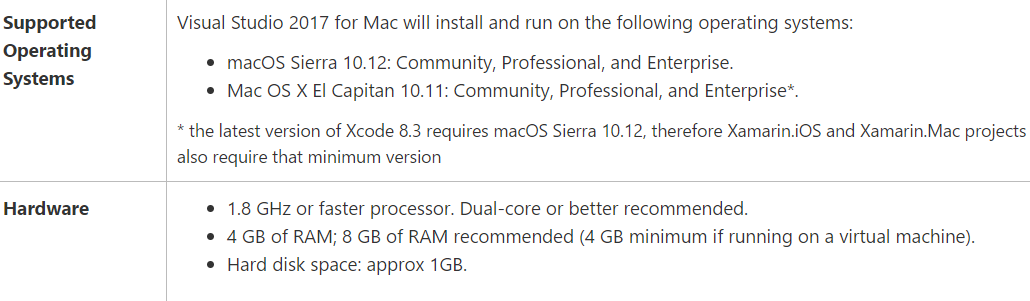


Рисунок 21 – Вимоги щодо ОС

Програма розробляється на платформі ASP.NET MVC 5 з використанням мови програмування C #. ASP.NET MVC являє собою фреймворк для створення Веб-додатків на платформі .NET, який реалізує паттерн MVC.

ASP.NET — технологія створення веб-застосунків і веб-сервісів від компанії Майкрософт. Вона є складовою частиною платформи Microsoft.NET і розвитком старішої технології Microsoft ASP. У цей час останньою версією цієї технології є ASP.NET 5.

ASP.NET зовні багато в чому зберігає схожість із старішою технологією ASP, що дозволяє розробникам відносно легко перейти на ASP.NET. У той же час внутрішній устрій ASP.NET істотно відрізняється від ASP, оскільки вона заснована на платформі. NET і, отже, використовує всі нові можливості, що надаються цією платформою. Після випуску сервера Internet Information Services 4.0 в 1997 році, компанія Microsoft почала досліджувати можливість нової моделі веб-застосунків, яка задовольнить скарги на ASP, особливо пов'язані з відділенням оформлення від змісту, і яка дозволить писати «чистий» код. Робота з розробки такої моделі була доручена Марку Андерсу, менеджеру команди IIS, і Скотту Гутрі, що прийшов на роботу в Microsoft в 1997. Андерс і Гутрі розробили початковий проект протягом двох місяців, і Гутрі написав код первісного прототипу під час різдвяних канікул 1997 року. Прототип XSP був написаний на Java, але скоро було вирішено побудувати нову платформу на основі Common Language Runtime (CLR), бо на платформу Java у компанії Microsoft закінчувалась ліцензія. Гутрі описав це рішення як «величезний ризик», тому що успіх нової розробки був пов'язаний з успіхом CLR, яка, як і XSP, перебувала на ранній стадії розробки. Хоча ASP.NET бере свою назву від старої технології Microsoft ASP, вона значно від неї відрізняється. Microsoft повністю перебудувала ASP.NET, ґрунтуючись на Common Language Runtime (CLR), який є основою всіх застосунків Microsoft .NET. Розробники можуть писати код для ASP.NET, використовуючи практично будь-які мови програмування, що входять у комплект .NET Framework (C#, Visual Basic.NET, і JScript.NET). ASP.NET має перевагу у швидкості в порівнянні зі скриптовими технологіями, тому що при першому зверненні код компілюється і поміщається в спеціальний кеш, і згодом тільки виконується, не вимагаючи витрат часу на парсинг, оптимізацію, і т. д.

**Переваги ASP.NET**

Розглянемо переваги ASP.NET.

1. ASP.NET має перевагу у швидкості в порівнянні з іншими технологіями, заснованими на скриптах (PHP, тощо).
2. Розширюваний набір елементів управління і бібліотек класів дозволяє швидше розробляти застосунки.
3. ASP.NET спирається на багатомовні можливості .NET, що дозволяє
4. писати код сторінок на C#, VB, C/C++ та ін.

4. Поділ візуальної частини та бізнес-логіки.

5. Розширювана модель обробки запитів.

Моде́ль-вигляд-контро́лер (або Модель-вид-контролер, англ. Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення. Цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування. Застосовується для відокремлення даних (модель) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача. Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності. Архітектурний шаблон Модель-Вид-Контролер (MVC) поділяє програму на три частини. У тріаді до обов'язків компоненту Модель (Model) входить зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель. Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

MVC поділяє цю частину системи на три самостійні частини: введення даних, компонент обробки даних і виведення інформації. Модель, як вже було відмічено, інкапсулює ядро даних і основний функціонал з їх обробки. Також компонент Модель не залежить від процесу введення або виведення даних. Компонент виводу Вигляд може мати декілька взаємопов'язаних областей, наприклад, різні таблиці і поля форм, в яких відображається інформація. У функції Контролера входить моніторинг за подіями, що виникають в результаті дій користувача (зміна положення курсора миші, натиснення кнопки або введення даних в текстове поле).

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам Моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через Контролер внесе зміни до Моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

ASP.NET MVC являє собою платформу для створення сайтів і веб-додатків з використанням патерну (або шаблону) MVC (model - view - controller). Робота над новою платформою була розпочата в 2007 році, а в 2009 році з'явилася перша версія. В результаті до теперішнього моменту (2017 рік) вже було випущено 5 версій платформи, а сам фреймворк знайшов велику популярність по всьому світу завдяки своїй гнучкості та адаптивності. Шаблон MVC, що лежить в основі нової платформи, має на увазі взаємодію трьох компонентів: контролера (controller), моделі (model) і вигляд (view).

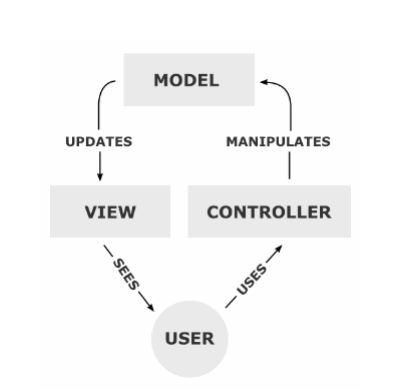


Рисунок 22 – Модель MVC

Контролер (controller) представляє клас, з якого власне і починається робота програми. Цей клас забезпечує зв'язок між моделлю і представленням. Отримуючи вводяться користувачем дані, контролер виходячи з внутрішньої логіки при необхідності звертається до моделі і генерує відповідне подання.

Вигляд (view) - це власне візуальна частина або призначений для користувача інтерфейс програми - наприклад, html-сторінка, через яку користувач, що зайшов на сайт, взаємодіє з веб-додатком.

Модель (model) представляє набір класів, що описують логіку використовуваних даних.

**Можливості платформи ASP.NET MVC**

Платформа ASP.NET MVC надає наступні можливості. По – перше, пооділ завдань програми (логіка введення, бізнес-логіка і логіка призначеного для користувача інтерфейсу), широке можливості тестування і розробки на основі тестування. Всі основні контракти платформи MVC засновані на інтерфейсі і підлягають тестуванню за допомогою макетів об'єкта, які імітують поведінку реальних об'єктів програми. Додаток можна піддавати модульному тестування без запуску контролерів в процесі ASP.NET, що прискорює тестування і робить його більш гнучким. Для тестування можливе використання будь-якої платформи модульного тестування, сумісної з .NET Framework. Наступною перевагою є можливість розширення і доповнення платформи. Компоненти платформи ASP.NET MVC можна легко замінити або налаштувати. Розробник може підключати власний механізм уявлень, політику маршрутизації URL-адрес, сериализацию параметрів методів дій і інші компоненти. Платформа ASP.NET MVC також підтримує використання моделей контейнера впровадження залежності (DI) і інверсії елемента управління (IOC). Модель впровадження залежності дозволяє впроваджувати об'єкти в клас, а не очікувати створення об'єкта самим класом. Модель інверсії елемента управління вказує на те, що якщо один об'єкт вимагає інший об'єкт, то перші об'єкти повинні отримати другий об'єкт із зовнішнього джерела (наприклад, з файлу конфігурації). Це полегшує тестування. Безперечно перевагою є розширена підтримка маршрутизації ASP.NET. Цей потужний компонент зіставлення URL-адрес дозволяє створювати додатки з зрозумілими URL-адресами, які можна використовувати в пошуку. URL-адреси не повинні містити розширення імен файлів і призначені для підтримки шаблонів іменування URL-адрес, які забезпечують адресацію, оптимізовану для пошукових систем (SEO) і для передачі репрезентативного стану (REST).

Підтримка існуючих функцій ASP.NET. ASP.NET MVC дозволяє використовувати такі функції, як перевірка справжності за допомогою форм і Windows, перевірка достовірності за URL-адресою, членство і ролі, кешування виведення і даних, управління станом сеансу і профілю, спостереження за працездатністю, система конфігурації і архітектура постачальника.

# КЕРІВНИЦТВО КОРИСТУВАЧА

Користувач знайомиться з запропонованою програмою у браузері. Перед ним відкривається початкова сторінка квестів VO\_Quests:

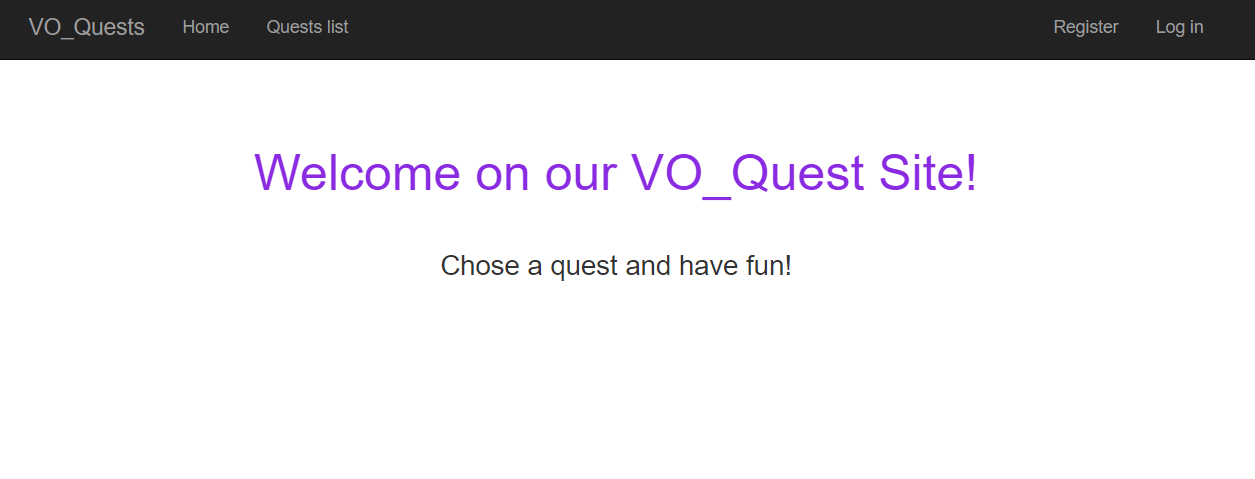


Рисунок 23 – Початкова сторінка

Якщо після цього користувач перейде на сторінку списку квестів, посилання на яку знаходиться у верхньому меню, він побачить список квестів, вільних до бронювання.

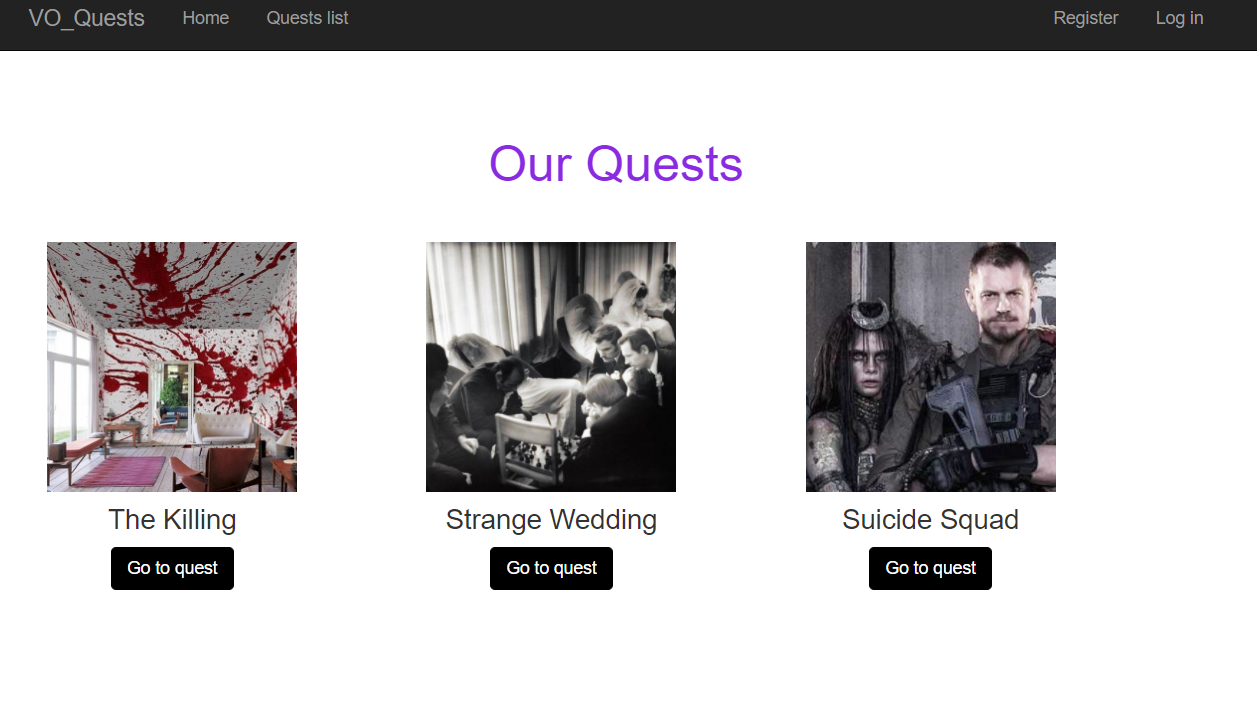


Рисунок 24 – Список квестів

Після вибору того чи іншого квесту, користувач або перейде на сторінку деталей квесту, або ж йому буде запропоновано зареєструватись чи залогінитись.

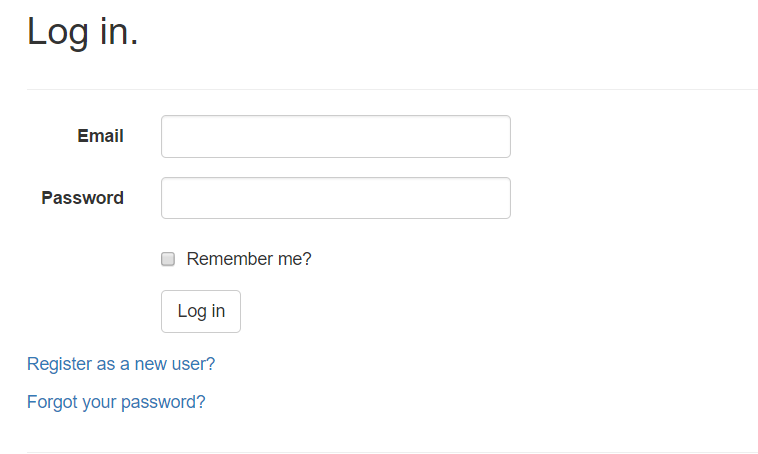


Рисунок 25 – Сторінка реєстрації користувача

У випадку, якщо користувач зареєстрований та залогінений, перед ним відкриється сторінка деталей окремого кресту на випадаючий список з доступним для бронювання часом.

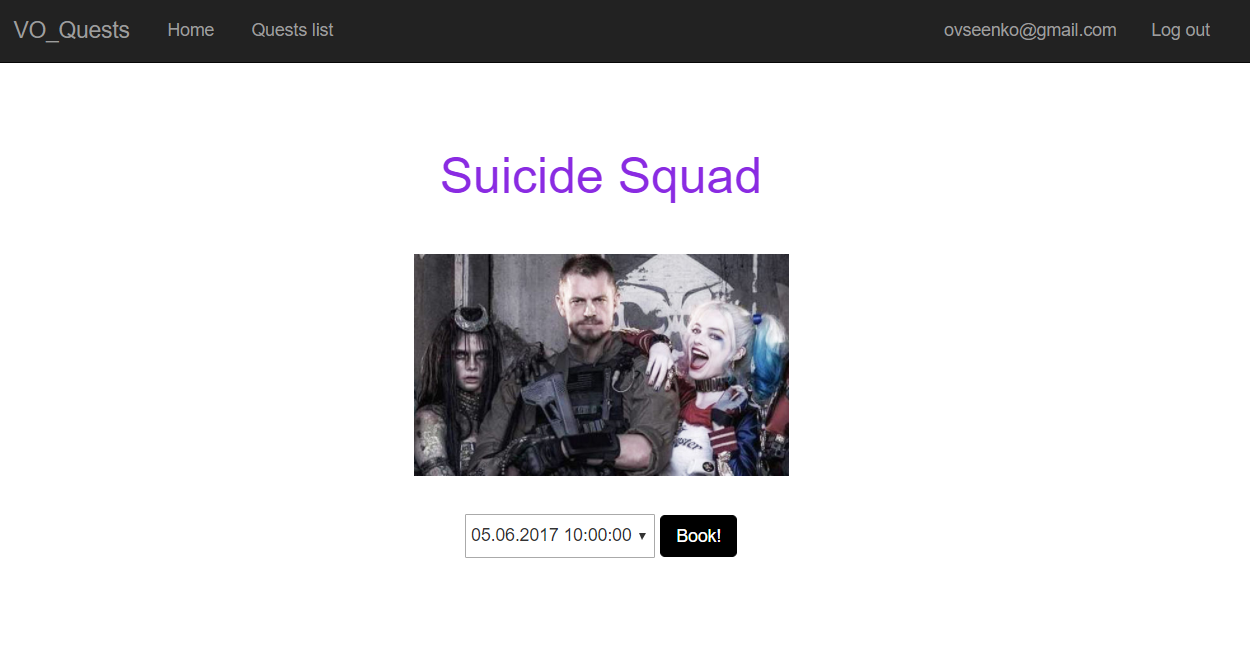


Рисунок 26 – Сторінка деталей окремого квесту

Після підтвердження реєстрації, у разі успішного виконання запису до бази даних, користувач побачить перед собою повідомлення про успішне виконання свого запиту. В іншому випадку на екран буде виведене повідомлення про помилку.

# ВИСНОВКИ

У ході курсової роботи ми розглянули програму онлайн-бронювання квестів, виконану у програмі Microsoft Visual Studio 2017 на мові програмування C#. Проект складається з двох частин, одна з яких відповідає за зв'язок з базою даних – отримання інформації з неї та занесення до неї відомостей, а інша – за формування візуального відображення проекту у браузері та основну логіку проекту. У ході курсової роботи ми детально розглянули структуру бази даних, класи та їхню взаємодію між собою, а також основні паттерни, якими ми користувались під час розробки програми.

Результатом даної роботи з точки зору користувача є веб-сторінка, яка пропонує на вибір кілька доступних до бронювання квестів. Після здійснення бронювання до бази даних заносяться відомості про користувача, деталі квесту та час, на який було здійснено бронювання.

Результати роботи було викладено на GitHub, де з ними і можна ознайомитись:

<https://github.com/Ovseienko/VO_Quests>

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Веб-сайт. [Електронний ресурс] : <https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82>.
2. Паттерн Unit of Work. [Електронний ресурс] : <https://metanit.com/sharp/mvc5/23.3.php>
3. Документация по ASP.NET MVC 4 [Електронний ресурс] : <https://msdn.microsoft.com/ru-ru/library/gg416514(v=vs.108).aspx>
4. Introduction to ASP.NET Core [Електронний ресурс] : <https://docs.microsoft.com/en-us/aspnet/core/>
5. Браузер. [Електронний ресурс] : https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80
6. Tim Berners-Lee: WorldWideWeb, the first Web client. [Електронний ресурс] : https://www.w3.org/People/Berners-Lee/WorldWideWeb.html
7. Статистика StatCounter. [Електронний ресурс] : http://gs.statcounter.com/#browser\_version-eu-daily-20090301-20090623
8. Microsoft Visual Studio. [Електронний ресурс] : https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio
9. Скачиваемые файлы для Visual Studio. [Електронний ресурс] : https://www.visualstudio.com/ru/downloads/
10. Google Chrome. [Електронний ресурс] <https://ru.wikipedia.org/wiki/Google_Chrome>
11. Быстрый запуск веб-приложений. [Електронний ресурс] : <https://www.google.ru/chrome/browser/features.html>.
12. Джейсон, Прайс; Майк, Гандэрлой Visual C# .NET. Полное руководство; КОРОНА принт, 2004.
13. Нейгел, К. C# 2005 для профессионалов; Вильямс, 2006.
14. Рихтер, Джефри CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#; Питер, 2007.