МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

СИСТЕМА КРАУДФАНДИНГА ДЛЯ СТУДЕНТСЬКИХ СТАРТАП-ПРОЕКТІВ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402. 21810210

Виконав студент 4-го курсу, групи 402
_____ Г. І. Єфремов
«__» червня 2022 р.

Керівник: ст.викладач
_____В. В. Кошовий
«__» червня 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20____р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Єфремову Георгію. Івановичу

1. Тема кваліфікаційної роботи «Система краудфандинга для студентських стартап-проектів».

Керівник роботи Кошовий Віталій Володимирович, ст.викладач Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» ____ 20__ р. № _____

2. Строк представлення кваліфікаційної роботи студентом «__» _____ 20__ р.

3. Очікуваний результат роботи: сайт із системою краудфандинга для студентських стартап-проектів.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):
- вибір інструментальних засобів створення застосунку;
- програмна реалізація веб сторінки;

5. Перелік графічного матеріалу: сторінок – 77 , таблиць – 2 , рисунків – 63 , посилань – 17 , презентація.

6. Завдання до спеціальної частини: «Охорона праці при користуванні екранними пристроями».

7. Консультанти:

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А.А., канд. техн. наук, доцент	

Керівник роботи ___ст.викладач Кошовий В.В._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання _Єфремов Г.І. _____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання « 20 » ___листопада____ 2021 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: «Система краудфандинга для студентських стартап-проектів»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	20.11.2021	20.11.2021	виконано
2	Отримання завдання на виконання БКР	20.11.2021	20.11.2021	виконано
3	Складання календарного плану роботи на весь період виконання БКР	08.12.2021	08.12.2021	виконано
4	Отримання завдання на переддипломну практику	23.05.2022	23.05.2022	виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	24.05.2022	02.06.2022	виконано
6	Розробка звіту з переддипломної практики	03.06.2022	05.06.2022	виконано
7	Виконання БКР: аналіз предметної сфери, вибір інструментальних засобів створення застосунку, розробка застосунку, тестування	27.02.2022	21.05.2022	виконано
8	Попередній захист БКР на засіданні комісії кафедри	30.05.2022	30.05.2022	виконано
9	Доробка та остаточне оформлення БКР	31.05.2022	24.06.2022	виконано
10	Подання БКР рецензенту	26.06.2022	26.06.2022	
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	26.06.2022	26.06.2022	
12	Захист БКР перед екзаменаційною комісією (ЕК)	29.06.2022	29.06.2022	

Розробив студент _____Єфремов Г.І._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи _____ст.викладач Кошовий В.В._____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 202__ р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи

студентки групи 402 ЧНУ ім. Петра Могили

Єфремова Георгія Івановича

Тема: «Система краудфандинга для студентських стартап-проектів»

Керівник: ст.викладач Кошовий Віталій Володимирович.

Об’єкт дослідження – збір фінансів та команди для реалізації стартапу.

Предмет дослідження – веб застосунок для студентських стартапів.

Мета – створення інструменту, що дозволятиме студентам більш легко створювати стартапи, шукати для них фінансування та знаходити необхідних людей, що можуть їм допомогти, з його реалізацією.

У першому розділі розглядається 11 У другому – 24 У третьому 56 В останньому спеціальному розділи було розглянуто нормативну базу охорони праці при користуванні екранними приладами.

У результаті виконання роботи було зроблено наступні висновки 67

Сторінок – 77 , таблиць – 2 , рисунків – 63 , посилань – 20 , додатків – 0

Ключові слова: студентський-стартап проект, стартап, база даних, краудфандинг, веб застосунок краудфандинга для студентських стартап-проектів.

ABSTRACT

for bachelor's qualification work

of a student of 402 group at Petro Mohyla Black Sea National University

Yefremov Heorhii Ivanovich

Topic: "Crowdfunding system for student startup projects"

Supervisor: senior teacher Koshoviy Vitaliy Volodymyrovych.

The object of the research is to raise funds and teams to implement a startup.

The subject of the research are methods of rapid fundraising for startups.

The goal is to create a tool that will allow students to more easily create startups, seek funding for them and find the right people who can help them with its implementation.

The first section presents 11 The second section describes 24 The third section specifies 56 In the last section the normative base of labor protection at use of screen devices was considered.

As the result of the performed work, conclusions 67

Pages – 77, tables – 2, figures – 63, links – 20, appendices – 0

Keywords: student-startup project, startup, database, crowdfunding, web crowdfunding application for student startup projects.

ЗМІСТ

ВСТУП 2
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ПОШУК ТЕОРЕТИЧНИХ ВІДОМОСТЕЙ . 3
1.1 Переваги та недоліки краудфандингу. 3
1.2 Приклади сайтів краудфандингу 6
1.3 Види краудфандингу 9
Висновки до розділу 1 11
2 ОБГРУНТУВАТИ ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РОЗРОБКИ 13
2.1 Мова програмування 13
2.2 Середовище розробки 14
2.3 База даних 17
2.4 Razor pages 18
2.5 Microsoft Identity 19
2.6 SmtpClient Class 20
2.7 Dependency Injection 21
2.8 LINQ 21
2.9 Entity Framework 22
Висновки до розділу 2 24
3 СТВОРЕННЯ ВЕБ ЗАСТОСУНКУ ДЛЯ КРАУДФАНДИНГУ СТУДЕНТІВ 26
3.1 Реалізація бази даних 26
3.2 Взаємодія із базою даних 34
3.3 Контролери 42
3.4 Frontend 45
3.5 Огляд проекту 48
Висновки до розділу 3 57
4. ОХОРОНА ПРАЦІ 59
Висновки до розділу 4 65
ВИСНОВКИ 67
СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ 69

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

CRUD – Create Read Update Delete

EF core – entity framework core

IDE – integrated development environment

LINQ – Language-Integrated Query

ORM – object-relational mapping

SMTP – Simple Mail Transfer Protocol

SQL – Structured Query Language

VR – virtual reality

Пояснювальна записка

до кваліфікаційної роботи

на тему:

СИСТЕМА КРАУДФАНДИНГА ДЛЯ СТУДЕНТСЬКИХ СТАРТАП-ПРОЕКТІВ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810210

Виконав студент 4-го курсу, групи 402
_____*Г. І. Єфремов*
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Керівник: *канд. техн. наук, доцент (б.в.з.)*
(наук. ступінь, вчене звання)
_____*В. В. Кошовий*
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Миколаїв – 2022

ВСТУП

Краудфандинг – це спосіб для багатьох людей зробити те, що зазвичай вони зробити не мають можливості. Загалом це збір грошей на якусь ідею яка здалась людям цікавою [1].

Багато стартапів змогли перерости у великі проекти саме завдяки краудфандингу. Людина, яка прагне використовувати краудфандинг зазвичай використовує такі сайти, щоб зібрати невеликі суми грошей від осіб, які, зазвичай, не є професійними фінансистами. Вони просто бачать цікаву ідею, та допомагають фінансово її реалізувати. Звичайно потім вони отримують або сподіваються отримати якусь вигоду.

У наш час є багато краудфандингових систем. Найпопулярніші серед них: Kickstarter, Ulule, Crowdculture, та інші.

У сфері краудфандингу є велике різноманіття систем, але немає системи, яка б розширювала можливості для студентів. Вони все ще можуть використовувати звичайні системи для збору коштів, але немає такої системи, де можна було б зібрати команду. Тому, було вирішено розробити веб-застосунок системи краудфандингу збору коштів та набору команди для студентів університетів.

Мета роботи : створення інструменту, що дозволятиме студентам більш легко створювати стартапи, шукати для них фінансування та знаходити необхідних людей, що можуть їм допомогти, з його реалізацією.

Об’єкт дослідження – збір фінансів та команди для реалізації стартапу.

Предмет дослідження – методи швидкого збору грошей задля стартапів.

Відповідно до мети виділені наступні **завдання** дипломної роботи:

- аналіз предметної області, пошук теоретичних відомостей;
- обґрунтувати вибір інструментальних засобів розробки;
- створення веб застосунку для краудфандингу студентів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ПОШУК ТЕОРЕТИЧНИХ ВІДОМОСТЕЙ

1.1 Переваги та недоліки краудфандингу.

Краудфандинг на основі винагород, незважаючи на свою більшу складність, має багато переваг перед альтернативними способами фінансування, такими як венчурний капітал або банківські позики,— він дає змогу творцям підтвердити свої ідеї щодо продуктів на ринку, створити базу клієнтів на ранніх стадіях та отримати цінні відгуки клієнтів [1].

Краудфандінговий вклад можна охарактеризувати як збір засобів для ділових цілей і не тільки. Через краудфандінгові платформи можна фінансувати різні типи проектів. У сфері економічної діяльності зазвичай інноваційні підприємства з високою ймовірністю отримання високого доходу в майбутньому мають шанси на приватне фінансування.

Загалом виділяють наступні переваги краудфандингу:

Він забезпечує доступ до капіталу. На початковому етапі підприємець може подумати, що поза своєю мережею він може залучити капітал лише від акредитованих інвесторів, венчурних капіталістів та банків. Це не правда. Краудфандинг — чудовий альтернативний спосіб фінансування підприємства, і це можна зробити, не відмовляючись від акцій та не накопичуючи борги. Платформи краудфандингу, що ґрунтуються на винагородах, дозволяють підприємцям збирати кошти від спільноти в обмін на просте надання своїх матеріальних продуктів або інших відносних подарунків.

Він хеджує ризики. Створення компанії – дуже ризикована та складна подорож. Крім пошуку достатнього фінансування, завжди є витрати, які неможливо передбачити, проблеми з перевіркою на ринку та інші люди, які хочуть отримати частину вашого підприємства, щоб допомогти запустити його. Запуск краудфандингової кампанії хеджує ці ризики і є цінним навчальним досвідом. Краудфандинг у його нинішньому вигляді дозволяє

підприємцю отримати схвалення ринку та уникнути втрати капіталу, перш ніж зробити все можливе та вивести концепцію продукту на ринок.

Мозковий штурм. Однією з найбільших проблем для малого бізнесу та підприємців є можливість закрити всі дірки, які можуть мати підприємство на ранній стадії. Проводячи краудфандингову кампанію, підприємець має можливість залучати натовп та отримувати коментарі, відгуки та ідеї. Цей зворотний зв'язок надзвичайно цінний, тому що може допомогти зрозуміти деякі аспекти їхнього бізнесу, про які раніше не думали. Це також може надихнути на інші ідеї!

Він знайомить із потенційними постійними клієнтами. Кампанія з краудфандингу не тільки дозволяє підприємцю уявити бізнес та продукт, але й дає йому можливість поділитися повідомленням та метою, що стоїть за ним. Люди, які спостерігають за кампанією підприємця і вирішили зробити свій внесок, вірять в успіх компанії в довгостроковій перспективі. По суті ці люди є першопроходьцями. Ранні послідовники є дуже важливими для будь-якого бізнесу, оскільки вони допоможуть поширити інформацію, не вимагаючи нічого натомість. Такі люди дбають про бренд та послання підприємства та, ймовірно, залишаться його лояльними клієнтами протягом усього терміну його існування.

Це безкоштовний піар. Імпульс, створений успішними краудфандинговими кампаніями, привертає потенційні інвестиції з традиційних каналів та увагу засобів масової інформації. Історії успіху цікаво читати, і репортери завжди прагнуть їх. У наші дні краудфандинг є унікальною та популярною галуззю, і незліченну кількість підприємців, які досягли в ньому успіху, в результаті досягли більшого успіху та популярності.

Надає можливість передпродажу. Запуск краудфандингової кампанії дає підприємцю можливість продати продукт або концепцію, які вони ще не вивели на ринок. Це хороший спосіб оцінити реакцію користувачів та

проаналізувати ринок, щоб вирішити, чи слід продовжувати чи орієнтуватися на цю концепцію.

Відсутність штрафів. На краудфандингових платформах «все чи нічого» (це означає, що ви отримуєте зібрані кошти лише у тому випадку, якщо ви досягаєте 100% або більше своєї мети з фінансування) є так багато переваг і немає плати за участь. Якщо підприємець ставить за мету і не досягає її, штрафу немає. Усі кошти повертаються кожному вкладнику, підприємець нічого не отримує, як і платформа. З іншого боку, якщо проект зі збору коштів успішний, усі виграють та стають частиною успішного краудфандингового проекту. У разі успіху середня комісія для платформ становить близько 5% загальної суми залучених коштів. По суті, краудфандинг — це чудовий спосіб для підприємців отримати фінансування та доступ до інформації, потрібних їм для перевірки, реалізації та розвитку своїх підприємств. Те, що почалося як соціальний експеримент кілька років тому, було перевірено як життєздатний інструмент тисяч людей. Краудфандинг зібрав понад 1,5 мільярда доларів для створення та розвитку підприємств на всіх етапах. Оскільки галузь все ще розвивається і стає все більш ефективною, зараз найкращий час, щоб скористатися супутніми перевагами!

Одним із найвідоміших краудфандингових порталів є Kickstarter, де з карманів приватних інвесторів уже профінансовані сотні бізнесів — з великим чи меншим успіхом. Крім того, завдяки «соціальному вкладу» розумні часи Pebble Technology Corporation отримали шанс побачити світ. Краудфандінг в кращому випадку вираховується мільйонами доларів [2].

Але не існую нічого безкоштовного у рамках краудфандингової платформи молоді підприємці можуть отримувати гроші від приватних інвестицій, однак це не благодійність, а інвестори не вкладають свій капітал тільки через те, що вірять на слово. Все добре читається і люди, які вкладають гроші в той чи інший проект, розраховують на подальшу фінансову віддачу.

Перевагою краудфандингу є те, що таким чином можна отримати необхідний капітал, не проходячи процедуру отримання кредиту. Часто малі та середні підприємства не мають достатньої кредитоспроможності, щоб взяти кредит у банку. Притягнення інституційних інвесторів також може бути проблематичним.

При краудфандингу досить добре підготувати рекламну кампанію ваших пропозицій і отримати дійсно інноваційний проект, щоб отримати бажану суму краудфандингу.

Недоліком краудфандингу є безпека бізнес-ідеї. Є необхідність поділитися їм з інтернет-спільнотою.

1.2 Приклади сайтів краудфандингу

Першою з таких платформ хотілося б розглянути Indiegogo(див. рис. 1.1). Заснована в 2008 році, платформа залишається одним з найпопулярніших місць в Інтернеті для фінансування проектів у всіх можливих сферах. Користуються ним як приватні особи, так і великі бренди, і щомісяця його відвідують понад 15 мільйонів користувачів. Найбільший проект Indiegogo - автоматизований вулик Flow Hive - зібрав понад 12 мільйонів доларів [3].

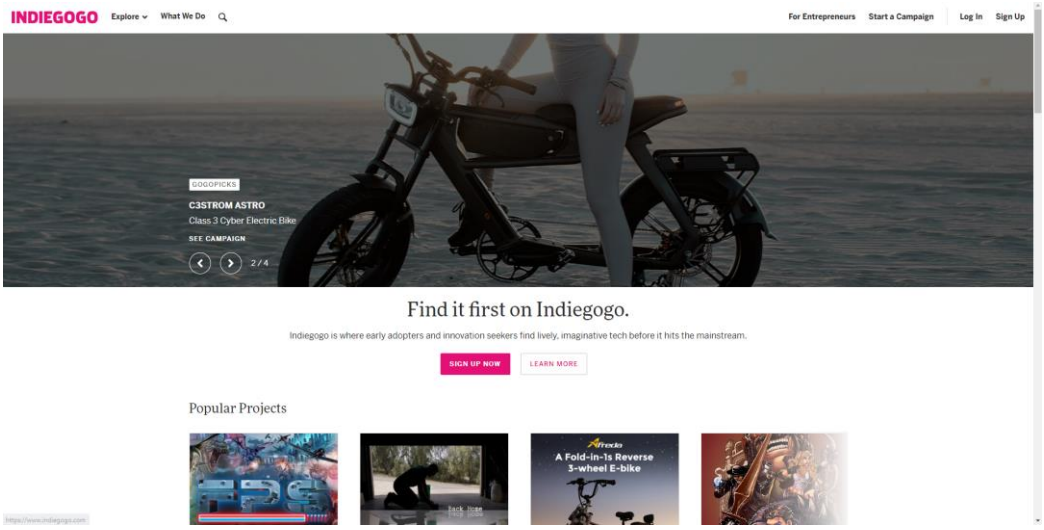


Рисунок 1.1 — Веб сторінка Indiegogo

Досить популярним також є Kickstarter(див. рис. 1.2). Понад 257 000 проектів у всіх можливих сферах, майже 9,5 мільйона користувачів, які їх підтримують, і майже 2 мільярди доларів загальна сума виплаченої суми - це Kickstarter в цифрах. Заснований у 2009 році Kickstarter став символом для стартап-спільноти – найбільшим проектом, реалізованим там, став розумний годинник Pebble Time, який отримав понад 20 мільйонів доларів [4].

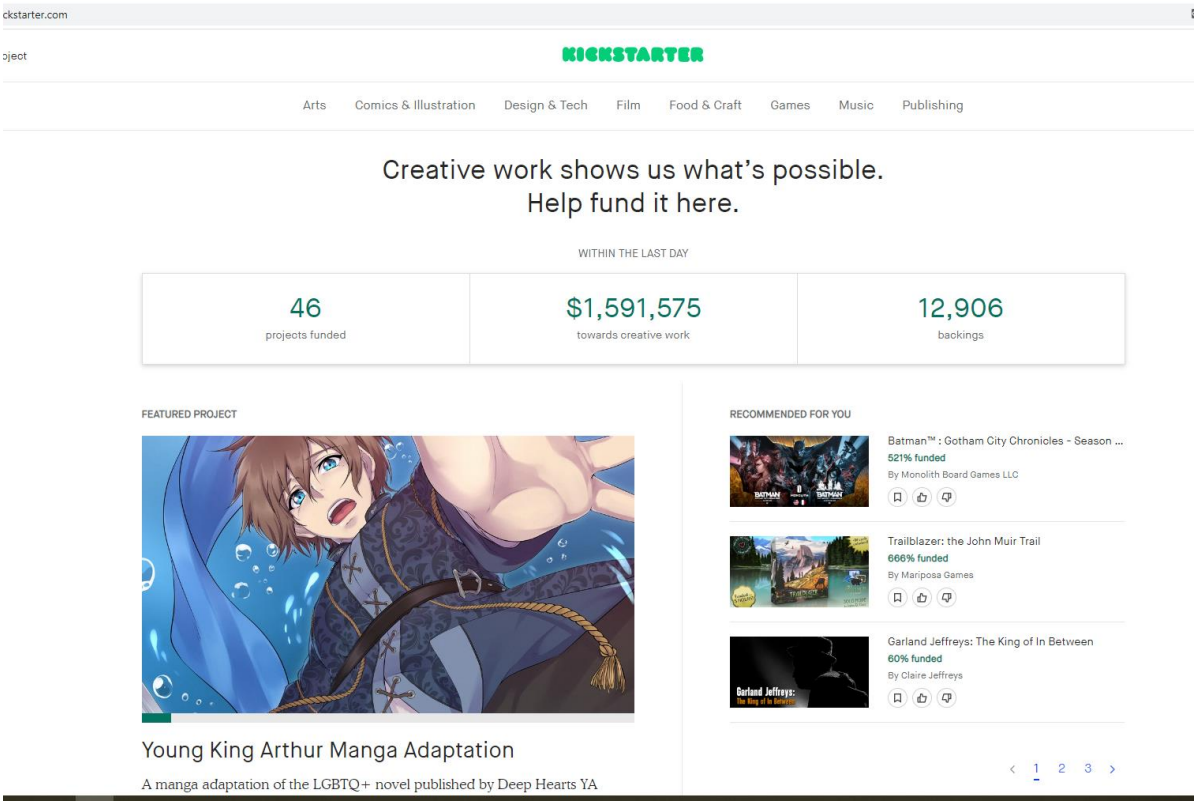


Рисунок 1.2 — Веб сторінка Kickstarter

Ще одна цікава платформа – Патреон(див. рис. 1.3). Створена в 2013 році, ця платформа дозволяє художникам з різних сфер збирати кошти. Він також популярний серед творців різноманітного веб-контенту. Зареєстровані там творці можуть розраховувати на регулярну, періодичну підтримку своїх шанувальників, а також мають можливість отримати конкретні суми за певну роботу. Однією з найвідоміших людей, які використовують цю форму фінансування, є Аманда Палмер [5].

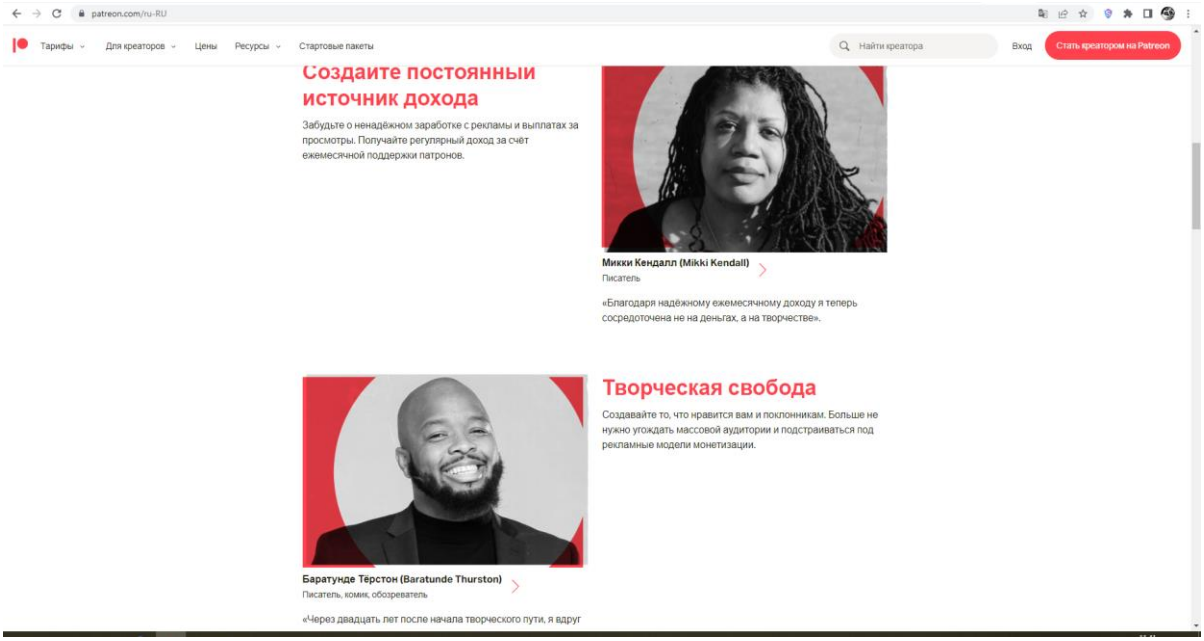


Рисунок 1.3 — Веб сторінка Патреон

Далі розглянемо Експеримент(див. рис. 1.4). Цей сайт був заснований у 2012 році колишніми дослідниками Вашингтонського університету під назвою Microryza. Ця досить нішова платформа спрямована на краудфандинг дослідницьких проектів. Він працює за моделлю «все або нічого», і на відміну від найпопулярніших краудфандингових сайтів, користувачі, які підтримують проекти, не отримують нагород. Проте вони можуть брати участь у кожному етапі дослідження, яке вони фінансують [6].

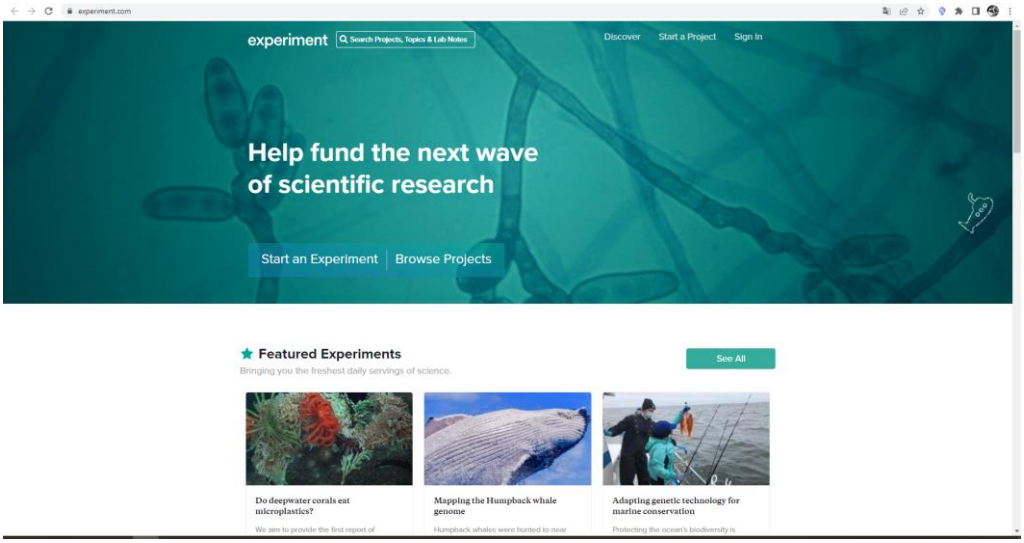


Рисунок 1.4 — Веб сторінка Експеримент

Також можна розглянути цікаву польську платформу Wspieram.to(див. рис. 1.5). Пплатформа, що працює з 2011 року, представляє себе як «польський Kickstarter та Indiegogo» – як і американські краудфандингові гіганти, вона дозволяє реалізовувати проекти в будь-якій сфері та нагороджує користувачів, які їх підтримують. Хоча спільнота supporting.to наразі сплатила майже 14 мільйонів злотих, лише 2 успішні кампанії перевищили суму в 500 тис. злотих. Злотих [7].

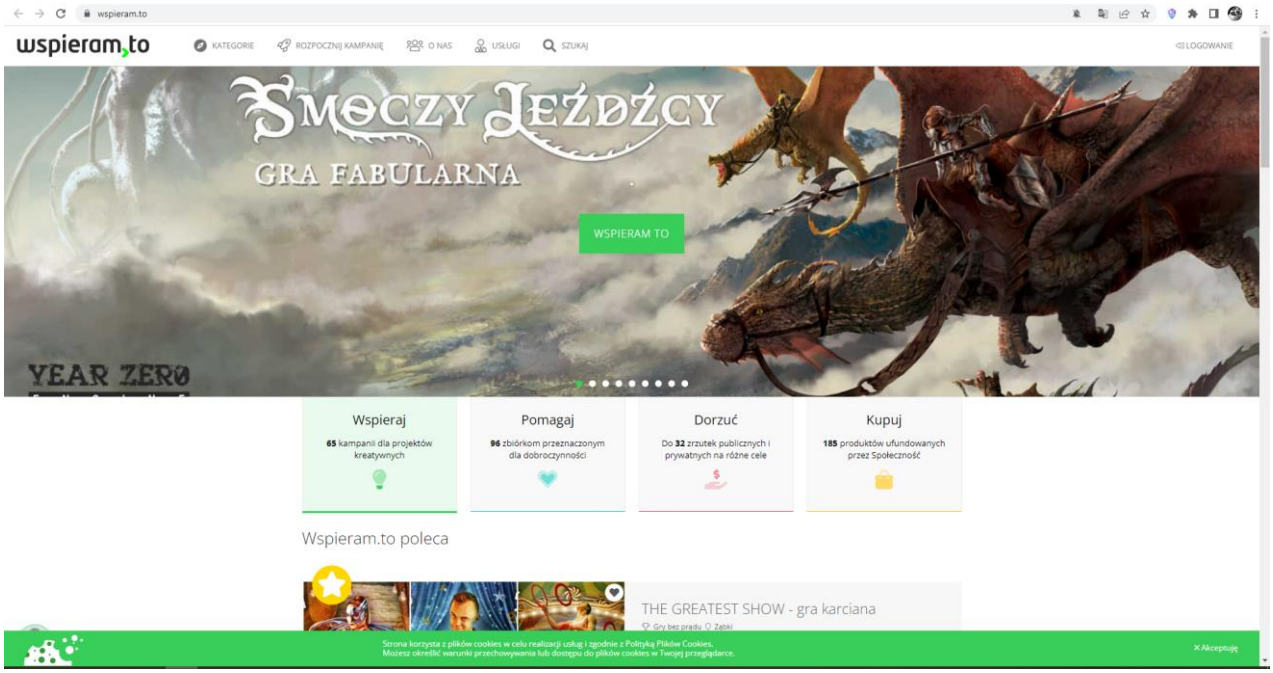


Рисунок 1.5 — Веб сторінка Wspieram.to

1.3 Види краудфандингу

Краудфандинг акцій (краудінвестінг) це рішення, в якому компанія (наприклад, стартап) або інша організація пропонує користувачам порталу частки у проекті в обмін на оплату. Виплата також пов'язана з наступним прибутком – у вигляді дивіденду. Це не завжди єдиний прибуток - часто компанії, які використовують цю форму краудфандингу, також пропонують інвесторам, які купують більшу кількість акцій, інші переваги (від знижок на продукти до можливості співпраці у їх створенні) – наприклад є платформи, такі як Beesfund, Findfunds чи Crowdway [2].

Все частіше краудфандинг акцій у ширшому масштабі, коли пропонуються більші пакети акцій, приваблює венчурні фонди та фонди акцій, а також інших потенційних інвесторів. Такий краудфандинг «вищого рівня» часто називають краудінвестінгом. Наприклад, Варшавська фондова біржа запустила свою платформу, призначену для брокерів.

Краудфандинг акцій, очевидно, найкраще працює у разі продуктів орієнтованих на широкі групи аудиторії. Наприклад, Pora na Pola, Coffeedesk та Kubota запропонували свої акції у формі краудфандингу. В даний час краудфандинг у цій формі також популярний в ігровій, харчовій та конопляній галузях.

Борговий краудфандинг аналогічний попередній формі краудфандингу, але в цьому випадку компанія не пропонує акції, а подає заявку на отримання кредиту, який має бути погашений протягом певного періоду (зазвичай 2-3 роки), переважно це форма соціального кредиту. але призначений для фірм. Прибутковість залежить від конкретних інвестицій, але зазвичай становить близько 7-9%.

З цією дією менше формальностей, ніж у разі отримання кредиту в банку, а платформа має контрольну функцію — так вона перевіряє заявку на юридичне обґрунтування, оцінює платоспроможність заявника, а потім встановлює процентну ставку. Платформи, що працюють за цією моделлю, це, наприклад, Crowdy або Mintos (які також пропонують соціальні кредити).

Краудфандинг на основі попереднього продажу це, мабуть, найвідоміший вид краудфандингу – за цією формулою працює найбільша краудфандингова платформа – кікстартер. У цій моделі компанія збирає кошти на реалізацію проекту або його окремих етапів, а після його створення постачає створений продукт чи послугу людям, які підтримали його розробку, і, можливо, починає продавати його на постійній основі. У цій формулі «колекції фізичних продуктів чи програмних рішень працюють найкраще».

Дуже схожою формою фінансування є краудфандинг на основі призів - тут донор не обов'язково отримує конкретний продукт, фінансування якого підтримується, але вони також можуть бути винагородами, пов'язаними з ним. Нагороди також засновані на підтримці онлайн-творців на таких платформах, як Patronite, де ті, хто підтримує конкретного виконавця, блогера або ютубера, можуть отримати, наприклад, подяку за фільм, доступ до контенту, недоступному для широкої публіки, або іншу подяку за їх підтримувати.

Краудфандинг пожертвувань не зовсім орієнтований на стартапи — він носить філантропічний характер та спрямований на підтримку хворих чи постраждалих. У цю категорію входять переважно благодійні збори коштів та ініціативи некомерційних організацій. Він носить філантропічний характер, тому донори не можуть розраховувати на будь-яку винагороду за надану підтримку.

Висновки до розділу 1

У ході дослідження переваг та недоліків краудфандингу було визначено, що **перевагами краудфандингу є** забезпечення доступу до капіталу, хеджування ризиків, мозковий штурм, знайомство із потенційними клієнтами, безкоштовний піар, можливість передпродажу, відсутність штрафів. **Недоліком краудфандінга є** безпека бізнес-ідеї, а саме необхідність поділитися нею з інтернет-спільнотою.

Також у ході дослідження сфери краудфандингу було визначено його **види**:

– краудфандинг акцій (краудінвестінг), в якому компанія або інша організація пропонує користувачам порталу частки у проекті в обмін на оплату;

– борговий краудфандинг, в якому компанія подає заявку на отримання кредиту, який має бути погашений протягом певного періоду;

– краудфандинговий кікстартер, під час якого компанія збирає кошти на реалізацію проекту, а після його створення постачає створений продукт чи послугу людям, які підтримали його розробку, і, можливо, починає продавати його на постійній основі.

Під час дослідження сфери було проаналізовано приклади сайтів краудфандингових проектів. У ході аналізу було визначено **основні функції цих сайтів**:

– реєстрація та авторизація на сайті;
– головна сторінка з активними проектами;
– перегляд опису конкретного стартапу;
– можливість редагування стартапів;
– можливість залишати коментарі до стартапів та відповідати на коментарі інших користувачів;
– функція збору коштів для стартапу.

2 ОБГРУНТУВАТИ ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РОЗРОБКИ

2.1 Мова програмування

Для написання веб застосунку було вибрано мову програмування C#. C# це універсальна, високорівнева, об'єктно-орієнтована мова програмування, яка є відповіддю Microsoft на Java. C# тісно інтегрований з платформою .NET, яка є як платформою, так і середовищем виконання. C# був створений і найчастіше використовується для написання програм для систем Windows, але оскільки .NET Framework знайшов свій шлях до систем Linux і Mac, можна створити рідне програмне забезпечення цією мовою практично для будь-якої платформи. Крім того, C# використовується для створення серверних веб-додатків за допомогою платформи ASP.NET. Завдяки таких інструментам, як Xamarin, можна створювати кросплатформні мобільні додатки. Більше того, C# використовується в популярному ігровому движку Unity, за допомогою якого можна створювати ігри для ПК, консолей, мобільних пристроїв або веб-сайтів і навіть платформ VR (віртуальна реальність). Тому це кросплатформна мова програмування, яка використовується при розробці даного веб застосунку. Також за допомогою C# було створено веб-сайт StackOverflow та такі ігри, як Terraria, Magicka, Bastion [9].

C# відносно легко вивчити. Для тих, хто стикався з такими мовами, як Java, синтаксис буде дуже знайомий. Семантика трохи дружніша, ніж C ++. Як і в Java, нам також не потрібно керувати пам’яттю, а процес розробки програмного забезпечення відбувається набагато швидше. Це також сильно типізована мова, що означає, що кожна змінна повинна мати оголошений збережений тип даних, і програма вийде з ладу та навіть не почне компіляцію, на відміну від мови зі слабким типом, яка не генеруватиме. Єдиним недоліком цього рішення є те, що ми повинні оголошувати більше інформації для

програми, але це робить її більш однозначною та легшою для виявлення помилок.

Далі хотілося б навести деякі статистичні данні. Станом на 31 березня 2022 року згідно з індексом TIOBE, C# є 5-ю за популярністю мовою, зберігаючи свої позиції з минулого року. У свою чергу, згідно з рейтингом Spectrum , він знаходиться на 6-й позиції, а рік тому був на 23-му місці. У Stack Overflow C# це четверта за кількістю тегів мова з 1 530 302 запитами. На Github він посідає 6-е місце з 480 745 проектами, 953 з яких мають понад 1000 зірочок. За даними Meetup.com , це 6-та за величиною спільнота з 760 групами та 636 119 членами по всьому світу. Окремо можна зазначити популярність движка Unity, який може похвалитися тим, що на цьому движку створено 34% найпопулярніших мобільних ігор. Це також позитивно впливає на розголос C#.

C# все ще розробляється Microsoft, яка не збирається припиняти його підтримку. Його остання версія 10.0 була випущена 8 листопада 2021 року. Движок для розробки ігор Unity все ще розробляється та вдосконалюється, тому це хороший вибір для тих, хто цікавиться розробкою ігор. Додатковою перевагою є те, що вона дозволяє писати програмне забезпечення для платформ VR (Virtual Reality), і ця технологія стає все більш популярною. На веб-сайті Unity ви можете побачити приблизні дані про те, що 90% ігор VR на Samsung Gear і 53% на Oculus Rift створено за допомогою Unity.

2.2 Середовище розробки

Під час розробки програмного застосунку нам необхідно використовувати якусь IDE, тобто Інтегроване середовище розробки. Це може бути навіть звичайний Блокнот, що є стандартною програмою windows. Проте звичайно працювати в такому середовищі не дуже зручно. Тому при розробці на мові C# найчастіше використовують такі IDE, як наприклад Visual Studio(див. рис. 2.1) та JetBrains Rider(див. рис. 2.2) [18].

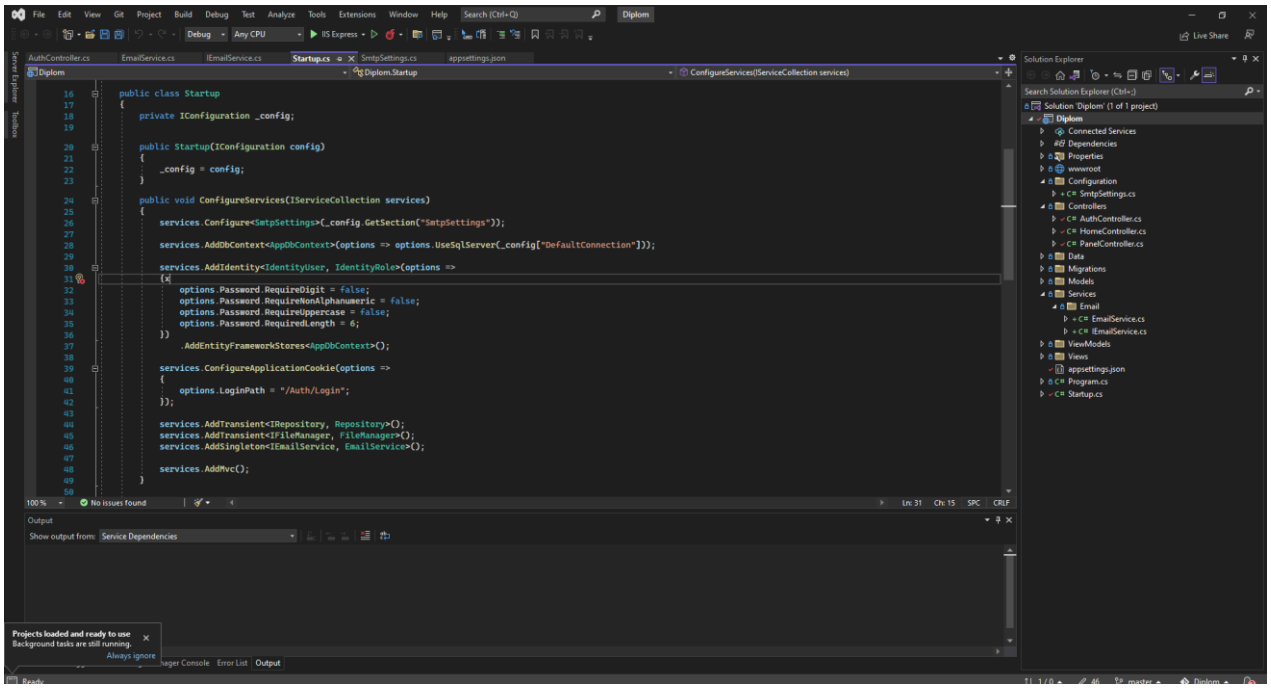


Рисунок 2.1 — IDE Visual Studio

На мою думку, як і багатьох інших людей, що програмують на C# Rider має дуже багато переваг над Visual Studio. Саме тому команди розробників переходять від Visual Studio до Rider. Ось декілька причин того, чому Rider є за багатьма пунктами кращою IDE ніж Visual Studio:

– Rider, на відміну від Visual Studio, не занурюється в 32-розрядні процеси. Навіть якщо у Rider є лише внутрішні процеси, такі як SWEA (Аналіз повного рішення), код буде створено гладко без будь-яких пауз і перешкод. І, як відзначають більшість користувачів, які працювали з Visual Studio і Rider, останній набагато стабільніший і швидший;

– JetBrains Rider є кросплатформним, він може працювати на платформах Windows, Mac або Linux з однаковою функціональністю та стабільністю. Visual Studio працює переважно на платформі Windows. А якщо ви хочете перейти на Linux або Mac, вам потрібно буде придбати додаткові рішення: Visual Studio Code (для Linux) і Visual Studio для Mac. Основним недоліком є те, що версії Visual Studio для Mac і Linux мають різну функціональність і дизайн, до яких потрібно звикнути. Rider, як зовні, так і на основі функцій, однаковий на всіх платформах, тому, якщо

користувач вирішить перейти з Windows, Mac або Linux, він знайде середовище розробки, з яким вони знайомі, і не витрачатиме дорогоцінний час на навчання;

– Rider містить більшість функцій, ніж популярне розширення Visual Studio для розробників .NET, ReSharper. У Rider ми маємо вражаючий набір рефакторингу, перевірки коду та контекстних дій для всіх мов і підтримуваних технологій. Visual Studio також має набір рефакторингу та перевірки помилок коду, але набагато обмеженіший, ніж запропонований Rider та ReSharper;

– варто звернути увагу на те, якими інструментами раніше користувалися користувачі JetBrains. Ті, хто вже знайомий з IntelliJ IDEA, WebStorm, DataGrip або іншими середовищами, перейдуть на Rider набагато швидше, ніж користувачі, які працювали лише з Visual Studio;

– і, нарешті, остання перевага: рішення та проекти, з якими працює JetBrains Rider, повністю сумісні з Visual Studio і не використовують власні формати.

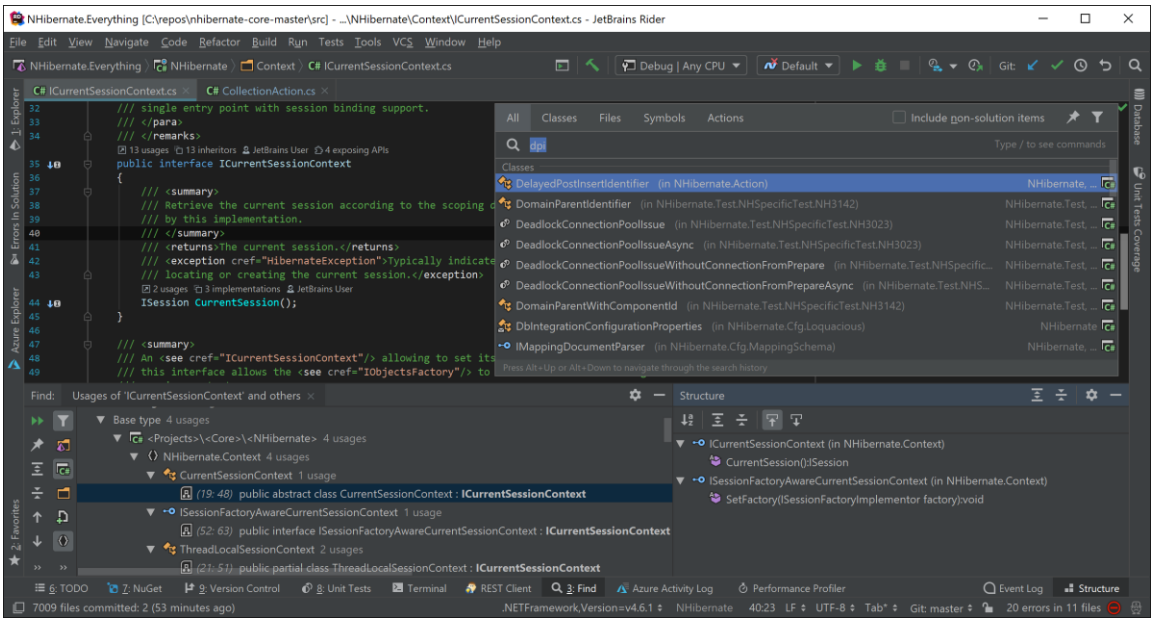


Рисунок 2.1 — IDE JetBrains Rider

Всі вище зазначені переваги зазвичай є достатнім аргументом для того, щоб перейти з Visual Studio на Rider. Але Visual Studio має одну перевагу, яка повністю нівелює усі переваги JetBrains Rider. Visual Studio – безкоштовна, а JetBrains Rider – ні(див. рис. 2.3).

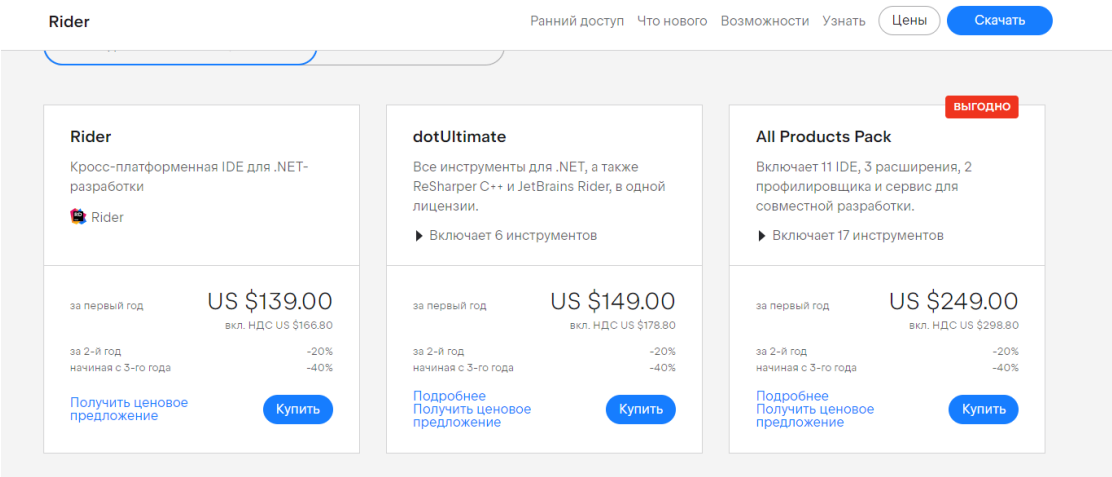


Рисунок 2.3 — Ціни на придбання програми JetBrains Rider

2.3 База даних

Для запису, читання та редагування даних у даному проекті було використано мову SQL, систему управління базами даних Microsoft SQL Server та SQL Server Management Studio у якості інтегроване середовище для управління всіма компонентами , що входять до складу Microsoft SQL Server.

SQL - структурована та декларативна мова запитів. Це мова домену, що використовується для створення, модифікації реляційних баз даних, а також для зберігання та отримання даних з цих баз даних [10].

Microsoft SQL Server - система управління базами даних , що підтримується і розповсюджується корпорацією Microsoft. Це основний продукт баз даних цієї компанії, який характеризується тим, що в якості мови запитів використовується Transact-SQL , що є розробкою стандарту ANSI / ISO.

MS SQL Server — це клієнт-серверна платформа баз даних [9] . У порівнянні з Microsoft Jet , який використовується в MS Access , він характеризується кращою продуктивністю, надійністю і масштабованістю.

SQL Server Management Studio - інтегроване середовище для управління всіма компонентами, що входять до складу Microsoft SQL Server. Це середовище має інструменти для налаштування, моніторингу та адміністрування екземплярів SQL Server. SQL Server Management Studio дозволяє будувати запити та сценарії, програма включає як редактор сценаріїв, так і графічні інструменти. Програма вперше з'явилася разом із Microsoft SQL Server 2005. Основною особливістю програми є Object Explorer , що дозволяє переглядати, вибирати та виконувати різноманітні дії над об'єктами сервера. Програмне забезпечення має безкоштовну «експрес-версію», яка трохи обмежена в порівнянні з повною версією – не можна керувати аналітичними , звітними та інтеграційними службами.

2.4 Razor pages

З випуском нової платформи ASP.NET Core 2 Microsoft та її спільнота надали нам абсолютно нову альтернативу підходу MVC (Model-View-Controller). Microsoft назвала його Razor Pages, і хоча це трохи інший підхід, але в чомусь він все ще схожий на MVC. Сторінка Razor дуже схожа на компонент перегляду ASP.NET MVC. Він має в основному такий же синтаксис і функціональність, що і MVC. Ключова відмінність сторінок Razor від MVC полягає в тому, що код моделі та контролера також міститься в самій сторінці Razor. Простіше кажучи, це дуже схоже на фреймворк MVVM. Він забезпечує двостороннє прив'язування даних і спрощує розробку з поодинокими проблемами. Хоча MVC чудово працює з веб-програмами, які мають велику кількість динамічних переглядів сервера, односторінковими програмами, REST API та викликами AJAX, але Razor Pages ідеально підходить для простих сторінок, які доступні лише для читання або вводять основні дані. Тепер

ASP.NET MVC став надзвичайно популярним для розробки веб-додатків, і він, безумовно, має свої переваги. Фактично, ASP.NET WebForms був спеціально розроблений як рішення MVVM в MVC. Але нові сторінки ASP.NET Core Razor – це наступна еволюція ASP.NET WebForms.

MVC це архітектурний шаблон, який використовується при розробці програмного забезпечення для реалізації користувацьких інтерфейсів. Хоча MVC є одним з найпопулярніших фреймворків і використовується мільйонами веб-розробників по всьому світу, але він все ще має свої недоліки. Одним із них є його складність. У ASP.NET MVC є купа концепцій, таких як TempData, RouteCollection, ViewData, Linq to SQL, дія контролера, лямбда-вираз, користувацький маршрут і HTML-помічники, які пов’язують модель, перегляд і контролер. Також у Razor Pages файли в основному більш організовані. У вас є Razor View і весь код за файлом, так само, як старі веб-форми ASP.NET.

2.5 Microsoft Identity

Microsoft Identity Web — це набір бібліотек ASP.NET Core, які спрощують додавання підтримки автентифікації та авторизації до веб-програм і веб-API [9]. Платформа ідентифікації Microsoft складається з кількох компонентів:

– Служба автентифікації, сумісна зі стандартом OAuth 2.0 і OpenID Connect, що дозволяє розробникам вставити кілька типів ідентифікаційних даних, зокрема: робочі або навчальні облікові записи, надані через Azure AD, особистий обліковий запис Microsoft, наприклад Skype, Xbox і Outlook.com, соціальні або локальні облікові записи за допомогою Azure AD B2C;

– Бібліотеки з відкритим кодом : бібліотеки автентифікації Microsoft (MSAL) та підтримка інших бібліотек, що відповідають стандартам;

– Портал керування додатками: можливість реєстрації та налаштування на порталі Azure, а також інші можливості керування Azure;

– API конфігурації додатків і PowerShell : програмна конфігурація ваших додатків за допомогою API Microsoft Graph і PowerShell, щоб ви могли автоматизувати свої завдання DevOps;

– вміст розробника: технічна документація, включаючи швидкі інструкції, навчальні посібники, інструкції та зразки коду.

Для розробників платформа ідентифікації Microsoft пропонує інтеграцію сучасних інновацій у сфері ідентифікації та безпеки, таких як автентифікація без пароля, поетапна автентифікація та умовний доступ. За допомогою платформи ідентифікації Microsoft можна написати код один раз і отримати доступ до будь-якого користувача. Є створити програму один раз, щоб вона працювала на багатьох платформах, або створити програму, яка функціонує як клієнтська, а також як ресурсна програма (API).

2.6 SmtpClient Class

Цей клас дозволяє програмам надсилати електронну пошту за допомогою протоколу Simple Mail Transfer Protocol (SMTP). Клас SmtpClientзастарів у Xamarin. однак:

– він входить до .NET Standard 2.0 і новіших версій, і тому має бути частиною будь-якої реалізації .NET, яка підтримує ці версії;

– він присутній і може використовуватися в .NET Framework 4 – .NET Framework 4.8.

Конструктор MailMessage приймає два об'єкти класу MailAddress як адресу відправника та адресу одержувача. SmtpClient надсилає та отримує електронну пошту. Існує лише чотири кроки, щоб надіслати простий електронний лист за допомогою класу SmtpClient. Нижче наведено процедуру відправки простого електронного листа:

– спочатку необхідно вказати назву SMTP-сервера;
– далі треба надати конкретні облікові дані SmptServer;
– потім створити повідомлення електронної пошти;
– і надіслати повідомлення електронною поштою.

2.7 Dependency Injection

Dependency Injection - це шаблон проектування, в якому об'єкт не ініціює свої залежності сам, а приймає їх ззовні [20]. Основна перевага такого підходу полягає в тому, що легше писати модульні застосунки. Використання Dependency Injection також може зробити наші об’єкти слабо зв’язаними. Такі об'єкти пов'язані між собою через абстракції, наприклад, через інтерфейси, що робить всю систему гнучкішою, адаптованішою і розширюваною. Нерідко для встановлення залежностей у подібних системах використовуються спеціальні контейнери – IoC-контейнери (Inversion of Control). Такі контейнери є свого роду фабриками, які встановлюють залежності між абстракціями та конкретними об'єктами і, як правило, керують створенням цих об'єктів. І якщо раніше в ASP.NET 4 та інших попередніх версіях треба було використовувати різні зовнішні IoC-контейнери для встановлення залежностей, такі як Ninject, Autofac, Unity, Windsor Castle, StructureMap, то ASP.NET Core вже має вбудований контейнер для впровадження залежностей, який представлений інтерфейсом IServiceProvider . А самі залежності ще називаються сервісами, тому контейнер можна назвати провайдером сервісів . Цей контейнер відповідає за зіставлення залежностей з конкретними типами та за впровадження залежностей у різні об'єкти. За встановлення сервісів у застосунку відповідає метод ConfigureServices , визначений у класі Startup.

2.8 LINQ

LINQ представляє просту та зручну мову запитів до джерела даних [9]. Як джерело даних може бути об'єкт, що реалізує інтерфейс IEnumerable

(наприклад, стандартні колекції, масиви), набір даних DataSet, документ XML. Але незалежно від типу джерела LINQ дозволяє застосувати до всіх той самий підхід для вибірки даних. Існує кілька різновидів LINQ:

– LINQ to Objects : застосовується для роботи з масивами та колекціями;
– LINQ to Entities : використовується при зверненні до баз даних через технологію Entity Framework;
– LINQ to XML : застосовується під час роботи з файлами XML;
– LINQ to DataSet : застосовується під час роботи з об'єктом DataSet;
– Parallel LINQ (PLINQ) : використовується для виконання паралельних запитів.

Крім стандартного синтаксису from .. in .. selectдля створення запиту LINQ, ми можемо застосовувати спеціальні методи розширення, які визначені для інтерфейсу IEnumerable. Як правило, ці методи реалізують ту ж функціональність, що і оператори типу LINQ where або orderby.

2.9 Entity Framework

Entity Framework представляє спеціальну об'єктно-орієнтовану технологію на базі фреймворку .NET для роботи з даними [19]. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework є вищим рівнем абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. Якщо фізично ми оперуємо таблицями, індексами, первинними та зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами. Перша версія Entity Framework – 1.0 вийшла ще у 2008 році та представляла дуже обмежену функціональність, базову підтримку ORM (object-relational mapping – відображення даних на реальні об'єкти) та один єдиний підхід до взаємодії з бд – Database First. З виходом версії 4.0 у 2010 році багато що

змінилося – з цього часу Entity Framework став рекомендованою технологією для доступу до даних, а в сам фреймворк були введені нові можливості взаємодії з бд – підходи Model First та Code First. Додаткові покращення функціоналу були з виходом версії 5.0 у 2012 році. І нарешті, в 2013 році був випущений Entity Framework 6.0, який має можливість асинхронного доступу до даних. Центральною концепцією Entity Framework є поняття сутності чи entity. Сутність є набір даних, асоційованих з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами та їх наборами. Будь-яка сутність, як і будь-який об'єкт із реального світу, має ряд властивостей. Наприклад, якщо сутність описує людину, ми можемо виділити такі властивості, як ім'я, прізвище, зростання, вік, вагу. Властивості необов'язково представляють прості дані типу int, але можуть представляти більш комплексні структури даних. І в кожної сутності може бути одна або кілька властивостей, які відрізнятимуть цю сутність від інших і унікально визначатимуть цю сутність. Подібні властивості називають ключами . При цьому сутності можуть бути пов'язані асоціативним зв'язком один-до-багатьом, один-до-одному і багато-багатьом, подібно до того, як у реальній базі даних відбувається зв'язок через зовнішні ключі. Відмінністю Entity Framework є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо не тільки витягувати певні рядки, що зберігають об'єкти з бд, але й отримувати об'єкти, пов'язані різними асоціативними зв'язками. Іншим ключовим поняттям є Entity Data Model . Ця модель зіставляє класи сутностей із реальними таблицями у БД. Entity Data Model складається з трьох рівнів: концептуального, рівень сховища та рівень зіставлення (мапінгу). На концептуальному рівні відбувається визначення класів сутностей, які у додатку. Рівень сховища визначає таблиці, стовпці, відносини між таблицями та типи даних, з якими зіставляється база даних, що використовується. Рівень зіставлення служить посередником між попередніми двома, визначаючи зіставлення між властивостями класу сутності та

стовпцями таблиць. Таким чином ми можемо через класи, визначені в додатку, взаємодіяти з таблицями з бази даних. Entity Framework передбачає три можливі способи взаємодії з базою даних:

– Database first : Entity Framework створює набір класів, що відображають модель конкретної бази даних;
– Model first : спочатку розробник створює модель бази даних, за якою Entity Framework створює реальну базу даних на сервері;
– Code first : розробник створює клас моделі даних, які будуть зберігатися в бд, а потім Entity Framework за цією моделлю генерує базу даних та її таблиці.

Висновки до розділу 2

У цьому розділі були розглянуті основні інструменти та технології, які використалися під час створення веб застосунку.

Для написання веб застосунку **було обрано мову програмування C#**. C# - це універсальна, високорівнева, об'єктно-орієнтована мова програмування. C# тісно інтегрований з платформою .NET, яка є як платформою, так і середовищем виконання.

Після порівняння середовищ розробки Visual Studio і JetBrains Rider було визначено, що Rider має більше переваг, ніж Visual Studio, але Visual Studio має одну перевагу, яка повністю нівелює усі переваги JetBrains Rider. Visual Studio – безкоштовна. Тому, **як середовище розробки проекту було обрано Visual Studio**.

Для запису, читання та редагування даних у даному проекті було використано мову SQL, і, **в якості системи управління базами даних, було обрано Microsoft SQL Server** та **SQL Server Management Studio у якості інтегрованого середовища** для управління всіма компонентами , що входять до складу Microsoft SQL Server.

При розробці проекту **були використані такі технології, як**:

– **Razor pages**, фреймворк, який забезпечує двостороннє прив’язування даних і спрощує розробку з поодинокими проблемами. Також у Razor Pages файли в основному більш організовані. У Razor View є весь код за файлом, так само, як старі веб-форми ASP.NET;

– **Microsoft identity**, набір бібліотек ASP.NET Core, які спрощують додавання підтримки автентифікації та авторизації до веб-програм і веб-API;

– **SmtpClient Class**, клас, який дозволяє програмам надсилати електронну пошту за допомогою протоколу Simple Mail Transfer Protocol (SMTP);

– **Dependency Injection**, шаблон проектування, в якому об'єкт не ініціює свої залежності сам, а приймає їх зовні, основною перевагою якого є полегшене написання модульних застосунків;

– **LINQ**, який представляє просту та зручну мову запитів до джерела даних;

– **Entity Framework**, спеціальна об'єктно-орієнтована технологія на базі фреймворку .NET для роботи з даними.

3 СТВОРЕННЯ ВЕБ ЗАСТОСУНКУ ДЛЯ КРАУДФАНДИНГУ СТУДЕНТІВ

3.1 Реалізація бази даних

При створенні бази даних використовувався підхід code first. У цьому проекті для зберігання в базі даних необхідними були наступні моделі: власне стартапи, та коментарі до них. Отже у проекті було створено папку models(див. рис. 3.1), котра зберігала у собі усі ці сутності.

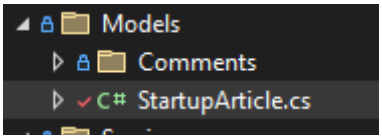


Рисунок 3.1 — Папка з моделями

Далі необхідно було створити клас для стартапів, але у всіх програмах на .NET Core є клас Startup, що є вхідною точкою до ASP.NET Core та здійснює конфігурацію програми, налаштовує сервіси, які програма буде використовувати, встановлює компоненти для обробки запиту або middleware. Тому клас що буде відображати наші стартапи було названо StartupArticle(див. рис. 3.2). Цей клас має наступні поля, що необхідні нам для роботи: поле ідентифікатор, назву, основну частину, поле що зберігає назву картинки, яка буде відображати стартап, опис, теги, категорію, до якої відноситься стартап, дату створення, та типізований класом MainComment список коментарів. Окремо можна зазначити, що поле дати створення одразу приймає значення поточного часу.

```
using Diplom.Models.Comments;
using System;
using System.Collections.Generic;

namespace Diplom.Models
{
    20 references
    public class StartupArticle
    {
        9 references | 0 exceptions
        public int Id { get; set; }
        6 references | 0 exceptions
        public string Title { get; set; } = "";
        3 references | 0 exceptions
        public string Body { get; set; } = "";
        6 references | 0 exceptions
        public string Image { get; set; } = "";
        3 references | 0 exceptions
        public string Description { get; set; } = "";
        3 references | 0 exceptions
        public string Tags { get; set; } = "";
        4 references | 0 exceptions
        public string Category { get; set; } = "";
        0 references | 0 exceptions
        public DateTime Created { get; set; } = DateTime.Now;
        5 references | 0 exceptions
        public List<MainComment> MainComments { get; set; }
    }
}
```

Рисунок 3.2 — Клас StartupArticle

Далі необхідно відкрити консоль менеджеру пакетів, це робиться через вкладку view(див. рис. 3.3)

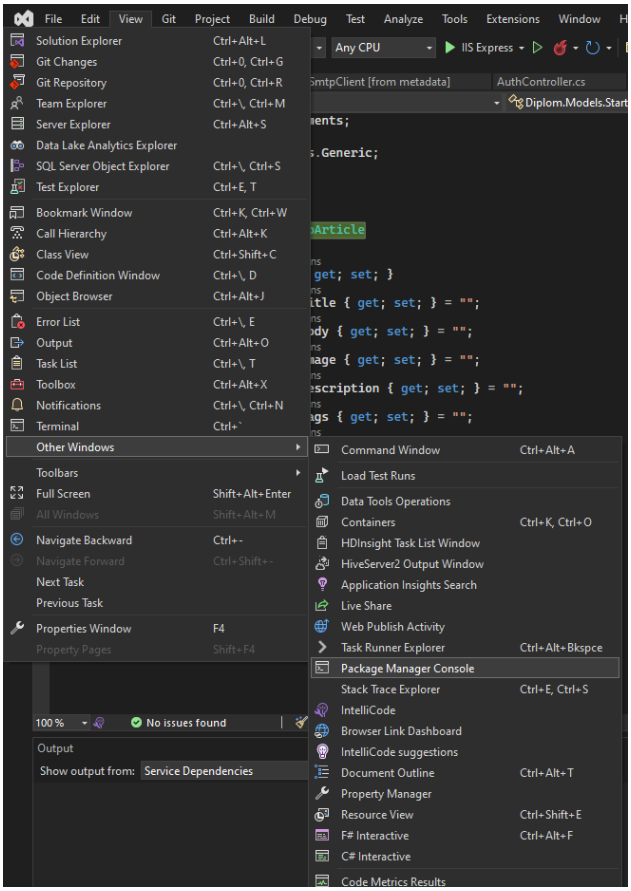


Рисунок 3.3 — Консоль менеджеру пакетів

Після цього створюємо міграцію(див. рис. 3.4), та оновлюємо базу даних(див. рис. 3.5).

```
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant a
packages may include dependencies which are governed by additional licenses. Follow the package

Package Manager Console Host Version 5.4.0.6292

Type 'get-help NuGet' to see all available NuGet commands.

PM> add-migration InitialCreate
Build started...
```

Рисунок 3.4 — Створення міграції

```
Build succeeded.
To undo this action, use Remove-Migration.
PM> update-database
Build started...
Build succeeded.
```

Рисунок 3.5 — Оновлення БД

Далі автоматично створюється папка migrations(див. рис. 3.6), у якій ми маємо всі наші міграції. За час написання веб застосунку було створено багато міграцій, які створювали нові та редагували вже створені таблиці у базі даних.

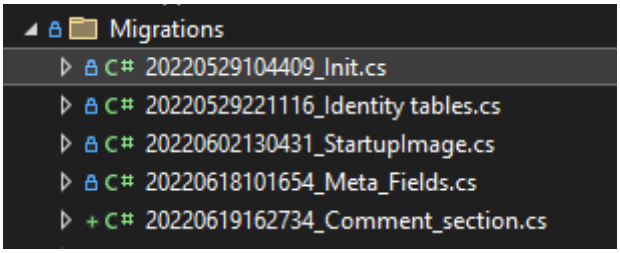


Рисунок 3.6 — Міграції

У якості приклада розглянемо першу міграцію. Будь які міграції, які створені автоматично чи написані вручну повинні мати два основних методи. Перший це метод Up(див. рис. 3.7), у якому за допомогою об’єкту класу MigrationBuilder ми можемо взаємодіяти із базою даних. А саме наприклад створювати таблицю за допомогою методу CreateTable, створити колонки таблиці наступним чином: Title = table.Column<string>(nullable: true), задати первинний ключ методом PrimaryKey, тощо.

```
protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.CreateTable(
        name: "Startups",
        columns: table => new
        {
            Id = table.Column<int>(nullable: false)
                .Annotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn),
            Title = table.Column<string>(nullable: true),
            Body = table.Column<string>(nullable: true),
            Created = table.Column<DateTime>(nullable: false)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_Startups", x => x.Id);
        });
}
```

Рисунок 3.7 — Метод Up

Ще крім методу Up кожна міграція повинна мати метод Down(див. рис. 3.8). На той випадок, якщо під час створення нової таблиці, редагуванні існуючої, додаванні зовнішніх ключів, чи якихось інших операцій із базою даних була допущенна помилка, чи з’явилась якась інша причина відмінити останні внесені зміни, нам необхідний саме цей метод. Він також, як і метод Up приймає у якості параметра об’єкт класу MigrationBuilder. Далі за допомогою методів цього класу ми можемо відмінити внесені зміни. Наприклад метод DropTable, що приймає у якості параметру назву таблиці, повністю її видаляє.

```
0 references | 0 exceptions
protected override void Down(MigrationBuilder migrationBuilder)
{
    migrationBuilder.DropTable(
        name: "Startups");
}
```

Рисунок 3.8 — Метод Down

Далі після того, як ми створили міграцію, та оновили базу даних ми можемо побачити результат через Microsoft SQL Management Studio(див. рис. 3.9). Після виконання всіх вищеописаних дій було створено нову базу даних із назвою Startup, у якій після всіх виконаних міграцій були таблиці, що зберігають у собі данні про студентські стартап-проекти, інформацію про користувачів, також тут є й окремі таблиці із ролями користувачив, які створив Microsoft Identity, та таблиці із коментарями.

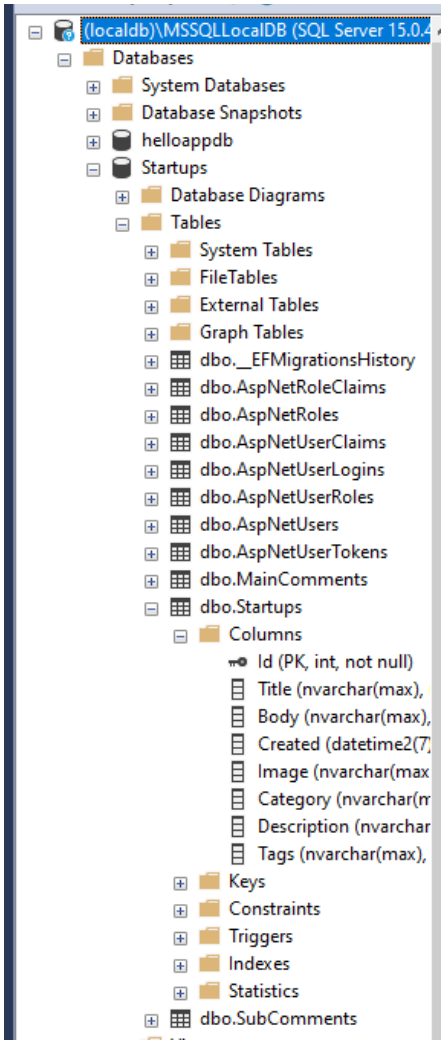


Рисунок 3.9 — База даних

Як можна помітити таблиць із коментарями – дві. Це зроблено для того щоб після написання коментаря до стартапу, до нього можна було написати відповідь. Отже один стартап може мати багато коментарів, а один коментар в свою чергу може мати багато відповідей. Це означає що ці сутності повинні мити відношення один до багатьох. Також коментарі та відповіді до коментарів, або ж підкоментарі мають дуже багато спільних полів із коментарями. Тому можна зробити клас від якого обидві ці сутності унаслідуються – базовий клас Comments(див. рис. 3.10). Тим самим ми реалізуємо один із основних принципів ООП – наслідування. Наслідування, разом з інкапсуляцією та поліморфізмом, є однією з трьох основних парадигм об'єктно-орієнтованого програмування. Наслідування дозволяє створювати

нові класи, які повторно використовують, розширюють та змінюють поведінку, визначену в інших класах. Клас, члени якого успадковуються, називається базовим класом , а клас, який успадковує ці члени, називається похідним класом. Оскільки в C# множинне наслідування класів не можливе, то похідний клас може мати лише один прямий базовий клас. Проте успадкування є транзитивним. Це означає, що якщо клас С є похідним від класу Б, а клас Б — від класу А, то клас С успадковує усі поля, оголошені як у класі Б так і у класі А.

```
using System;

namespace Diplom.Models.Comments
{

    public class Comment
    {
        1 reference | 0 exceptions
        public int Id { get; set; }
        4 references | 0 exceptions
        public string Message { get; set; }
        4 references | 0 exceptions
        public DateTime Created { get; set; }
    }
}
```

Рисунок 3.10 — Базовий клас Comment

Далі необхідно створити клас коментаря, до якого потім можна буде писати відповіді. Нехай цей клас буде мати назву MainComment(див. рис. 3.11).

```
using System.Collections.Generic;

namespace Diplom.Models.Comments
{
    4 references
    public class MainComment : Comment
    {
        2 references | 0 exceptions
        public List<SubComment> SubComments { get; set; }
    }
}
```

Рисунок 3.11 — Похідний клас MainComment

Клас, для відповідей на коментарі також є похідним від базового класу Comment. Щоб відокремити його, та підкреслити те, що він є додатком до основних коментарів назвемо цей клас SubComment(див. рис. 3.12).

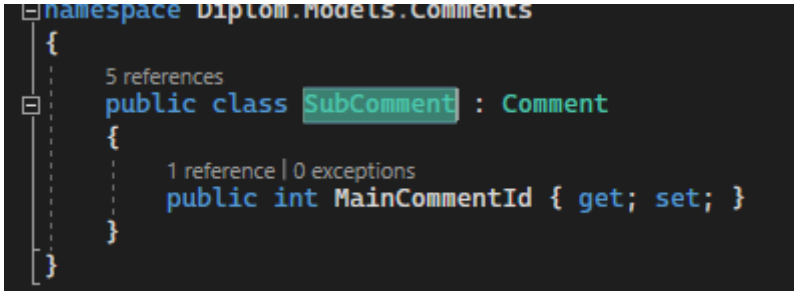


Рисунок 3.12 — Похідний клас SubComment

Як можна побачити зі скріншотів кожен із цих класів має лише по одному полю. Ці поля необхідні для реалізації одного із реляційних відношень. Загалом таких відношень є три типи:

– один до одного;
– один до багатьох;
– багато хто до багатьох.

У даному випадку використовується відношення один до багатьох, адже до одного коментаря можна написати багато відповідей, а кожному стартапу можна написати багато коментарів. Відношення один до багатьох має місце, коли одного запису батьківської таблиці може відповідати кілька записів у дочірній таблиці. Зв'язок один до багатьох є найпоширенішим для реляційних баз даних.

Для того, щоб під час міграції було створено відповідне відношення у моделі MainComment необхідно створити поле типом якого буде параметризований класом SubComment список. В свою чергу клас SubComment має містити поле, що буде відображати ідентифікатор відповідного “головного коментаря”. Типом такого поля має бути int.

Далі під час створення міграції буде автоматично створено дві нових таблиці. Зовнішні ключі, що необхідні для реалізації відношення один до багатьох також буде створено автоматично(див. рис. 3.13).

```
public partial class Comment_section : Migration
{
    0 references | 0 exceptions
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.CreateTable(
            name: "MainComments",
            columns: table => new
            {
                Id = table.Column<int>(nullable: false)
                    .Annotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn),
                Message = table.Column<string>(nullable: true),
                Created = table.Column<DateTime>(nullable: false),
                StartupArticleId = table.Column<int>(nullable: true)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_MainComments", x => x.Id);
                table.ForeignKey(
                    name: "FK_MainComments_Startups_StartupArticleId",
                    column: x => x.StartupArticleId,
                    principalTable: "Startups",
                    principalColumn: "Id",
                    onDelete: ReferentialAction.Restrict);
            });

        migrationBuilder.CreateTable(
            name: "SubComments",
            columns: table => new
            {
                Id = table.Column<int>(nullable: false)
                    .Annotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn),
                Message = table.Column<string>(nullable: true),
                Created = table.Column<DateTime>(nullable: false),
                MainComment = table.Column<int>(nullable: false),
                MainCommentId = table.Column<int>(nullable: true)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_SubComments", x => x.Id);
                table.ForeignKey(
                    name: "FK_SubComments_MainComments_MainCommentId",
                    column: x => x.MainCommentId,
                    principalTable: "MainComments",
                    principalColumn: "Id",
                    onDelete: ReferentialAction.Restrict);
            });

        migrationBuilder.CreateIndex(
            name: "IX_MainComments_StartupArticleId",
            table: "MainComments",
            column: "StartupArticleId");

        migrationBuilder.CreateIndex(
            name: "IX_SubComments_MainCommentId",
            table: "SubComments",
            column: "MainCommentId");
    }
```

Рисунок 3.13 — Міграція Comment_section

Як результат отримано дві таблиці, що мають зовнішні ключі(див. рис. 3.14).

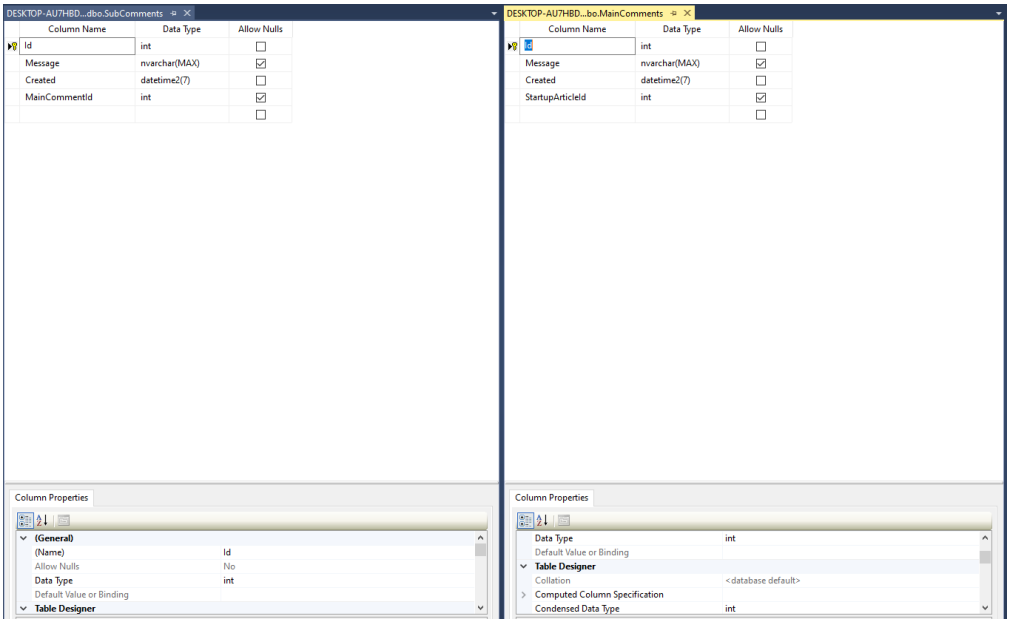


Рисунок 3.14 — Таблиці коментарів у БД

3.2 Взаємодія із базою даних

Щоб користуватися базою даних у проекті її спочатку необхідно підключити. Для цього використовуємо метод сервісу AddDbContext, параметризований створеним нами класом AppDbContext, у якому знаходяться ті моделі, які ми хочемо бачити в базі даних(див. рис. 3.15).

```
0 references | 0 exceptions
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<SmtpSettings>(_config.GetSection("SmtpSettings"));

    services.AddDbContext<AppDbContext>(options => options.UseSqlServer(_config["DefaultConnection"]));
```

Рисунок 3.15 — Підключення БД

Строка підключення зберігається в окремому файлі appsettings(див. рис. 3.16).

```
appsettings.json  Startup.cs  AppDbContext.cs
Schema: https://json.schemastore.org/appsettings.json
{
  "DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=Startups;Trusted_Connection=True;MultipleActiveResultSets=true",
  "Path": {
    "Images": "wwwroot/content/startup/"
  },

  "SmtpSettings": {
    "From": "studentStartup@gmail.com",
    "Server": "test",
    "Username": "testuser",
    "Password": "testpass"
  }
}
```

Рисунок 3.16 — Файл appsettings.json

Далі необхідно зазначити, що будь-яка програма що працює з базами даних повинна складатися з декількох слоїв. Оди відповідає за взаємодію із користувачем, другий при наявності за бізнес логіку, третій за взаємодію із базою даних. Для того щоб відокремити рівень взаємодії з базою даних створюється папка із назвою Data. В цій папці знаходиться папка Repository, папка FileManager, та клас AppDbContext(див. рис. 3.17).

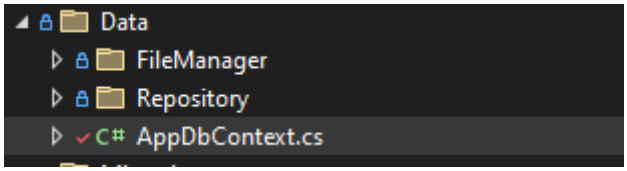


Рисунок 3.17 — Папка Data

У папці Repository знаходиться інтерфейс IRepository, та клас що його реалізує Repository. Це зроблено для реалізації патерну Dependency Injection.

Інтерфейс IRepository має сигнатури всіх методів, які нам будуть необхідні для операцій, які ми будемо проводити над коментарями, та стартап-проектами(див. рис. 3.18).

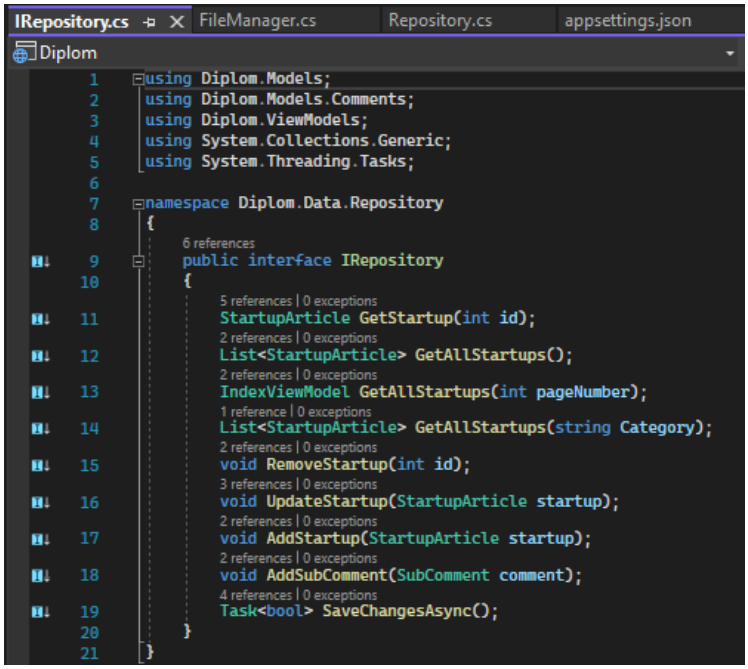


Рисунок 3.18 — Інтерфейс IRepository

Як можна побачити ту використовуються звичайні CRUD операції, тобто отримання даних, створення, оновлення та видалення. Entity Framework дозволяє легко виконувати дані операції.

Реалізує всі ці методи клас Repository. У цьому класі досить багато методів, але завдяки LINQ майже всі вони дуже невеликі. Завдяки мові запитів LINQ працювати із різними колекціями стає дуже просто. Наприклад, для того щоб отримати стартапи за певною категорією нам необхідний лише невеликий метод, що приймає у якості параметру категорію, а повертає параметризований список(див. рис. 3.19).

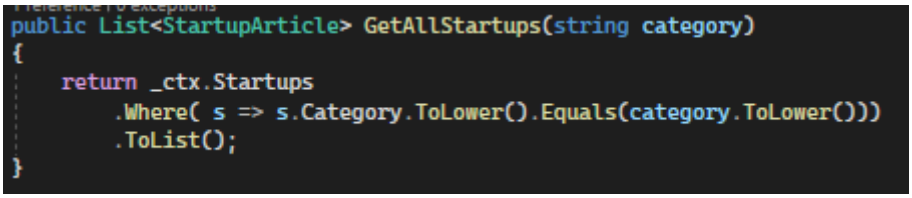


Рисунок 3.19 — Метод для пошуку стартапів за категорією

Для зручності читання метод займає три строки, але загалом його можна було написати й однією використовуючи при цьому лямбда вираз. Розглядаючи цей метод можна побачити використання об’єкту _ctx класу AppDbContext. Цей об’єкт строюється у конструкторі класу, та необхідний нам для взаємодії із базою даних через LINQ методи(див. рис. 3.20).

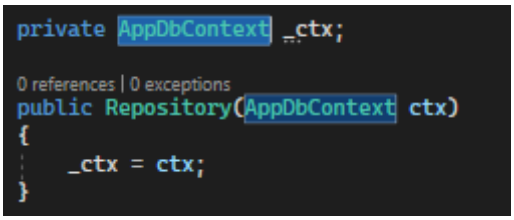
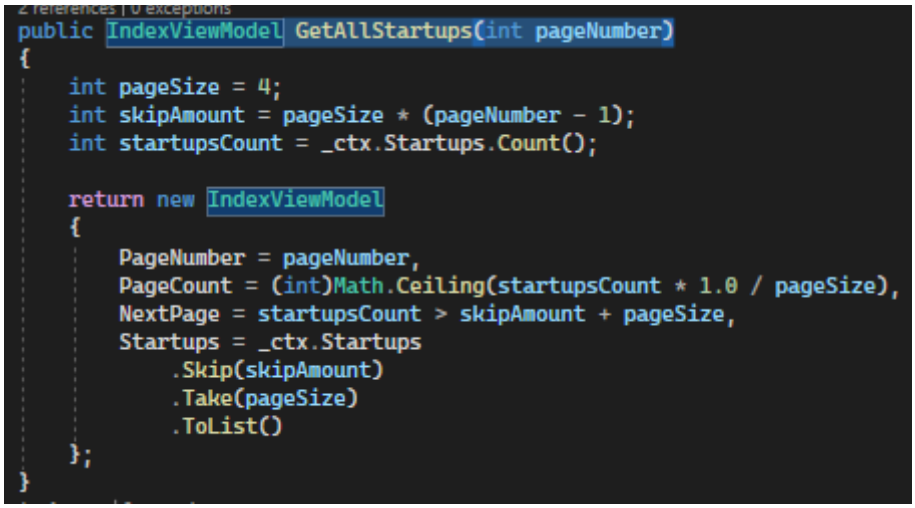


Рисунок 3.20 — Конструктор класу Repository

Далі метод пошуку стартапів за категорією GetAllStartups реалізує наступну логіку. Метод Where шукає у базі даних стартапи, що відповідають певній умові. Умова ця полягає у знаходженні усіх стартапів, категорія яких дорівнює категорії стартапу, яку передали у поточний метод. Також обидва значення категорії, які порівнюються приводяться до маленьких літер, з метою уникнення помилок за допомогою метода ToLower. Далі за допомогою метода ToList стартап-проекти, що відповідають необхідній категорії додаються до списку.

Більш складнішою є реалізація методу GetAllStartups(int pageNumber). Цей метод необхідний, щоб на одній сторінці відображалося не більше чотирьох конкретних стартапів, адже їх може бути дуже багато і скролити вниз усю сторінку буде не дуже зручно(див. рис. 3.21).



```
2 references | 0 exceptions
public IndexViewModel GetAllStartups(int pageNumber)
{
    int pageSize = 4;
    int skipAmount = pageSize * (pageNumber - 1);
    int startupsCount = _ctx.Startups.Count();

    return new IndexViewModel
    {
        PageNumber = pageNumber,
        PageCount = (int)Math.Ceiling(startupsCount * 1.0 / pageSize),
        NextPage = startupsCount > skipAmount + pageSize,
        Startups = _ctx.Startups
            .Skip(skipAmount)
            .Take(pageSize)
            .ToList()
    };
}
```

Рисунок 3.21 — Метод для пошуку стартапів на певній сторінці

У якості параметра цей метод приймає поточну сторінку. Далі розраховується скільки стартапів повинно бути на поточній сторінці і скільки треба пропустити. Потім за допомогою методу LINQ skip ми пропускаємо стартапи, що вже були відображені, та вибираємо тільки чотири нових стартап-проекта. Далі формуємо список із стартапів та повертаємо його.

Загалом клас Repository має наступні методи(див. рис. 3.22):

– AddStartup, що приймає у якості параметра об’єкт класу StartupArticle та додає його до бази даних за допомогою метода LINQ Add;

– GetAllStartups без параметра, це метод для отримання повного списку усіх стартап-проектів студентів, що зберігаються у базі даних;

– GetAllStartups, що приймає у якості параметру номер сторінки. Деталі реалізації цього методу вже були описані вище;

– GetStartup, що приймає у якості параметра ідентифікатор стартапу. Цей метод повертає не лише стартап за певним ідентифікатором, а й шукає коментарі, що були написані до цього стартап-проекту. Це можливо через те що було створено зв'язок один до багатьох між стратапами, коментарями та відповідями на коментарі. Відбувається пошук за допомогою таких методів LINQ як FirstOrDefault, Include та ThenInclude;

– RemoveStartup, що приймає у якості параметру ідентифікатор стартапу, потім знаходить цей стартап за допомогою методу GetStartup і на останок видаляє його методом Remove;

– UpdateStartup, що приймає у якості параметру об’єкт класу StartupArticle. Цей метод дозволяє змінювати значення стартап-проекту за допомогою методу Update;

– AddSubComment що приймає у якості параметру об’єкт класу. Цей метод додає до бази даних нову відповідь на коментар.

```
2 references | 0 exceptions
public void AddStartup(StartupArticle startup)
{
    _ctx.Startups.Add(startup);
}

2 references | 0 exceptions
public List<StartupArticle> GetAllStartups()
{
   return _ctx.Startups.ToList();
}
2 references | 0 exceptions
public IndexViewModel GetAllStartups(int pageNumber)
{
    int pageSize = 4;
    int skipAmount = pageSize * (pageNumber - 1);
    int startupsCount = _ctx.Startups.Count();

    return new IndexViewModel
    {
        PageNumber = pageNumber,
        PageCount = (int)Math.Ceiling(startupsCount * 1.0 / pageSize),
        NextPage = startupsCount > skipAmount + pageSize,
        Startups = _ctx.Startups
            .Skip(skipAmount)
            .Take(pageSize)
            .ToList()
    };
}
1 reference | 0 exceptions
public List<StartupArticle> GetAllStartups(string category)
{
    return _ctx.Startups
        .Where( s => s.Category.ToLower().Equals(category.ToLower()))
        .ToList();
}

5 references | 0 exceptions
public StartupArticle GetStartup(int id)
{
    return _ctx.Startups
        .Include(s => s.MainComments)
            .ThenInclude(mc => mc.SubComments)
        .FirstOrDefault(x => x.Id == id);
}

2 references | 0 exceptions
public void RemoveStartup(int id)
{
    StartupArticle startupArticle = GetStartup(id);
    _ctx.Startups.Remove(startupArticle);
}

3 references | 0 exceptions
public void UpdateStartup(StartupArticle startup)
{
    _ctx.Startups.Update(startup);
}
```

Рисунок 3.22 — Методи класу Repository

Також окрім цих методів є ще один. SaveChangesAsync – асинхронний метод необхідний для збереження внесених до бази даних змін. Нерідко програма виконує такі операції, які можуть зайняти тривалий час, наприклад, звернення до мережевих ресурсів, читання-запису файлів, звернення до бази даних і так далі. Такі операції можуть серйозно навантажити програму. Особливо це актуально у графічних (десктопних або мобільних) додатках, де тривалі операції можуть блокувати інтерфейс користувача та негативно вплинути на бажання користувача працювати з програмою, або у веб-додатках, які мають бути готові обслуговувати тисячі запитів на секунду. У синхронному додатку при виконанні тривалих операцій в основному потоці цей потік просто блокувався б на час виконання операції. І щоб тривалі операції не блокували загальну роботу програми, в C# можна задіяти асинхронність. Асинхронність дозволяє винести окремі завдання з основного потоку до спеціальних асинхронних методів і при цьому більш економно використовувати потоки. Асинхронні методи виконуються в окремих потоках. Однак при виконанні тривалої операції потік асинхронного методу повернеться в пул потоків і використовуватиметься для інших завдань. А коли тривала операція завершить своє виконання, для асинхронного методу знову виділяється потік з пулу потоків, асинхронний метод продовжує свою роботу. Отже, оскільки ми маємо веб застосунок до якого потенційно могли б звертатися багато людей нам необхідно зробити внесення змін до бази даних асинхронним методом. Саме тому метод SaveChangesAsync є асинхронним.

Окрім всіх цих даних, що зберігаються у базі даних ми також маємо там зберігати картинки. Для роботи з файлами буде використовуватись клас FileStream. Цей клас надає потік для файлу, підтримуючи як синхронні, так і асинхронні операції читання та запису. Інтерфейс з необхідними методами та клас, що його реалізує зберігаються у папці FileManager(див. рис. 3.23). Це також є реалізацією патерну Dependency Injection.

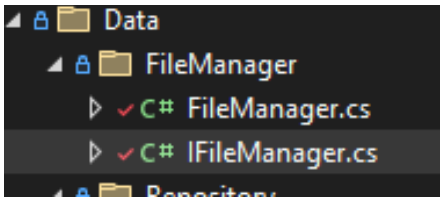


Рисунок 3.23 — FileManager

Інтерфейс IFileManager(див. рис. 3.24) має три методи:

– ImageStream, що приймає у якості параметру строку із назвою картинки, а повертає об’єкт класу FileStream. Цей метод необхідний для доступу до картинок, та їх подальшого відображення;

– SaveImage, цей метод повертає об’єкт класу Task параметризований типом string. Його суть полягає у збереженні малюнку до бази даних;

– RemoveImage – метод, що приймає об’єкт типу string, а повертає типу bool. Його мета полягає у видаленні малюнку стартапу із бази даних.

```
using Microsoft.AspNetCore.Http;
using System.IO;
using System.Threading.Tasks;

namespace Diplom.Data.FileManager
{
    6 references
    public interface IFileManager
    {
        2 references | 0 exceptions
        FileStream ImageStream(string image);
        2 references | 0 exceptions
        Task<string> SaveImage(IFormFile image);
        2 references | 0 exceptions
        bool RemoveImage(string image);
    }
}
```

Рисунок 3.24 — Інтерфейс IFileManager

Реалізує цей інтерфейс клас FileManager. Цей клас має конструктор(див. рис. 3.25), що приймає у якості параметру конфігурацію. Конфігурація програми в ASP.NET Core представляє об'єкт інтерфейсу IConfiguration. Конфігураційні установки зберігаються в файлі appsettings.

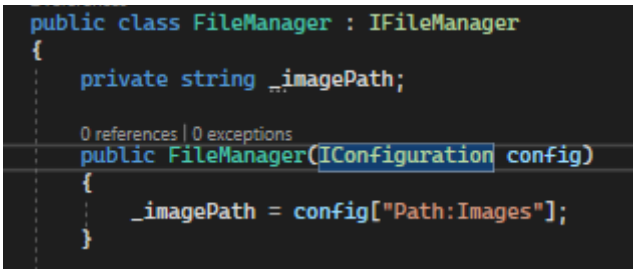


Рисунок 3.25 — Конструктор FileManager

Також окрім реалізації методів інтерфейсу IFileManager в даному класі наявен ще один метод із назвою ImageOptions. Цей метод не приймає параметрів, а повертає об’єкт класу ProcessImageSettings(див. рис. 3.26). Даний клас допомагає налаштовувати картинку та спосіб її зберігання.

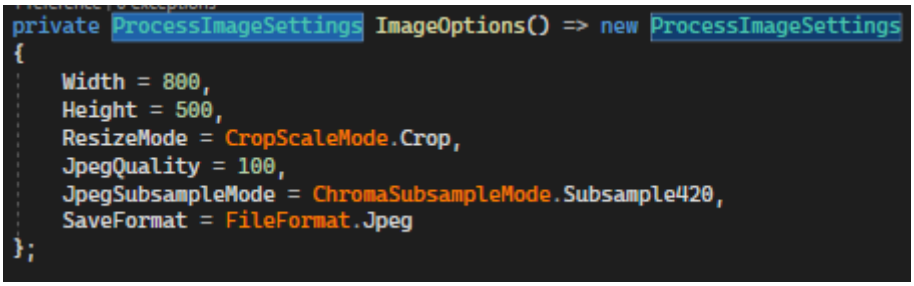


Рисунок 3.26 — Клас ProcessImageSettings

Для реалізації інших методів(див. рис. 3.27) ми використовуємо методи таких класів як File, Directory, MagicImageProcessor, FileStream. Під час збереження або видалення файлу можуть статися помилки, тому тіла методів краще заключити в блоки try, catch.

```
2 references | 0 exceptions
public FileStream ImageStream(string image)
{
    return new FileStream(Path.Combine(_imagePath, image), FileMode.Open, FileAccess.Read);
}

2 references | 0 exceptions
public bool RemoveImage(string image)
{
    try
    {
        var file = Path.Combine(_imagePath, image);
        if (File.Exists(file))
            File.Delete(file);
        return true;
    }
    catch(Exception e)
    {
        Console.WriteLine(e.Message);
        return false;
    }
}

2 references | 0 exceptions
public async Task<string> SaveImage(IFormFile image)
{
    try
    {
        var save_path = Path.Combine(_imagePath);
        if (!Directory.Exists(save_path))
        {
            Directory.CreateDirectory(save_path);
        }

        var mime = image.FileName.Substring(image.FileName.LastIndexOf('.'));
        var fileName = $"img_{DateTime.Now.ToString("dd-MM-yyyy-HH-mm-ss")}{mime}";

        using (var fileStream = new FileStream(Path.Combine(save_path, fileName), FileMode.Create))
        {
            MagicImageProcessor.ProcessImage(image.OpenReadStream(), fileStream, ImageOptions());
        }

        return fileName;
    }
    catch(Exception e)
    {
        Console.WriteLine(e.Message);
        return "Error";
    }
}
```

Рисунок 3.27 — Методи для роботи з картинками

3.3 Контролери

Центральною ланкою в архітектурі ASP.NET Core MVC є контролер. При отриманні запиту система маршрутизації вибирає для обробки запиту необхідний контролер і передає дані запиту. Контролер обробляє ці дані та посилає назад результат обробки. В ASP.NET Core MVC контролер представляє звичайний клас мовою C#, який успадковується від абстрактного базового класу Microsoft.AspNetCore.Mvc.Controller. За замовчуванням проект ASP.NET Core MVC містить щонайменше один контролер – HomeController. При використанні контролерів є деякі умовності. По-перше, у проекті контролери повинні розміщуватися в каталозі Controllers. І по-друге, назви контролерів зазвичай закінчуються на суфікс "Controller", решта ж до цього суфікса вважається ім'ям контролера, наприклад, HomeController. Але у принципі ці умовності необов'язкові.

Контролер, як і будь-який клас мовою C#, може мати поля, властивості, методи. За замовчанням HomeController має чотири методи, які можна назвати діями. Дії контролера - це публічні методи, які можуть зіставлятися із запитами. Наприклад, стандартний контролер містить метод Index – він має модифікатор public і тому може використовуватись для обробки запиту.

Щоб звернутися до контролера з веб-браузера, нам потрібно в адресному рядку набрати адресу_сайту/Ім'я_контролера/Дія_контролера . Так, за запитом адреси_сайту/Home/Index система маршрутизації за замовчуванням викличе метод Index контролера HomeController для обробки вхідного запиту. Методи у межах однієї дії можуть обслуговувати різні запити. Для вказівки типу запиту HTTP нам потрібно застосувати до методу один з атрибутів: [HttpGet] , [HttpPost] , [HttpPut] , [HttpDelete] та [HttpHead] . Якщо атрибут явно не вказано, то метод може обробляти всі типи запитів: GET, POST, PUT, DELETE.

У даному проекті є три основні контролери, які зберігаються у відповідній папці(див. рис. 3.28).

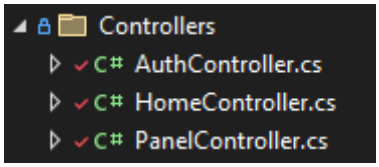


Рисунок 3.28 — Контролери

Спочатку розглянемо HomeController. Цей контролер у конструкторі(див. рис. 3.29) присвоює значення двом полям, які знадобляться нам для роботи із базою даних.

```
public class HomeController : Controller
{
    private IRepository _repos;
    private IFileManager _fileManager;

    0 references | 0 exceptions
    public HomeController(
        IRepository repos,
        IFileManager fileManager
        )
    {
        _repos = repos;
        _fileManager = fileManager;
    }
```

Рисунок 3.29 — Конструктор контролеру HomeController

Цей контролер має чотири основні дії(див. рис. 3.30):

– Index – ця дія необхідна для відображення поточної сторінки із чотирма стартап-проектами;

– Startup – це дія для відображення конкретного стартапу;

– Image – дія необхідна для доступу до певного малюнку;

– Comment – POST метод для додавання нових коментарів, чи відповідей до них.

```
0 references | 0 requests | 0 exceptions
public IActionResult Index(int pageNumber, string category)
{
    if(pageNumber < 1)
    {
        return RedirectToAction("Index", new {pageNumber = 1, category});
    }

    var vm = _repos.GetAllStartups(pageNumber);

    return View(vm);
}

0 references | 0 requests | 0 exceptions
public IActionResult Startup(int id) =>
    View(_repos.GetStartup(id));

[HttpGet("/Image/{image}")]
0 references | 0 requests | 0 exceptions
public IActionResult Image(string image) =>
    new FileStreamResult(_fileManager.ImageStream(image), $"image/{image.Substring(image.LastIndexOf('.') + 1)}");

[HttpPost]
0 references | 0 requests | 0 exceptions
public async Task<IActionResult> Comment(CommentViewModel vm)
{
    if(!ModelState.IsValid)
        return RedirectToAction("Startup", new { id = vm.StartupId });
    var startup = _repos.GetStartup(vm.StartupId);
    if(vm.MainCommentId == 0)
    {
        startup.MainComments = startup.MainComments ?? new List<MainComment>();
        startup.MainComments.Add(new MainComment
        {
            Message = vm.Message,
            Created = DateTime.Now
        });
        _repos.UpdateStartup(startup);
    }
    else
    {
        var comment = new SubComment
        {
            MainCommentId = vm.MainCommentId,
            Message = vm.Message,
            Created = DateTime.Now
        };
        _repos.AddSubComment(comment);
    }

    await _repos.SaveChangesAsync();
    return RedirectToAction("Startup", new { id = vm.StartupId });
}
```

Рисунок 3.30 — HomeController

Також ми маємо контролер AuthController. Цей контролер необхідний нам для авторизації, виходу із акаунту та реєстрації.

Login метод не тільки допомагає користувачу зайти до його акаунту, ай перевіряє чи є користувач адміністратором, чи студентом(див. рис. 3.31).

Рисунок 3.31 — POST метод Login

Останній контролер має всі необхідні методи, що знадобляться нам для панелі адміністрування, та панелі редагування та створення стартапів для користувачів.

3.4 Frontend

Для реалізації фронтендової частини(див. рис. 3.32) використовувалися сторінки razor та також моделі, що приймали з контролерів необхідні дані.

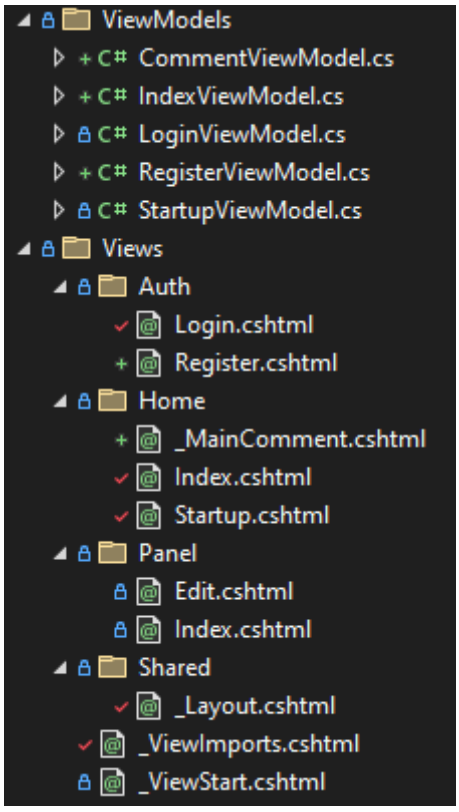


Рисунок 3.32 — Frontend проекту

Усі сторінки мають спільні компоненти, які були винесені до папки Shared. В ній знаходиться razor page _Layout(див. рис. 3.33).

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewBag.Title</title>
    <meta name="description" content="@ViewBag.Description"/>
    <meta name="keywords" content="@ViewBag.Keywords"/>
    <link href="~/css/site.css" rel="stylesheet"/>
</head>
<body>
    <div>
        <nav>
            <a asp-controller="Home" asp-action="Index">Головна</a>
            <a asp-controller="Home" asp-action="Index" asp-route-category="Збір_коштів">Збір коштів</a>
            <a asp-controller="Home" asp-action="Index" asp-route-category="Підбір_людей">Підбір людей до стартапу</a>
        @if (User.Identity.IsAuthenticated)
        {
              @if (User.IsInRole("Admin"))
              {
                  <a asp-controller="Home" asp-action="Index" asp-route-category="Panel"> Панель адміністратора </a>
              }
            <a asp-controller="Auth" asp-action="Logout">Вийти</a>
        }
        else
        {
        <a asp-controller="Auth" asp-action="Login">Авторизуватися</a>
        }
        </nav>
    </div>
    <div>
        @RenderBody()
    </div>
    @RenderSection("scripts", false)
</body>
</html>
```

Рисунок 3.33 — _Layout

Всі інші razor pages відображають інші сторінки проекту, такі як авторизація, реєстрація, сторінка стартап-проекту, панель адміністратора та інші(див. рис. 3.34).

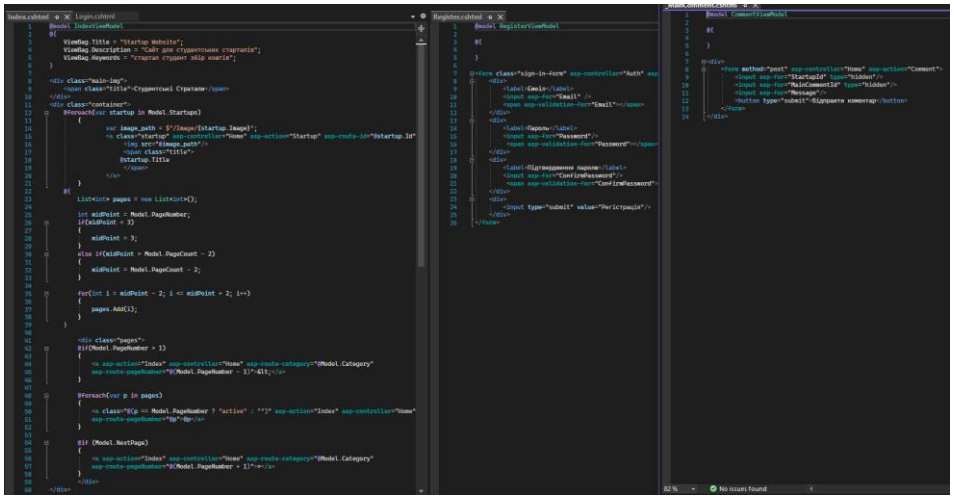


Рисунок 3.34 — Razor pages

У проекті також використовується CSS. Всі класи CSS зберігаються у файлі site.css(див. рис. 3.35).

Рисунок 3.35 — CSS класи

3.5 Огляд проекту

При заході на сайт ми можемо побачити стартапи, назви сайту, та чотири посилання, на які ми можемо перейти(див. рис. 3.36).

Рисунок 3.36 — Головна сторінка

Користувач, що не авторизувався на сайті може переглянути стартап, але не може створювати свої стартапи, коментувати їх, та редагувати. (див. рис. 3.37)

Рисунок 3.37 — Сторінка стартапу, для неавторизованого користувача

Також суть цього веб застосунку полягає в тому, що дуже часто студентські проекти мають не стільки велику потребу в грошах, скільки в людських ресурсах. Тому всі стартапи поділяються за категоріями на ті, що збирають гроші та на ті, що шукають людей. Ці категорії виставляються під час створення або редагування стартап-проекту. Відповідно побачити лише ті або інші стартапи можна за допомогою відповідних кнопок у header-і сайту. Якщо користувач ще не має акаунта, він може його створити натиснувши на посилання авторизації, а потім на кнопку реєстрації(див. рис. 3.38).

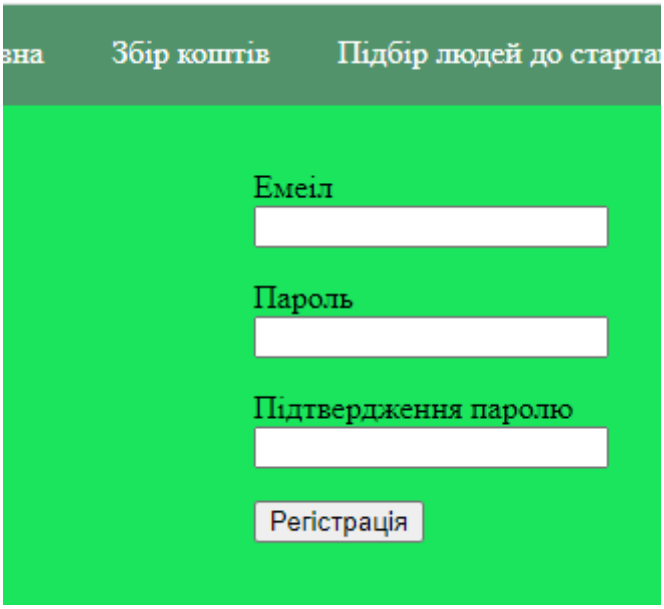


Рисунок 3.38 — Реєстрація нового користувача

Якщо ми спробуємо ввести дані не заповнивши поля, то виникне помилка(див. рис. 3.39).

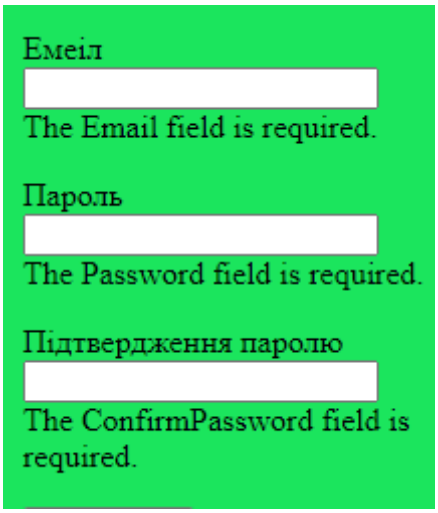


Рисунок 3.39 — Помилка реєстрації

Якщо ми спробуємо зареєструватися, але неправильно введемо електрону пошту, тобто вона не буде відповідати формату електронної пошти, то ми також отримаємо помилку(див. рис. 3.40).

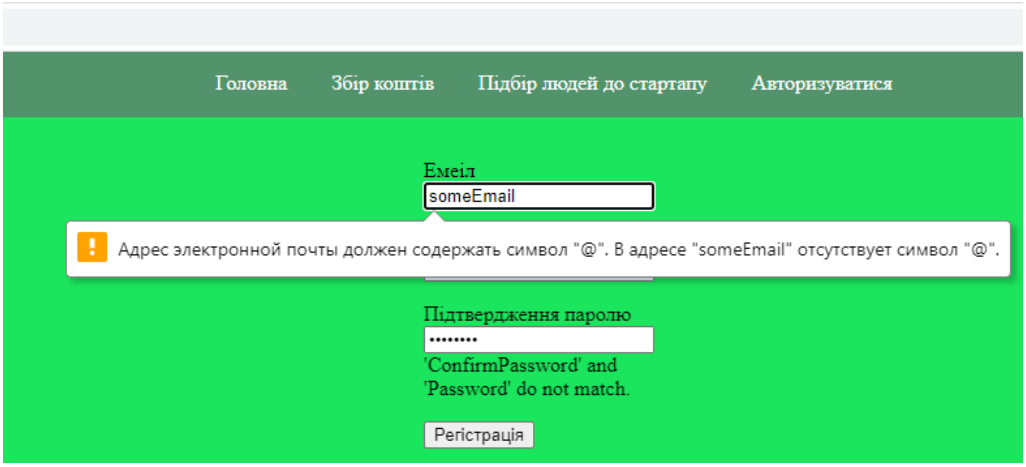


Рисунок 3.40 — Помилка введення електронної пошти

Якщо поля паролю, та підтвердження паролю не будуть збігатися, то ми також отримаємо помилку (див. рис. 3.41).

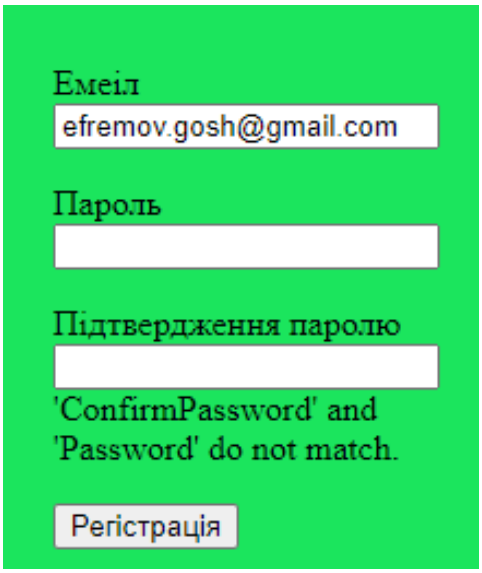


Рисунок 3.41 — Помилка збіжності полів

При успішній реєстрації ми опинимося на головній сторінці, та матимемо можливість створювати нові стартапи, додавати коментарі та вийти із акаунта(див. рис. 3.42).

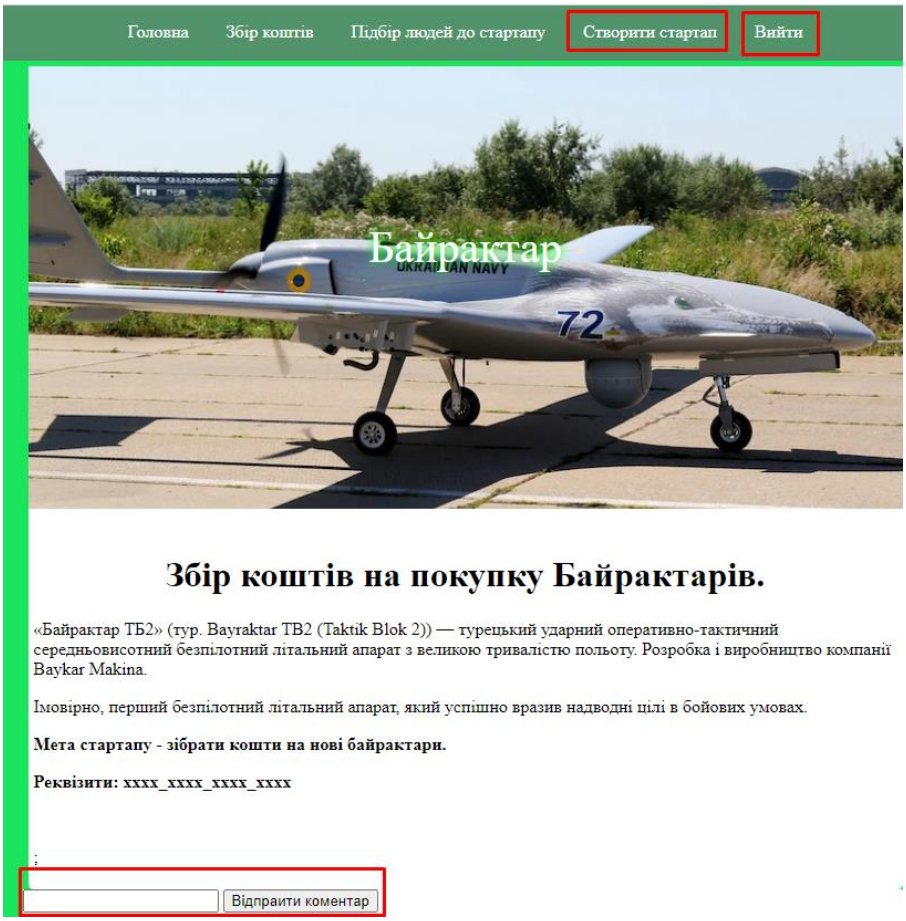


Рисунок 3.42 — Сторінка стартапу після реєстрації

Тепер можна додати коментар(див. рис. 3.43)

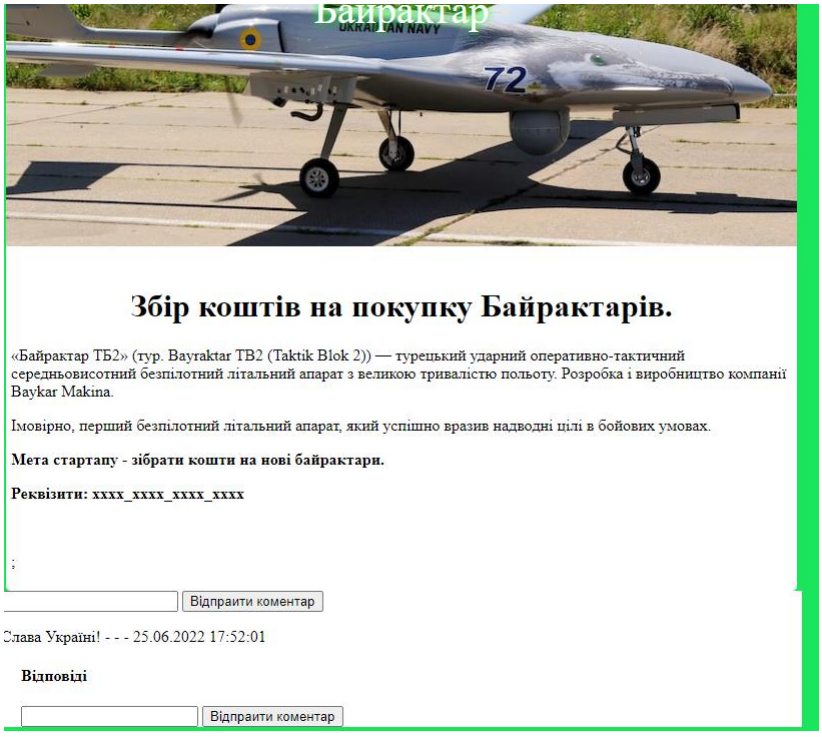


Рисунок 3.43 — Додавання коментарю

Цей коментар заноситься до бази даних та тепер зберігається там(див. рис. 3.44).

2	6	Палить кацапів	2022-06-19 23:55:36.5982340	NULL
3	7	Слава Україні!	2022-06-25 17:52:01.4422751	2008

Рисунок 3.44 — Додавання коментарю до бази даних

Можна додати відповіді на коментар, та створити ще декілька коментарів(див. рис. 3.45)

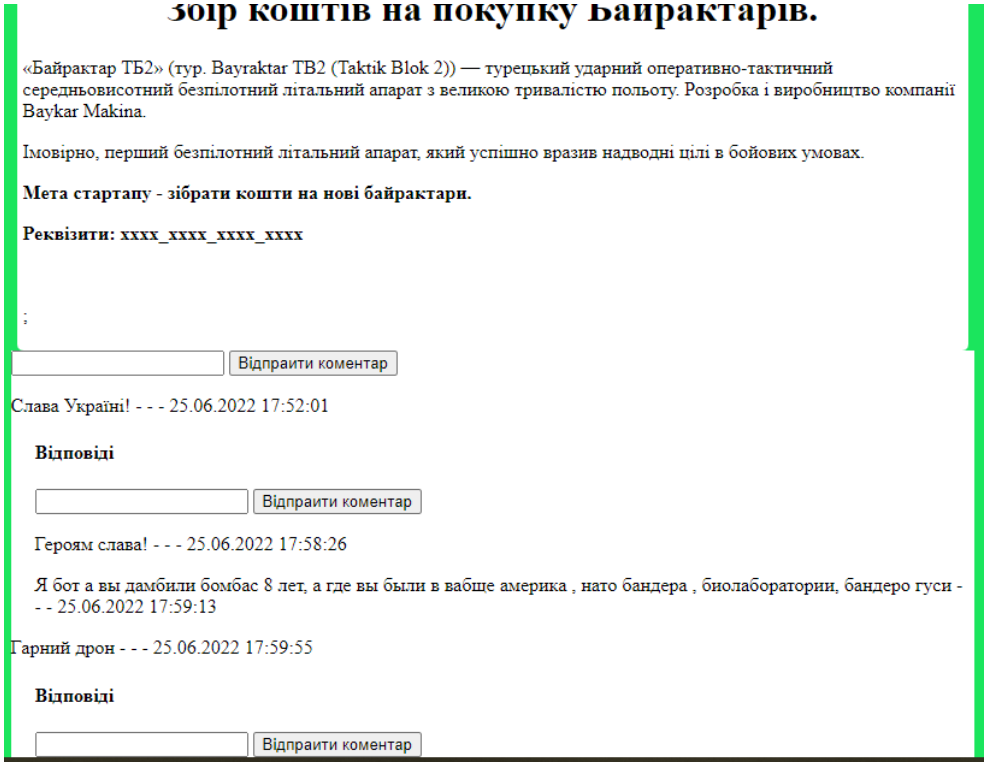


Рисунок 3.45— Додавання коментарів та відповідей на них

Також тепер ми можемо створити власний стартап-проект, для цього треба натиснути на посилання створити стартап. Таким чином ми перейдемо до сторінки створення стартапу. Тут можемо ввести назву стартапу, його опис, теги та мету стартапу(див. рис. 3.46).

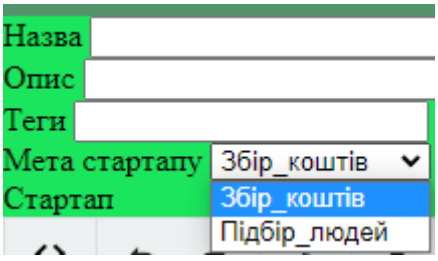


Рисунок 3.46 — Короткий опис стартапу

Також ми можемо додати головну картинку стартапу(див. рис. 3.47).

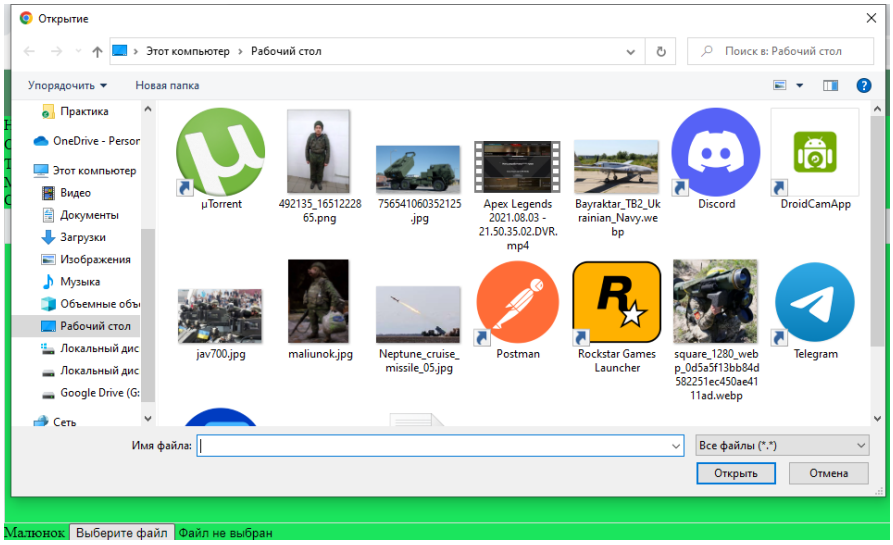


Рисунок 3.47 — Додавання малюнку

Стосовно основної частини стартап-проекту все більш складніше. Тут не достатньо лише текстового поля, чи простого завантаження картинки. Користувач може захотіти виділити текст курсивом, додати більше малюнків, створити список, додати посилання на щось, може йому знадобиться додати степені чи коефіцієнти. Все це не можливо, або дуже важко чи незручно зробити у звичайному текстовому полі. Отже нам необхідний певний текстовий редактор. На щастя в інтернеті є дуже багато безкоштовних текстових редакторів із відкритим кодом. Один із них – Trumbowyg(див. рис. 3.48). Інструкція як додати його до свого проекту. Він дуже зручний і має багато функцій, а головне він безкоштовний. Таким чином нам не потрібно буде витрачати час на його створення [8].

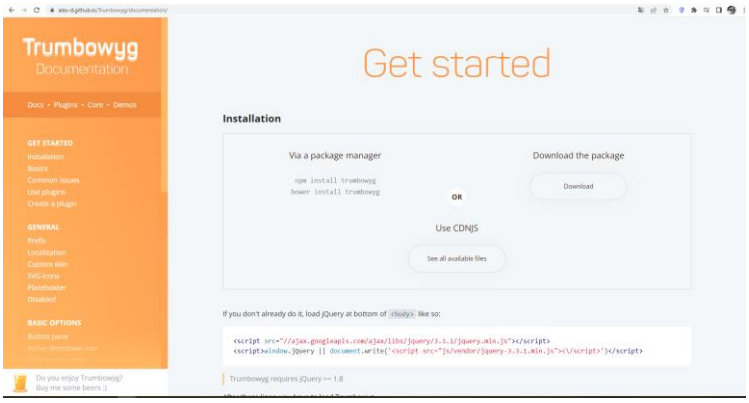


Рисунок 3.48 — Сайт Trumbowyg

Отже тепер під час створення стартапу ми можемо користуватись дуже зручним текстовим редактором(див. рис. 3.49).

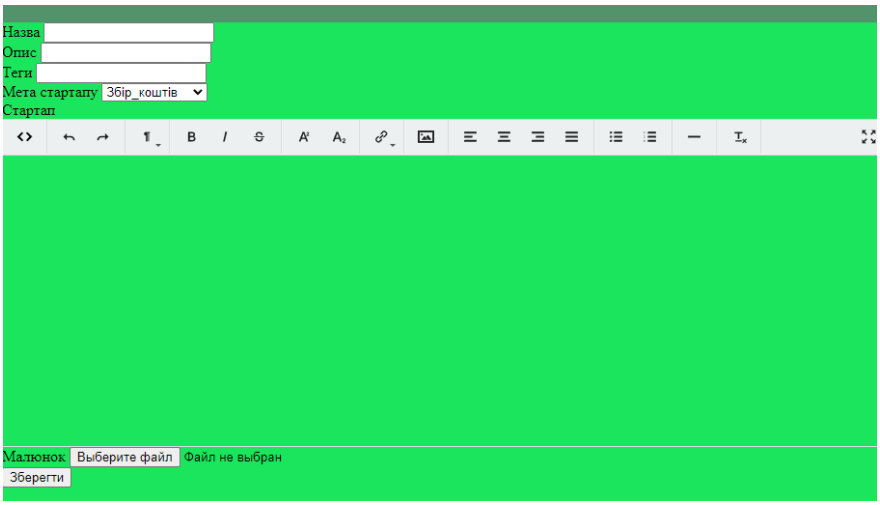


Рисунок 3.49 — Повна сторінка створення стартапів

Програмний код цього текстового редактору знаходиться у папці wwwroot(див. рис. 3.50), як і всі динамічні та статичні малюнки, а також css стилі.

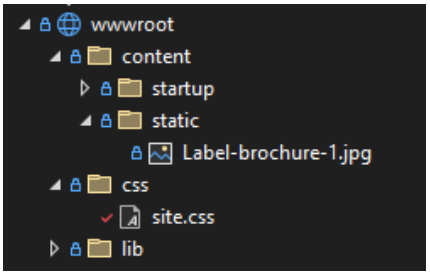


Рисунок 3.50 — Папка wwwroot

Спробуємо створити новий стартап використовуючи цей редактор та інші поля(див. рис. 3.51 – 3.53).

Рисунок 3.51 — Створення нового стартапу

Рисунок 3.52 — Сторінка після додавання нового стартапу

Рисунок 3.53 — Новий стартап

Якщо ж ми авторизуємося як адміністратор(див. рис. 3.54), то зможемо керувати всіма стартапами, редагувати їх та видаляти.

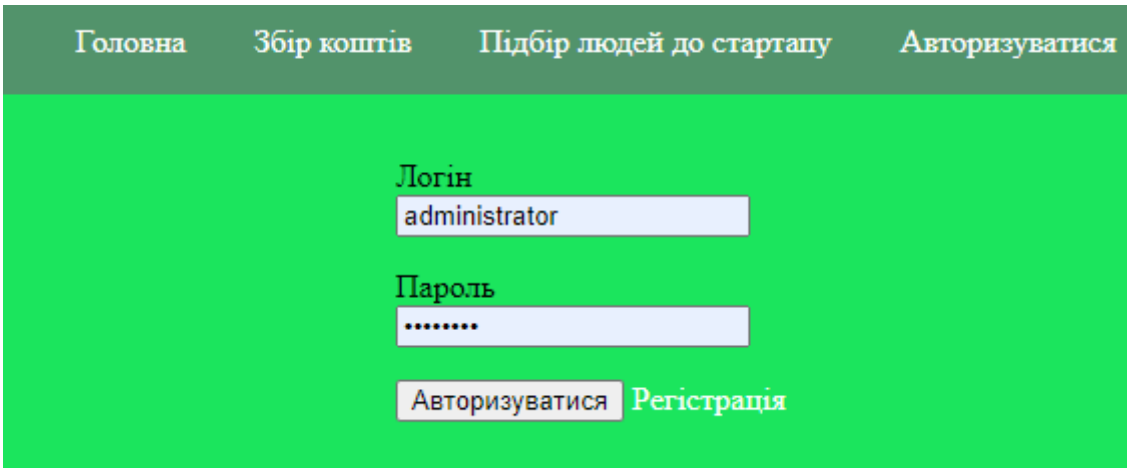


Рисунок 3.54 — Авторизація у якості адміністратора

Взаємодіяти із стартапами ми зможемо через амін панель(див. рис. 3.55).

Рисунок 3.55 — Панель адміністратора

Висновки до розділу 3

Під час реалізації проекту було розроблено базу даних, яка містить стартапи, коментарі та підкоментарі для стартапів.

Було розроблено веб застосунок, в якому були реалізовані наступні функці:

– авторизація користувачів, що мають різні ролі;
– реєстрація, створення, редагування та видалення стартапу;
– додавання картинок та їх збереження;
– додавання різного виду коментарів;
– пейджування та деякі інші функції.

У розділі дипломної роботи була розписна робота веб застосунку, а саме:

– показана архітектура та її реалізаці;
– розписані класи, інтерфейси, методи та інше;
– на практиці продемонстрована робота веб застосунку.
– були наведені приклади праці із деякими класами, технологіями та кодом у відкритому доступі;
– продемонстрована взаємодія із базою даних.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

СИСТЕМА КРАУДФАНДИНГА ДЛЯ СТУДЕНТСЬКИХ СТАРТАП-ПРОЕКТІВ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810210

Виконав студент 4-го курсу, групи 402
_____ *Г. І. Єфремов*

«___» _____ 2022 р.

Консультант к.т.н., доцент
_____*А. О. Алексєєва*

«___» _____ 2022 р.

Миколаїв – 2022

ЗМІСТ

ВСТУП 58
4. ОХОРОНА ПРАЦІ 59
4.1 Нормативна база в галузі охорони праці при роботі з екранними пристроями 59
4.2. Загальні вимоги до роботодавців 59
4.3 Вимоги до приміщення з використанням екранних пристроїв 60
4.4 Вимоги до роботи з екранними пристроями 63
4.5. Вимоги щодо режиму відпочинку та праці на підприємствах з екранними пристроями 64
Висновки до розділу 4 65

ВСТУП

В наш час все більше і більше видів праці переміщуються до комфортних офісів, або до дому наприклад через пандемію. Але навіть так будь який вид праці має певні правила поведінки, задля збереження здоров’я. Звичайно такі види праці як рятувальник, чи солдат є набагато більш небезпечними та мають значно більше ризиків, особливо у наш воєнний час. Але те що хтось працює у навіть набагато більш безпечних умовах не звільняє його від дотримання правил охорони праці.

Тривала робота у сидячому положенні тіла за екранними пристроями може у тривалій перспективі мати негативні наслідки, якщо не дотримуватись правил охорони праці.

Перш за все, слід зазначити, що незалежно від типу та характеру роботи, яку виконує працівник, роботодавець завжди повинен дотримуватися нинішніх правил захисту праці та заходів безпеки. Основою для їх дотримання є правильна організація роботодавця робочого середовища та забезпечення відповідних умов для її впровадження. Робота за комп’ютером також регулюється правилами безпеки.

Правила захисту праці та запобіжних заходів досить чітко визначають, які вимоги щодо захисту праці та безпеки повинні дотримуватися роботодавців під час найму працівників. Однак часто підприємці застосовують ці норми лише до виробничих працівників (у галузі робочого одягу та взуття), забуваючи, що офісні працівники також мають власні права. Комп'ютерні працівники особливо вразливі в цьому випадку. Робота за комп’ютером і довгі години до того, як монітор може негативно вплинути на здоров'я працівника, особливо погіршити його бачення, тому в цій ситуації так важливо відповідати нинішнім правилам безпеки та захисту праці.

4. ОХОРОНА ПРАЦІ

4.1 Нормативна база в галузі охорони праці при роботі з екранними пристроями

Екранний пристрій – електронний засіб для відтворення будь-якої графічної або буквено-цифрової інформації (на основі електронно-променевих трубок, рідких кристалів, плазми, проекцій, органічних світлодіодних дисплеїв та інших нових розробок у сфері інформаційних технологій) [12].

Мінімальні вимоги до безпеки та охорони здоров’я при використанні екранних пристроїв усіх типів і моделей установлюють вимоги безпеки та охорони здоров’я працівників при використанні екранних пристроїв, затверджені наказом Мінсоцполітики від 14.02.2018 р. № 207. всім суб'єктам господарювання, незалежно від форм власності, організаційно-правової форми та діяльності, та встановлюють мінімальні вимоги безпеки та охорони праці до робіт, пов'язаних із використанням екранних пристроїв, незалежно від їх типу та моделі.

4.2. Загальні вимоги до роботодавців

Нижче наведено частину вимог з документу про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями:

Зокрема, роботодавець зобов'язаний:

– організовувати роботу таким чином, щоб забезпечити безпечні та гігієнічні умови праці,

– забезпечити дотримання положень та правил захисту праці та безпеки на робочому місці, видавати інструкції щодо усунення недоліків у цьому плані та відстеження виконання цих інструкцій,

– реагувати на потреби забезпечення охорони здоров'я та безпеки праці та адаптації заходів, що вживаються для підвищення існуючого

рівня охорони здоров'я та життя працівників, враховуючи зміни умов роботи,

– забезпечити розробку послідовної політики щодо запобігання нещасних випадків у виробничих та професійних захворюваннях, з урахуванням технічних питань, організації праці, умов праці, соціальних відносин та впливу факторів робочого середовища,

– враховувати захист здоров'я підлітків, вагітних або годувальних працівників та інвалідів у рамках профілактичних заходів,

– забезпечити виконання замовлень, заяви, рішень та наказів керівних органів,

– забезпечити виконання рекомендацій інспектора соціальної праці.

4.3 Вимоги до приміщення з використанням екранних пристроїв

Робочі місця працівників, які використовують екранні пристрої, повинні відповідати ергономічним, антропологічним, психофізіологічним вимогам, характеру виконуваної роботи. Вони повинні бути сконструйовані та розраховані таким чином, щоб працівники мали можливість змінювати своє робоче положення та рухи. Освітлення робочого місця повинно створювати відповідний контраст між екраном і навколишнім середовищем, враховуючи вид роботи та відповідаючи ДСанПіН 3.3.2.007-98. Мікроклімат промислових об’єктів та робочих місць працівників із екранним обладнанням необхідно підтримувати на постійному рівні відповідно до ДСН 3.3.6.042-99 «Гігієнічні норми мікроклімату на промислових об’єктах», затверджених МОЗ та Головним державним санітарним лікарем України. від 01.12.1999 № 42 . Робочі місця з екранними пристроями не повинні обмежувати пересування працівників [13].

Робочі місця з використанням екранних пристроїв повинні відповідати ергономічним, антропологічним, психофізіологічним вимогам і характеру виконуваної роботи. Вони повинні бути розроблені та розраховані таким

чином, щоб працівники мали можливість змінювати свої робочі положення та рухи. Освітлення робочого місця повинно створювати відповідний контраст між екраном і навколишнім середовищем, враховуючи вид роботи та відповідаючи ДСанПіН 3.3.2.007-98. Відповідно до ДСН 3.3.6.042-99 «Стандарт мікроклімату гігієни промислових об'єктів», затвердженого МОЗ та державного головного санітарного лікаря, мікроклімат промислових об'єктів і робочих місць працівників має підтримуватися на постійному рівні України. Від 01.12.1999 № 42. Робочі місця з екранними пристроями не повинні обмежувати пересування працівників.

Крім цього для зменшення впливу негативних факторів на працівника, виробниче приміщення повинно відповідати нормам щодо параметрів мікроклімату, освітлення, шуму та вібрації, рівні електромагнітного та іонізуючого випромінювання.

По-перше, на усіх робочих місцях із електронно-обчислювальними машинами обов'язково має бути забезпечено оптимальні значення параметрів температури відносної вологості й рухливості повітря (ГОСТ 12.1.005-88, СН 4088-86) (табл. 4.1).

Таблиця 4.1 – Нормовані величини температури, відносної вологості та швидкості руху в робочій зоні виробничих приміщень з ВДТ

Пора року	Категорія робіт	Оптимальна температура повітря, °С, не більше	Оптимальна відносна вологість повітря, %	Оптимальна швидкість руху повітря, м/с
Холодна	Легка – Іа	22-24	40-60	0,1
	Легка – Іб	21-23	40-60	0,1
Тепла	Легка – Іа	23-25	40-60	0,1
	Легка – Іб	22-24	40-60	0,2

До категорії робіт Іа належать, що виконується робота сидячі і не потребує фізичного напруження людини, до категорії робіт Іб належать, що робота виконується сидячі, стоячи або пов'язані з ходінням та потребує деякого фізичного напруження.

По-друге, показники рівню позитивних та негативних іонів в повітрі приміщень з ВДТ мають відповідати санітарно – гігієнічним нормам №2152-80 (табл. 4.2).

Таблиця 4.2 – Рівні іонізації повітря приміщень при роботі на ВДТ

Рівні	Кількість іонів в 1 $см^3$ повітря	
	n+	n-
Мінімально необхідний	400	600
Оптимальний	1500-3000	3000-5000
Максимально допустимий	50000	50000

Вимірювання кількості іонів та його полярності порядку поточного нагляду проводиться 1 разів у квартал. Вимірювання проводяться також у випадках:

– встановлення нових або відремонтованих іонізаторів,

– організації нових робочих місць,

– впровадження нових технологічних процесів, що потенційно можуть. змінити іонний режим у зоні дихання персоналу [14].

Все необхідне обладнання, необхідне для роботи на комп'ютері, повинно бути в межах досяжності працівника, не змушуючи його змінити місце роботи.

4.4 Вимоги до роботи з екранними пристроями

На сьогодні існує ряд вимог, які повинні використовувати працівники на підприємствах, основним місцем роботи яких є місце за персональним комп'ютером. Ці вимоги можна поділити за наступними критеріями:

Ергономіка положення:

– робоче місце повинно бути достатньо великим, щоб вільно використовувати всі елементи з ручним управлінням у межах світла;

– відстань між моніторами повинна бути 60 см, а між працівником та задньою частиною іншого сусіднього монітора - 80 см;

– відстань між працівником та монітором повинна бути від 40 см до 75 см;

– для кожного працівника має бути 2 м2 вільної площі та 13 м2 обсягу приміщення, без обмежень на меблі, обладнання тощо;

– екран монітора повинен відповідати певним вимогам: написи на екрані повинні бути чіткими та розбірливими, зображення повинно бути стабільним, екран повинен бути охоплений просвітлюючим шаром або оснащеним відповідним фільтром тощо;

– клавіатура комп'ютера повинна бути не менше 10 см від краю таблиці і повинна бути окремим елементом основного обладнання робочого місця, його поверхня повинна бути матовою, а символи контрастні та вибіркові.

Крісло як елемент робочого місця біля екрану монітора:

– має бути стійким з як мінімум 5 опорною базою, на колесах з можливістю обертання на 360° навколо вертикальної осі;

– розміри спинки та сидіння повинні забезпечувати зручне положення тіла та свободу рухів;

– з можливістю регулювання висоти сидіння в діапазоні 40-50 см, рахуючи від підлоги;

– з можливістю регулювання висоти спинки та регулювання нахилу спинки в діапазоні 5° вперед і 30° назад;

– сидіння та спинка мають бути сконструйовані таким чином, щоб відповідати природному вигину хребта та стегна;

– повинні бути обладнані підлокітниками;

– за бажанням працівника, а також у випадках, коли висота крісла унеможливлює розміщення працівника ступнями на підлозі, робоче місце має бути обладнане підставкою для ніг.

Розміри та налаштування стола:

– висота столу повинна забезпечувати вільне положення працівника, зберігаючи прямий кут між плечем і передпліччям під час роботи за комп'ютером;

– висота столу, на якому розташований екран монітора, повинна забезпечувати відповідний кут огляду екрана монітора в діапазоні 20-50° вниз (враховуючи від горизонтальної лінії на рівні очей співробітника до лінії, проведеної від очима до центру екрана), верхнім краєм екран монітора не повинен бути над очима працівника;

– коліна працівника, що сидить за столом, не повинні торкатися стільниці, простір під столом має бути вільним;

– поверхня стільниці повинна бути матовою, бажано світлого кольору.

4.5. Вимоги щодо режиму відпочинку та праці на підприємствах з екранними пристроями

Обов'язок роботодавця – організувати роботу так, щоб кожен працівник після кожної години безперервної роботи за комп'ютером мав можливість змінити вид роботи на такий, який не обтяжуватиме зір або буде виконуватися в зміненому положенні тіла. Якщо роботодавець не може забезпечити зміну

роботи, він повинен забезпечити перерву не менше 5 хвилин на кожну годину роботи перед екраном. Така перерва зараховується до робочого часу працівника і не зменшує оплату праці.

При виконанні протягом дня робіт, що належать до різних видів трудової діяльності, за основну роботу з ВДТ ЕОМ і ПЕОМ слід вважати таку, що займає не менше 50% часу впродовж робочої зміни мають передбачатися:

– перерви для відпочинку і вживання їжі (обідні перерви);

– перерви для відпочинку і особистих потреб (згідно з трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності;

Встановлюються такі внутрішньо змінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні в залежності від характеру праці;

– для розробників програм із застосуванням ЕОМ, слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ВДТ;

– для операторів із застосування ЕОМ, слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп'ютерного набора слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожною години роботи за ВДТ.

Висновки до розділу 4

У ході виконання теми охорони праці проведено розслідування щодо організації праці працівників на підприємствах і в установах, а також щодо правил роботи з електронно-обчислювальними приладами. ІТ-професія з кожним роком стає все більш важливою: програмісти, дизайнери,

тестувальники, розробники баз даних, менеджери проектів — лише деякі з тих професіоналів, які потребують бізнесу як ніколи. Зростання попиту на таких спеціалістів було очевидним і до глобальної пандемії коронавірусу, але реальне життя фактично призвело до дефіциту людей, які можуть виконувати таку роботу на високому рівні. Люди такого види діяльності проводять багато часу з екранними пристроями у сидячому положенні і тому охорона праці також важлива в IT.

За допомогою проведення цієї роботи були освоєні права та обов’язки робітників та роботодавців. Також було детально розглянуто умови середовища, у яких працівники мають виконувати свої професійні обов’язки.

ВИСНОВКИ

Під час виконання кваліфікаційної бакалаврської роботи було досліджено сферу краудфандингу та його типів, проаналізовано та обрано інструменти для розробки проекту, розроблено веб застосунок – систему краудфандинга для студентських стартап-проектів.

Після дослідження сфери краудфандингу було визначено його переваги та недоліки. Перевагами є забезпечення доступу до капіталу, хеджування ризиків, мозковий штурм, знайомство із потенційними клієнтами, безкоштовний піар, можливість передпродажу, відсутність штрафів. Недоліком є безпека бізнес-ідеї, а саме необхідність поділитися нею з інтернет-спільнотою. Також у ході дослідження сфери було визначено його види, а саме краудфандинг акцій (краудінвестінг), борговий краудфандинг та краудфандинговий кікстартер. Під час дослідження сфери було проаналізовано приклади сайтів краудфандингових проектів. У ході аналізу було визначено основні функції цих сайтів - реєстрація та авторизація на сайті, головна сторінка з активними проектами, перегляд опису конкретного стартапу, можливість редагування стартапів, можливість залишати коментарі до стартапів та відповідати на коментарі інших користувачів, функція збору коштів для стартапу.

Після аналізу інструментів та технологій для розробки веб застосунку було обрано мову програмування C#, яка є універсальною, високорівневою, об'єктно-орієнтованою мовою програмування. Як середовище розробки проекту було обрано Visual Studio, з її найголовнішою перевагою – безкоштовність**.** Для запису, читання та редагування даних у даному проекті було використано мову SQL, і, в якості системи управління базами даних, було обрано Microsoft SQL Server та SQL Server Management Studio у якості інтегрованого середовища для управління всіма компонентами , що входять до складу Microsoft SQL Server. Були обрані такі технології, як Razor pages –

фреймворк, який забезпечує двостороннє прив’язування даних і спрощує розробку з поодинокими проблемами, Microsoft identity – набір бібліотек ASP.NET Core, які спрощують додавання підтримки автентифікації та авторизації до веб-програм і веб-API, SmtpClient Class – клас, який дозволяє програмам надсилати електронну пошту за допомогою протоколу Simple Mail Transfer Protocol (SMTP), Dependency Injection – шаблон проектування, в якому об'єкт не ініціює свої залежності сам, а приймає їх зовні, основною перевагою якого є полегшене написання модульних застосунків, LINQ, який представляє просту та зручну мову запитів до джерела даних, Entity Framework – спеціальна об'єктно-орієнтована технологія на базі фреймворку .NET для роботи з даними.

Під час реалізації проекту було розроблено базу даних, яка містить стартапи, коментарі та підкоментарі для стартапів.

Було розроблено веб застосунок, в якому були реалізовані наступні функці:

- авторизація користувачів, що мають різні ролі;
- реєстрація, створення, редагування та видалення стартапу;
- додавання картинок та їх збереження;
- додавання різного виду коментарів;
- пейджування та деякі інші функції.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Що таке краудфандинг Веб сайт: https://trends.rbc.ru/trends/innovation/60a4f17d9a79473292bfd627 (дата звернення: 02.06.2022).

2. Что такое краудфандинг: обзор платформ и советы начинающим. РБК. Веб сайт: https://trends.rbc.ru/trends/innovation/60a4f17d9a79473292bfd627/ (дата звернення: 03.05.2022).

3. Indiegogo. Веб сайт: https://www.indiegogo.com (дата звернення: 04.06.2022)

4. Kickstarter. Веб сайт: https://www.kickstarter.com/ (дата звернення: 04.06.2022)

5. Patreon. Веб сайт: https://www.patreon.com/ (дата звернення: 04.06.2022)

6. Experiment. Веб сайт: https://experiment.com/ (дата звернення: 04.06.2022)

7. Wspieram. Веб сайт: to https://wspieram.to/ (дата звернення: 04.06.2022)

8. Trumbowyg. Javascript текстовий редактор Веб сайт: https://alex-d.github.io/Trumbowyg/documentation/ (дата звернення: 13.05.2022)

9. Metanit. Сайт про програмування. Веб сайт: https://metanit.com/ (дата звернення: 12.06.2022)

10. Microsoft technical documentation. Офіційний сайт із документацією Майкрософт. Веб сайт: https://docs.microsoft.com/en-us/ (дата звернення: 15.06.2022)

11. Охорона праці на підприємстві: що потрібно знати? Веб сайт: https://te.dsp.gov.ua/ohorona-pratsi-na-pidpryyemstvi-shho-potribno-znaty/\ (дата звернення 27.05.2022)

12. Охорона праці в IT компанії. Веб сайт: https://legalitgroup.com/service/ohorona-praci-v-it-kompanii/ (дата звернення 27.05.2022)

13. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. Наказ: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення 27.05.2022)

14. Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень. Веб сайт: https://dnaop.com/html/2296/doc (дата звернення 27.05.2022)

15. Санітарні норми ультрафіолетового випромінювання у виробничих приміщеннях. Веб сайт: https://dnaop.com/html/2299/doc%20-%D0%A1%D0%9D_4557-88 (дата звернення 27.05.2022)

16. Природне та штучне освітлення. Веб сайт: https://dnaop.com/html/45036/doc-%20%D0%A1%D0%9D%D0%B8%D0%9F_II-4-79_ (дата звернення 27.05.2022)

17. Санітарні норми мікроклімату виробничих поміщень. Веб сайт: https://dnaop.com/html/34094/doc%20-%D0%94%D0%A1%D0%9D_3.3.6.042-99 (дата звернення 27.05.2022)

18. C# programming guide Веб сайт: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/ (дата звернення 17.05.2022)

19. Entity Framework Core Веб сайт: https://docs.microsoft.com/en-us/ef/core/ (дата звернення 19.05.2022)

20. Dependency injection Веб сайт: https://habr.com/ru/post/350068/(дата звернення 01.06.2022)