Хмельницький національний університет

Факультет програмування та комп'ютерних і телекомунікаційних систем

Кафедра інженерії програмного забезпечення

ДИПЛОМНИЙ ПРОЕКТ

Довідково-інформаційна система
Назва теми

комп'ютерного клубу

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр ДППЗ. 1701112.01.05.ПЗ

Виконав студент IV курсу група ІПЗ-17-1 _____ І. О. Зайцев
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ О. Г. Онишко
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Г. І. Радельчук
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

10 06 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем

Кафедра Інженерії програмного забезпечення

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
05 02 2021 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Зайцеву Ігорю Олександровичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Довідково-інформаційна система комп'ютерного клубу

Керівник проекту (роботи) Онишко Оксана Григорівна, канд. пед. наук, доцент

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 05.02.2021 р. № 11

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2021 р.

3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування програмного забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди)

6. Консультанти розділів дипломного проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Радельчук Г. І., доцент кафедри ІПЗ		
Антиплагіат	Гурман І. В., доцент кафедри ІПЗ		

7. Дата видачі завдання « 05 » лютого 2021р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних там ДП	01.12 – 30.12.2020	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2021	
3 Проектування програмного забезпечення	01.02 – 28.02.2021	
4 Програмна реалізація	01.03 – 10.04.2021	
5 Тестування програмного забезпечення	11.04 – 30.04.2021	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2021	
7 Попередній захист ДП	Травень 2021 (згідно графіка)	
8 Перевірка ДП на плагіат, нормконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки)	26.05 – 30.05.2021	
9 Підготовка до захисту та захист ДП	з 01.06.2021	

Студент _____ І. О. Зайцев
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ О. Г. Онишко
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проекту: «Довідково-інформаційна система комп'ютерного клубу».

Автор проекту: Зайцев Ігор Олександрович.

Керівник проекту: Онишко Оксана Григорівна.

Пояснювальна записка: 88 с., 19 рис., 11 табл., 3 дод., 15 джерел.

Графічна частина: 14 презентаційних слайдів.

ДОВІДКОВО-ІНФОРМАЦІЙНА СИСТЕМА, ВЕБ-ДОДАТОК, PHP, MY SQL, БАЗА ДАНИХ, CSS

Мета дипломного проекту: розробка довідково-інформаційної системи комп'ютерного клубу.

У дипломному проекті проведено аналіз предметної області та її інформаційного забезпечення, визначено вимоги до довідково-інформаційної системи, розроблено її архітектуру та спроектована структура бази даних комп'ютерного клубу та самого додатку.

При розробці програмної системи використано мову програмування PHP, сервер бази даних MySQL та таблиці стилів CSS.

Практична значимість отриманих результатів полягає у розробці готового до використання програмного продукту у вигляді веб-додатку.

Впровадження довідково-інформаційної системи дозволяє отимізувати роботу клубу, надає можливість розмежування роботи адміністратора та користувача і значно полегшити використання.

10.06.21
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	ДПІПЗ. 1701112.01.05.ПЗ	Пояснювальна записка	88		
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	14		

Змн.	Арк.	№ докум.	Підпис	Дата	ДПІПЗ.1701112.01.05.ВД		
Виконав		Зайцев І.О.		10.06	Довідково-інформаційна система комп'ютерного клубу. Відомість документів	Літ.	Арк. 1 / Акрушів 1
Керівник		Онишко О.Г.		10.06			
				10.06		ХНУ, ІПЗ-17-1	
Н. Контр.		Радельчук Г.І.		10.06			
Зав. Каф.		Бедратюк Л.П.		10.06			

ЗМІСТ

ВСТУП

Ще недавно захоплення комп'ютерними іграми визивало незадоволення батьків, але сьогодні дохід професійних кіберспортсменів може досягати мільйонів доларів. Останні десять років, ознаменувалися появою нового спортивного спрямування, в якому українці займають топові місця.

Кіберспорт на сьогоднішній день є найшвидше зростаючим сегментом індустрії розваг у всьому світі, і наша країна не є винятком. Глобальний ринок eSports в 2019 році становив $ 1 млрд, а до 2022 року повинен був досягти $ 1,8 млрд, з кожним роком додаючи в середньому по 22%. Це прогнози міжнародних агентств з аналітики ігрової індустрії і кіберспорту. Прогнози на 2022 роки ще більш оптимістичні – $ 2,3 млрд [8].

Не пустопорожня гра за комп'ютером, а напружене тренування – деякі комп'ютерні ігри на сьогоднішній день перетворилися на спортивні дисципліни. Кіберспортивні змагання стали явищем вселенського масштабу, а гравці часом піддаються таким же фізичним і психологічним навантаженням, як і в традиційних видах спорту.

На підйомі зростання кіберспорту розвивається і напрямок комп'ютерних клубів. Для них це основний вектор розвитку. Ця тенденція відбувається та відслідковується у всьому світі.

Наприклад, у Кореї, де кіберспортсменів навчають в університетах, а кіберспорт визнаний олімпійською дисципліною другого рівня, відкрито понад 30 тис. комп'ютерних клубів, які розташовані майже в кожному будинку.

У деяких містах України за останній рік число комп'ютерних клубів досягло 145%, а в столиці за цей самий період кількість таких закладів збільшилася приблизно в 1,7 рази.

Клубний сегмент кіберспорту виявився дуже гнучким до змін. Під час самоізоляції комп'ютерні клуби стали проводити для гравців онлайн турніри. За час пандемії деяким з них вдалося не тільки не втратити, але і збільшити нову аудиторію. Люди були позбавлені звичних для них розваг і класичного спорту і

з задоволенням грали в онлайн ігри. Зараз нова аудиторія попрямувала у клуби, що відновлюють свою роботу. Адже придбати ігровий комп'ютер і необхідну складові під силу не кожному гравцеві.

Зростаюча аудиторія комп'ютерних клубів потребує розвитку інфраструктури ринку, підвищення доступності клубів і арен, сучасних концепцій, високого рівня складових, а також нових форматів розваг. Технології віртуальної, доповненої або змішаної реальності цілком можуть отримати новий драйвер розвитку саме на майданчиках комп'ютерних клубів. До речі, одним з найбільш великих гравців ринку віртуальної реальності є фірма Valve, відома такими продуктами, як Counter-Strike або Dota 2, найбільш популярними іграми серед відвідувачів кіберклубів.

Кіберспорт називають змагання з комп'ютерних ігор, в яких беруть участь професійні гравці [15]. Найбільшим попитом з кіберспортивних дисциплін користуються: Dota 2, Counter-Strike: Global Offensive та Fortnite, League of Legends, StarCraft II. Найбільш відомі турніри – це The International (грають в Dota 2), Evolution Championship Series (файтинг), Intel Extreme Masters (Starcraft II, Counter-Strike: Global Offensive), Fortnite World Cup Finals та LoL World Championship і PUBG Global Championship, ESL Pro League. Ці заходи організовуються на найвищому рівні: з високоякісною відеотрансляцією і мільйонними призовими фондами для переможців [9].

Формально датою створення кіберспорту можна вважати 1972 рік – в цей періоді пройшли перші в світі згадані в пресі змагання по комп'ютерній грі Spacewar. Турнір мав назву «Міжгалактичні олімпійські ігри». Місце проходження було в одному з приміщень Стенфордського університету, а головним призом була річна підписка на журнал Rolling Stone.

Виробники цифрової техніки досить швидко зорієнтувались і виявили в новій індустрії великий потенціал. Вони почали фінансувати масштабні змагання з уже більш цінними призами: так, однією зі знакових дат в історії кіберспорту є червень 1997 року. Тоді американець Денніс «Thresh» Фонг став переможцем змагань Red Annihilation по Quake 2. У якості призу геймер

отримав Ferrari 328 GTS, яка раніше належала ведучому програмісту id Software і автору першого в світі шутера Doom Джон Кармак. Той турнір провела компанія Intergraph, яка купила студію id Software, яка розробила Quake.

Відтоді грошову винагороду в кілька десятків тисяч доларів для кібертурнірів стало звичним явищем. Переможець будь якого великого дуельного турніру по Quake 3 Arena, який проходив в Далласі, Джонатан «Fatal1ty» Вендел отримав нагороду $40 тисяч. У Швеції турнір в цій же грі одного і того ж року приніс переможцям $12 тисяч. Кілька місяців по тому в Quake 3 Arena змагалися вже в Великобританії, і кращі кіберспортсмени отримали чек на $10 тисяч.

Весь цей час кіберспорт залишався нішевим явищем. Зламним став 2000 рік, коли пройшов перший турнір World Cyber Games (WCG) – його дивилися мільйони глядачів, тисячі охочих брали участь у відбіркових етапах, і сотні людей грали в фіналі, стаючи професіоналами. Ця подія незабаром стала називати Олімпійськими іграми цифрового світу, а його головною метою стало просування комп'ютерних ігор в маси.

Кіберспортсмени WCG доводили свої вміння в таких дисциплінах, як WarCraft 3, Starcraft: Brood War, Counter Strike та Quake III Arena, FIFA, і з плином часом їх список поповнювався. Організатором змагань була однойменна компанія з кореї World Cyber Games Inc., яку фінансувала Samsung. Цей грошовий зв'язок виявився настільки міцним, що коли в лютому 2014 року Samsung вирішила закрити фінансування, WCG закрили.

Вчені German Sports University ретельно вивчили кібератлетів і прийшли до висновку, що їх без натяжки можна вважати справжніми спортсменами. Представники деяких дисциплін виконують по 400 рухів на клавіатурі і миші за одну хвилину. Це в чотири рази більше, ніж може нетренована людина. При цьому рухи асиметричні, що вимагає одночасної активності різних ділянок мозку. Такий рівень напруги не спостерігається в інших видах спорту.

Кількість гормону кортизолу у кіберспортсменів майже таке ж, як у автогонщиків. У поєднанні з високим пульсом це нагадує стан марафонця.

Кіберспорт також вимагає великої психологічної витривалості. Дослідження Університету Чичестера показує, що на топових турнірах кіберспортсмени стикаються з 51 типом стресових факторів. Серед них – проблеми комунікації в команді і страх виступу перед реальною аудиторією. Рівень психологічного навантаження, як у професійних спортсменів в традиційних спортивних дисциплінах, наприклад, футболі або регбі.

До речі, вже в 2024 році кіберспортсмени мають можливість поїхати на Олімпійські ігри. Про це вперше повідомив співпрезидент комітету з проведення Ігор у Парижі Тоні Естангет в 2017 році.

Кіберспорт, так само як і традиційні види спорту, є комерційним. Інвестори можуть придбати кіберспортивні команди за сотні мільйонів доларів, а спонсорські контракти на даний момент є одним з основних джерел фінансування індустрії. Також організатори турнірів отримують прибуток і з продажу квитків і прав на трансляцію.

«Продажі квитків на Чемпіонаті світу з футболу 2018 принесли ФІФА $541 млн, а доходи від ліцензування комп'ютерних ігор і спортивних онлайн-турнірів за той же чотирирічний період склали $600 млн. Якщо говорити про Dota 2, то призовий фонд турніру The International 2018 перевищив $25 млн, а кількість переглядів побило рекорди перегляду тенісного Вімблдону», – говорить старший юрист ЮФ Eterna Law Артем Кузьменко.

У 2018 році в кіберспортивні дисципліни грали більше двох мільярдів чоловік. Кількість глядачів зростає з року в рік. Цьогоріч, наприклад вона зросла на 18% і становить вже близько 380 млн.

У 1990-і і 2000-і кіберспортсменами можна було назвати завсідників комп'ютерних клубів: в їх локальних мережах щодня розгорталися перестрілки в шутерах і битви армій в стратегіях.

На міжнародний рівень українці вийшли в 2001 році, коли компанія Samsung провела в Україні відбір на World Cyber Games. Місцеві фінали проходили по всій країні, а переможці отримали оплачену поїздку на міжнародний турнір з призовим фондом в сотні тисяч доларів.

З 2006 року в світі почала стрімко набирати популярність гра DotA – кастомна гра на базі WarCraft 3. У 2011 році українська кіберспортивна команда Natus Vincere (NaVi) виграла в цій дисципліні $1 млн в чемпіонаті The International по Dota 2 [7]. Це привернуло увагу тисяч геймерів і спровокувало бум зростання кіберспорту.

До речі, саме Dota 2 стала джерелом значних заробітків для багатьох українських кіберспортсменів: з тих, хто знаходиться в топі-25 найбагатших українських кібератлетів, вісімнадцять досягли фінансового успіху за рахунок цієї дисципліни. Друга за кількістю зароблених грошей дисципліна – це Counter-Strike: Global Offensive, а третя – Hearthstone.

У нашій країні існує Федерація кіберспорту України, і вона теж активна: за два дні до 2020 року під її егідою відбувся перший офіційний фінал Кубка України по мобільному кіберспорту в дисципліні PUBG Mobile з призовим фондом в 200 000 грн. Також в грудні вона провела гранд-фінал першого чемпіонату України з кіберспорту з призовим фондом 1 млн грн.

Українці розвивають кіберспорт не тільки тут, але і на глобальних ринках. Так, керуючі партнери технологічного холдингу ТЕСНІА і esportainment-компанії WePlay! Esports Юрій Лазебніков і Олег Крот будують кіберспортивну арену в Лос-Анджелесі.

У цілому, представники індустрії сходяться на думці, що сьогодні кіберспорт в Україні не має перешкод для розвитку, і кіберспорт як бізнес все ще формується, постійно привертаючи увагу нових спонсорів.

Зважаючи на все вище викладене, можна зробити висновок про доцільність та актуальність створення довідково-інформаційної системи комп'ютерного клубу, яка повинна забезпечити злагоджену та ефективну взаємодію відвідувачів та власників клубу.

Мета проекту – розробка довідково-інформаційної системи комп'ютерного клубу.

Об’єкт дослідження – процес організації роботи та взаємодії відвідувачів та адміністратора клубуза допомогою довідково-інформаційної системи комп’ютерного клубу.

Предмет – система створення довідково-інформаційної системи комп’ютерного клубу.

Виходячи із мети, об’єкту та предмету дослідження можна окреслити завдання на дипломний проект:

- дослідити предметну область;
- здійснити аналіз існуючих програмних рішень даної галузі;
- розробити технічне завдання;
- спроектувати розробку програмного забезпечення;
- розробити інструкцію користувача.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Змістовний аналіз та опис предметної області, її структурних і функціональних особливостей

Хто сьогодні приходить в комп'ютерні клуби, що вони шукають, і як надихнути людей на те, щоб в епоху всепроникного інтернету вони приходили в ваш заклад, грали і найважливіше – поверталися. Для цього потрібно знати своїх гостей, і тільки тоді можна буде зробити так, щоб ці люди на протязі тривалого часу приносили дохід комп'ютерному клубу.

Перше, що потрібно знати про середньостатистичного клієнта – це чоловік віком до 30 років. Дівчата-геймери, всупереч розхожому стереотипу, теж існують, але досить часто уникають комп'ютерних клубів. Незважаючи на інший стереотип про те, що найбільш захопленими гравцями є хлопці шкільного віку, близько 75% гравців – молоді люди старші 20 років, студенти, що підробляють в офісах або на роботах, що не вимагають особливої кваліфікації. Проте, більшість з них здобуває вищу освіту, і, в перспективі, будуть скоріш за все офісними співробітниками, програмістами або опанують іншу творчу спеціальність.

Незважаючи на те, що денна активність людей вище, найбільша кількість гравці йде в комп'ютерні клуби ввечері і вночі.

Власники комп'ютерних клубів сходяться на тому, що досить часто ночами не вистачає вільних машин, тому можна запропонувати своїм клієнтам можливість бронювати комп'ютери заздалегідь.

Інший популярний типаж клієнта – це людина, яка заходить в комп'ютерний клуб попрацювати, терміново перевірити чи відправити пошту, роздрукувати щось. При цьому, досить часто такий клієнт погано розбирається в комп'ютерах – наприклад, самотні бабусі, які хочуть зв'язатися зі своїми дітьми чи онуками по Skype. Нехтувати такими клієнтами ні в якому разі не

можна – по-перше, важливість гарної репутації клубу ніхто не відміняв, а власникам не потрібні погані відгуки по «сарафанному радіо», а по друге, побачивши, що їм хочуть допомогти, такі клієнти будуть з радістю повертатися. Саме тому, що адміністратори в клубах, в масі своїй, лінуються допомагати клієнтам досить часто потік клієнтів з часом зменшується.

Як в процентному співвідношенні різняться клієнти? 60% відвідувачів все ж приходять пограти в ігри, в той час як 40% потрібно доступ до інтернету або до тієї чи іншої програми. З урахуванням специфіки – а гравцям властиво бути досить гучними – потрібно розділити зал свого клубу на кілька зон. Людям, яких цікавить робота, не потрібен такий самий потужний комп'ютер, як наприклад геймерам.

Потрібно врахувати, що основний дохід комп'ютерного клубу приносять постійні клієнти, які раз по раз повертаються пограти, і, досить часто, приводять своїх друзів. Тому, крім інформування аудиторії, завдання інтернет клубу – забезпечувати лояльність користувачів. Домогтися лояльності найпростіше, купуючи її. Потрібно створити в своєму клубі програму лояльності, яка буде пропонувати користувачеві накопичувальну знижку в залежності від кількості його візитів.

Ще один корисний момент, для розширення клієнтської бази – унікальні послуги. Школа комп'ютерної грамотності, тренувальна база для кіберспорту або банальні послуги з роздруківки матеріалів – все це допоможе збільшити доходи у кілька разів.

Як збільшити дохід комп’ютерного клубу?

Кожен власник бізнесу, пов'язаного з кіберспорту (комп'ютерний клуб, кібер-арена або інтернет кафе), прагне зробити його успішним і прибутковим. Для збільшення прибутку, підприємці, як правило, скорочують витрати або нарощують обсяг доходів. Оскільки клієнти комп'ютерного клубу приходять пограти на потужних ПК, про економію на обладнанні не може бути й мови. Тому власник змушений шукати додаткові можливості збільшення виручки. Наприклад, це може бути за рахунок збільшення загальної кількості клієнтів.

Однак, той факт, що комп'ютерний клуб є оф лайн бізнесом і прив'язаний до місця розташування, робить його кількість клієнтів обмеженим, а це не приведе до бажаного результату.

Як же ж можна збільшити продажі і почати заробляти більше?

Кращим рішенням буде розвиток дійсної клієнтської бази і збільшення середнього чека. Погодинні тарифи і абонементи складають основну частину чека, тому підвищення їх вартості може негативно відбитися на лояльності відвідувачів. Інший підхід, полягає в тому, щоб поліпшити досвід клієнтів і дозволити їм витрачати більше під час відвідування комп'ютерного клубу. Потрібно поліпшити клієнтський досвід, розширюючи свої послуги, зокрема, продаючи додаткові товари, такі як їжа, напої, футболки з фірмовими написами та атрибутику, послуги друку та багато іншого, а це означає, що потрібно буде вести фінансову звітність та облік кожного товару. Для цього потрібно створити довідково інформаційну систему.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Для аналізу програмно-технічного забезпечення було розглянуто декілька довідково-інформаційних систем. Першим є клуб «PlayHUB» [10], фрагмент довідково-інформаційної системи якого зображено на рисунку 1.1.

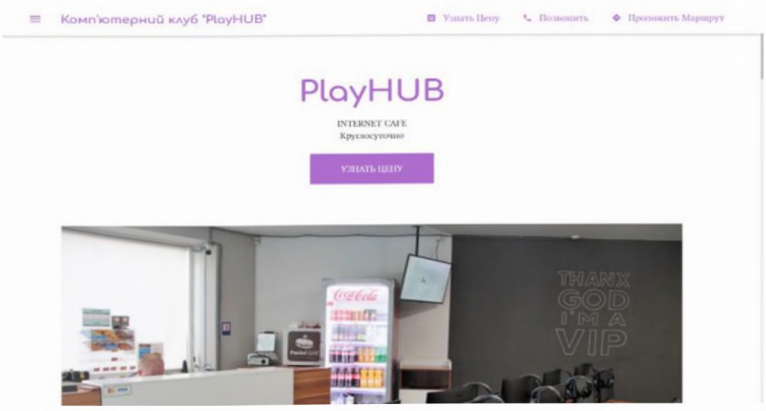


Рисунок 1.1 – Головна сторінка Комп’ютерного клубу PlayHUB

Якщо ми проаналізуємо дану довідково-інформаційну систему то можемо зробити висновок, що сторінка не інформативна. Вона не відображає головну суть системи. Інформація розташована в важкодоступних місцях і не є достатньою. Якщо зайти вкладку «Узнать цену» то для того щоб побачити її потрібно спочатку зареєструватись, визначити час коли відбудеться сеанс роботи, а вже аж тоді можна взнати ціну на послугу. Це досить довго і не зручно. Для літніх людей, яких ми теж хочемо залучити в клуб це практично не можливо з їхнім рівнем навичок. Тому, на мою думку, довідково-інформаційна система повинна бути максимально спрощеною і доступною чого в цій довідково-інформаційній системі немає.

Наступна довідково-інформаційна система комп'ютерного клубу «It land» [13], зображена на рисунку 1.2, є більш яскравою, але, як і в попередній, малоінформативна. Вона розрахована на сучасного молодого користувача і не враховує людей середнього та похилого віку, які теж починають все більше користуватись комп'ютерами для пошуку потрібної інформації.

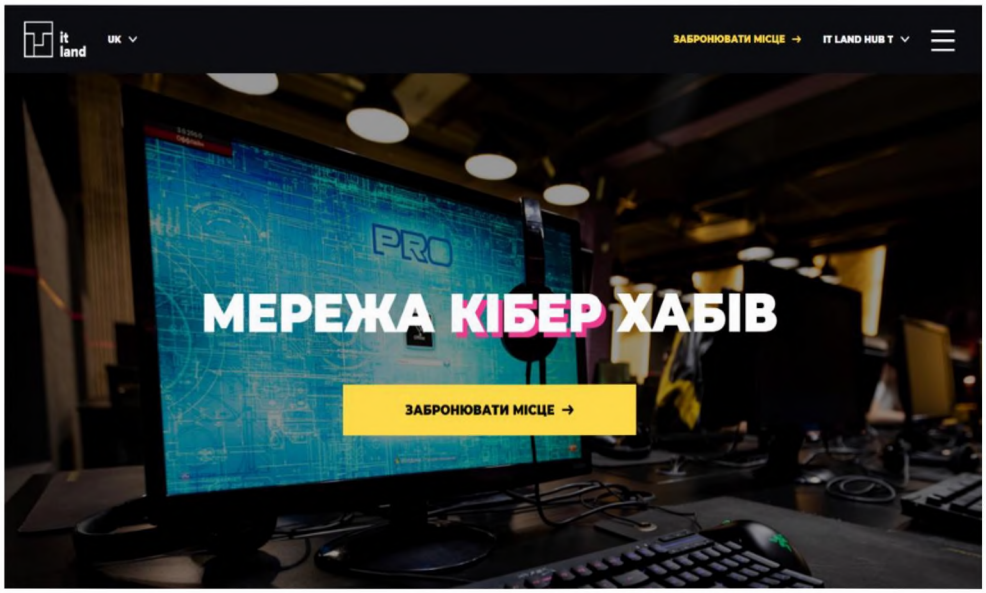


Рисунок 1.2 – Довідково-інформаційна система комп'ютерного клубу «It land»

Гарний приклад – це вкладка «Забронювати місце», що зображена на рисунку 1.3. Якщо ми подивимось на цю вкладку то зрозуміємо, що як

забронювати місце зрозуміло тільки постійним клієнтам і повну інформацію про роботу може знати тільки виключно адміністратор.

Рисунок 1.3 – Довідково-інформаційна система комп’ютерного клубу It land, вкладка «Забронювати місце»

1.3 Визначення вимог до програмного забезпечення та розробка технічного завдання

Що ж повинно бути в нашій довідково-інформаційній системі?

Перш за все вона повинна бути розбита на два інтерфейси: для відвідувачів та адміністратора клубу.

При входженні в систему відвідувач повинен зареєструватися та створити свій обліковий запис.

В інтерфейсі вони можуть поповнювати свій баланс, вибирати тарифи і абонементи, переглядати і запускати різні ігри, перевіряти новини вашого клубу, а також купувати товари в модулі «Магазин».

З іншого боку, адміністратори клубу можуть керувати всіма комп'ютерами і їх резервами, розміщувати новини, оновлювати список доступних ігор,

відстежувати кількість доступних ПК, обробляти замовлення клієнтів, а також вивантажувати звіти.

Таким чином, використовуючи такі системи, можна не тільки організувати процес продажів за допомогою модуля «Магазин» в інтерфейсі геймера, а й управляти всім комп'ютерним клубом. Варто також врахувати, що починаючи з 2020 року під час пандемії COVID-19 комп'ютерні клуби повинні дотримуватися встановлених епідеміологічних правил, і програмне забезпечення повинно бути достатньо гнучким, щоб допомогти бізнесу адаптуватися до нових норм.

Що ж повинна вирішувати довідково-інформаційна система комп'ютерного клубу?

Крім стандартних функцій для управління комп'ютерним клубом, система повинна бути призначена для задоволення всіх потреб, пов'язаних з процесом продажів. Весь робочий процес, повинен бути автоматизований починаючи з того, як клієнт дізнається про доступні у продажі товари, і закінчуючи фінансовими звітами. Система повинна включати в себе онлайн каталог товарів з переліком додаткових товарів і послуг де повинні бути такі категорії як їжа, напої, послуги і засоби безпеки (це маски, антисептики тощо). З огляду на останні зміни, викликані пандемією COVID-19, ця функція також допоможе зберегти дистанцію між відвідувачами, так як їм не потрібно вставати з-за ПК, щоб зробити замовлення.

Управління асортиментом і залишками

Як тільки почне продаватися товар через модуль «Магазин», потрібно буде обраховувати залишки. Без системи управління залишками, з товарами, які проходять через комп'ютерний клуб, буде повна плутанина. Тому потрібен модуль який дозволяє вести централізований облік кожного товару, забезпечуючи єдине джерело правдивої інформації про продавця, ціни продажу і покупки, специфікаціях і кількості наявних на складі одиниць товару. Це однозначно допоможе дотримуватись запобіжних заходів від COVID-19, при ефективному управлінні складом, і виключить можливість необхідного

відсутності товару, тим самим усунувши необхідність для відвідувачів виходити за їжею, напоями або масками.

Оформлення замовлення

За допомогою онлайн-каталогу клієнти інтернет клубу повинні мати можливість оформити онлайн замовлення і відправити його прямо в інтерфейс адміністратора для обробки. Це позбавить співробітників, а саме адміністратора, від паперової тяганини.

Відстеження статусу замовлення і отримання повідомлень

Досить важко відповідати весь час на численні запити клієнтів про їх замовлення. Це відволікає працівника від основної роботи. І оскільки, в силу пандемії коронавіруса люди змушені тримати соціальну дистанцію, будь-які взаємодії між людьми стали ще більш небажаними. За допомогою модуля "Магазин" в оболонці клієнта відвідувачі повинні мати можливість відстежувати стан замовлення в режимі реального часу, сидячи за комп'ютером, а менеджер обробити замовлення швидше, так як він отримує миттєві повідомлення і може зв’язатись з клієнтом онлайн.

Інтегрована каса

При продажу додаткових товарів, створення каси стане обов'язковим інструментом. З інтегрованим касовим модулем повинна бути можливість створювати квитанції в один клік і обробляти операції за готівковим або безготівковим розрахунком.

Історія замовлень

Управління фінансами і підготовка щомісячних звітів стають ще більш болючими, коли все ведеться в паперовому вигляді. Цифрові рішення в цьому випадку значно економлять час. Інформаційно-довідкова система повинна дозволяти клієнтам і співробітникам бачити історію покупок.

Виходячи з нашого досвіду, надійне рішення для управління комп'ютерним клубом має включати в себе наступні можливості:

– інтерфейс для клієнта і адміністратора;
– карту управління ПК;

– модуль резервування ПК;
– цінові пропозиції та програму лояльності;
– модуль «Магазин»;
– онлайн-каталог товарів;
– модуль управління складом;
– створення замовлень;
– систему повідомлень і статуси відстеження замовлень;
– вбудовану касову систему;
– історію замовлень для ведення фінансових звітів.

Виходячи з вище сказаного можна зробити висновки про те, що існує безліч способів збільшити дохід комп'ютерного клубу. Але лише деякі з них не завдають шкоди лояльності клієнтів. Кращим способом збільшення доходу, виходячи з наших досліджень, є створення такої довідково-інформаційної системи яка дозволить збільшення середнього чека за рахунок розширення асортименту товарів і послуг. Більш того, це позитивно позначиться на якості обслуговування клієнтів. Щоб уникнути паперової тяганини, людських помилок і зосередитися на пріоритетах, потрібно власнику комп'ютерного клубу використовувати довідково-інформаційну систему.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Проектування архітектури програмного забезпечення

Перед початком розробки будь-якої програми слід ретельно розписати алгоритми її роботи, структуру компонентів, структуру бази даних і т.д. Для цього призначені UML-діаграми. UML (англ. Unified Modeling Language) – уніфікована мова моделювання, що використовується у парадигмі об'єктно-орієнтованого програмування.

Ця мова є складовою частиною уніфікованого процесу розробки програмного забезпечення. Ця мова є мовою широкого профілю, вона є відкритим стандартом, що використовує графічні позначення у процесі створення абстрактної моделі системи, під назвою UML-модель. UML була винайдена для визначення, візуалізації, проектування й документування в основному програмних систем.

Ця мова моделювання може бути використана на всіх фазах життєвого циклу аналізу бізнес-систем та розробки прикладних програм.

UML створювалася для кращої організації процесу розробки програмних систем, що дозволяє покращити ефективність їх реалізації у декілька разів і помітно поліпшити якість кінцевого продукту.

В UML проектуванні є наступні види діаграм:

- діаграми класів;
- діаграми компонентів;
- діаграми прецедентів;
- діаграми діяльності (алгоритми);
- діаграми прецедентів;
- діаграми послідовностей;
- діаграми об’єктів;
- діаграми розгортки (розміщення).

Незважаючи на те, що UML є загально визнаним стандартом мови моделювання, вона часто піддається критиці через такі причини:

- надмірність мови;
- неточна семантика;
- труднощі у вивченні та застосуванні;
- візуальна неоднорідність.

Діаграма прецедентів – в UML, це діаграма, яка показує відношення між акторами та прецедентами в системі.

Діаграма прецедентів являється графом, що включає в себе множини акторів та прецедентів (варіантів використання) які обмежені границею системи (прямокутник), асоціацій між акторами та прецедентами, залежностей серед прецедентів, та залежностей узагальнення між акторами. Діаграми прецедентів показують елементи моделі варіантів використання.

Основний зміст даної діаграми знаходиться в наступному: проектована система подається у вигляді безлічі сутностей чи акторів, що працюють із системою за допомогою так званих варіантів використання. Варіант використання вживають для опису послуг, які система пропонує актору. Іншими словами, кожен варіант використання надає деякий набір дій, які опрацьовує система при діалозі з актором. При цьому не визначається те, яким чином буде реалізована спільна робота акторів із системою.

Діаграма компонент – в UML, діаграма, на якій створюються компоненти, залежності та зв'язки між ними.

Діаграма компонент показує залежності між компонентами програмного забезпечення, та складається з складових вихідних кодів, бінарних складових, та складових, що можуть бути виконаними.

Модуль програмного забезпечення може бути зображено як компоненту. Деякі компоненти зустрічаються під час компіляції, інші – під час компонування, або в процесі роботи програми. Діаграма компонент показує лише структурні характеристики тому для відображення окремих екземплярів компонент слід використовувати діаграму розгортання.

Діаграма компонентів розробляється для таких цілей:

- візуалізації загальної структури початкового коду програмної системи;
- специфікації використаного варіанту програмної системи;
- забезпечення використання окремих фраґментів програмних кодів декілька раз;
- подання концептуальної і фізичної схем баз даних

У створенні діаграм компонентів приймають участь як системні аналітики й архітектори, так і програмісти. Діаграма компонентів створює узгоджений перехід від логічного подання до відповідної реалізації проекту у вигляді програмних кодів. Одні компоненти можуть бути створені тільки на етапі компіляції програмних кодів, інші – в частині його виконання.

Діаграма компонентів показує загальні залежності між компонентами, та розглядає їх як класифікатори.

Діаграма класів це статичне представлення структури моделі. Вона показує статичні елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів може включати в себе позначення для пакетів та вкладених пакетів. Додатково, діаграма класів може включати позначення деяких елементів поведінки, але їх розвиток розкривається в інших типах діаграм. Діаграма класів застосовується для представлення статичної структури моделі системи в позначенні класів об'єктно-орієнтованого програмування.

На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, та відносини між ними.

Для кращої роботи довідково-інформаційної системи комп’ютерного клубу бажаним є забезпечення можливості реєстрації та авторизації користувачів. На рисунку 2.1 зображено вигляд авторизації користувача схематично за допомогою UML діаграми прецедентів.

Повний опис прецедента дає програмісту повну інформацію про функціонування кожного модуля програми. Це звільняє його від придумування

непередбачених в документації модулів та дозволяє повністю присвятити себе програмній реалізації продукту.

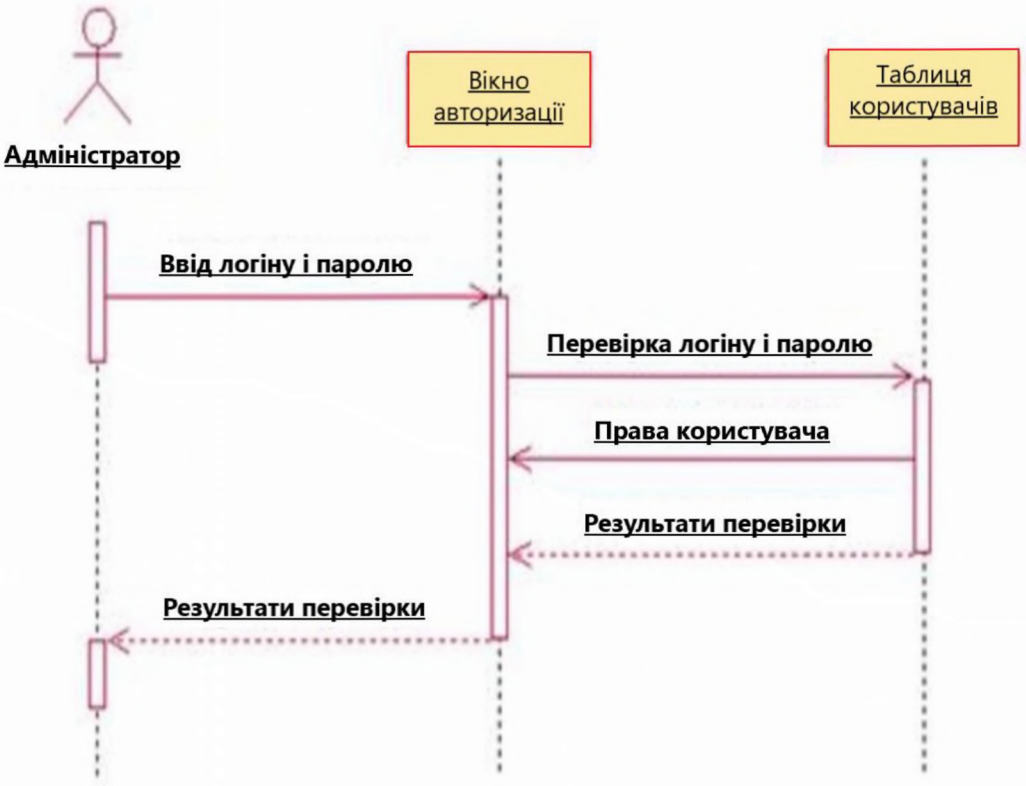


Рисунок 2.1 – Діаграма послідовності для прецедента «Авторизація»

Далі дамо характеристику засобам програмного забезпечення, якими будемо користуватись для створення довідково-інформаційної системи комп'ютерного клубу.

До найбільш популярних можна віднести PHP та MySQL. Переглянемо і проведемо аналіз даних засобів в порівнянні з іншими відомими на сьогодні засобами реалізації проекту.

ASP (Active Server Pages) є більш вживаною «мовою» сценаріїв Microsoft. Взагалі, ASP – це не мова, а розширення у Visual Basic для створення сценаріїв.

Тому всякому, хто знайомий з Visual Basic, відносно легко освоїти ASP. Але є і недоліки. Наприклад, ASP як правило працює повільніше, ніж PHP. Основу ASP утворює архітектура, яка створена на COM. Тому коли програма ASP робить звернення до бази даних або здійснює виведення даних для клієнта,

це робиться за посередництва ком-об'єктів інших сервісів NT або рівнів операційної системи. Ці навантаження, які пов'язані з COM навантаження накопичуватися і приводити до того, що у всі рази, окрім видачі простих сторінок коли трафік має середню потужність, продуктивність буде невисокою. Інший недолік - це ASP не завжди може підходити для перенесення на інші платформи і інтеграції за допомогою засобами GNU, та з середовищами і серверами open source. Беручи до уваги що ASP є фірмовою системою Microsoft, вона як правило застосовується з її ж Internet Information Server (IIS), через що ASP як правило вибирають обмежено – для 32- розрядних систем Windows, тому що для багатьох серверів ця технологія є безкоштовним застосуванням. Існують версії ASP і для UNIX такі як, ChilliSoft ASP, і ряду інтерпретаторів ASP для різних систем і веб-серверів, але для них вартість системи якщо враховувати її продуктивності може виявитися досить високою. Проте технологія ASP.NET достатньо відрізняється.

В перспективі ASP може суттєво покращити свою продуктивність та можливість масштабування. Але реальних переваг можна досягти тільки при умові значних витрат на різні додаткові сервери.

PHP може працювати практично на будь-яких платформах, а версії Cold Fusion підходять тільки для Win32, Solaris, Linux і HP/UX. Для PHP потрібно фундаментальних початкових навиків програмування, на відміну від Cold Fusion з покращеним інтегрованим середовищем розробки (IDE) і покращеними мовними конструкціями. PHP не має таких вимог до ресурсів. Так як PHP розробленим спеціально для Інтернету, то він кращий в цій області ніж Perl, оскільки Perl створювався для багатьох використань, що відбразилося на його вигляді. Форма і синтаксис Perl роблять складнішим читання сценаріїв Perl і їх модифікацію, тоді коли вона потрібна. Не зважаючи на те, що Perl користуються достатньо довго і він широко підтримується, цей засіб став складною конструкцією з доповнень та розширень, що є часто просто надлишковими.

Формат PHP кращий для сприйняття, а при збереженні його гнучкості PHP простіше інтегрується з вже існуючим HTML і пропонує функціональність,

яка схожу з Perl, але із значно з більшою витонченістю. PHP простіше використовувати, ніж Java, за його допомогою легко будувати веб-сервера-застосування, які володіють тими ж перевагами гнучкості і масштабованості. Використовуючи PHP, не потрібно володіти 5-річним досвідом розробки програмного забезпечення, для того щоб створювати прості динамічні сторінки – для цього досить бути спостережливим, навіть маючи невеликий досвід програмування. Крім того, Java може обходиться дорожче, оскільки в більшості компаній як правило встановлюють іншу машину для Java Enterprise і користуються Oracle або іншим дорогим програмним забезпеченням. Не зважаючи на це PHP потребує подальшого розвитку, оскільки не володіє такою ж тотожністю і багатьма іншими зручними можливостями, одними з яких є пул об'єктів або відображення баз даних, які є в Java. Отже мова PHP є кращим варіантом для розробки програмного коду у взаємодії з базою даних сайту.

Під CMS, тобто системою керування контентом, або Content Management System, визначають програмне забезпечення, яке використовується при організації та забезпечення процесу створення, управління і редагування тексту, мультимедійних чи графічних елементів сайту в мережі Інтернет або в локальних комп'ютерних мережах.

Є багато різних систем управління контентом, які створені за допомогою різних технологій, серед яких є багато різних безкоштовні і платних програм. Основним елементом будь-якої CMS є сховища інформації. У багатьох сучасних CMS сховищах інформації є реляційна база даних. Тобто вона складається з таблиць з встановленими відношеннями.

Якщо CMS хоче зберегти інформацію, то вона заносить її до бази даних, в якій для кожної сутності відводиться окрема таблиця.

Інформацію для відображення CMS бере із бази даних. Для того, щоб побачити інформацію в форматі HTML використовуються шаблони. Шаблон – це файл, який в собі містить дизайн сторінки, який створений засобами спеціальної мови. Наприклад, це може бути HTML-код певним чином

розмічений, в якому міститься інформація яка показує місцезнаходження елементів, які можна брати з бази даних.

Іншою частиною CMS є система користувачів та їх ролей. Під роллю користувача мається на увазі набір тих дій, які користувач має змогу здійснювати в даний час. В новітніх CMS, зазвичай, можна визначити кілька користувацьких ролей, наприклад, адміністратор, користувач відвідувачі. Певному користувачу можна привласнити свою роль, причому надання ролей може виконувати або адміністратор, або це можна робити автоматично.

Широкому впровадженню та використанню CMS на сьогоднішній день сприяє досить багато причин. Одна із основних це ускладнення функціоналу сучасних сайтів. Створення одного і того самого програмного модулю кожного разу не є раціональним, тому почалось програмування спеціальних бібліотек з корисними функціями, а потім з'явились спеціальні рішення, за допомогою яких розпочалась розробка універсальних CMS.

Іншим, не менш досить важливим аргументом став процес спрощення самих CMS. Досить велика кількість хостингів пропонують можливість загрузки готових безкоштовних чи платних версій CMS.

Схема роботи CMS така: нехай користувач увійшов на сайт, який знаходиться під керуванням CMS, і виконав там певну дію. Перш за все, CMS повинна зрозуміти, яка саме реакція повинна бути на цю дію. Далі модуль, що відповідає за обробку користувацького запиту, фіксує інформацію, що надійшла, до бази даних. Система проводить запит до бази даних та отримує інформацію, що необхідно відобразити, або формує повідомлення.

Інформація з бази даних пересилається до модуля, який бере поточну тему і додає в неї всю необхідну інформацію. В результаті виконання цих дій створюється HTML-код, який відсилається до користувача. Результат виконаної роботи користувач бачить зразу в себе на дисплеї.

Які ж переваги CMS:

– використання шаблонів дозволяє автоматично змінити зовнішній вигляд всього сайту, незалежно від його наповнення;

– вміст взагалі відокремлений від дизайну сайту, що максимально спрощує процес редагування вмісту сайту.

– більша частина CMS це модульна архітектура, а її функціонал можна збільшувати за допомогою плагінів і модулів.

– більша частина CMS має технічну підтримку, в основному — у вигляді інтернет-спільноти користувачів, що можуть не тільки надавати поради іншим користувачам, але і брати участь у розробці і вдосконаленні CMS, що гарантує весь час вдосконалення таких CMS.

– використання CMS сприяє економії часу розробки сайту.

Існують безкоштовні і платні системи управління контентом. Як правило, більшою популярністю користуються безкоштовні системи. Найбільш уживаними є WordPress, Joomla і Drupal. Дамо характеристику кожному з них.

WordPress – досить популярна CMS, яка дозволяє програмувати різноманітні сайти, але в основному вона є движком для блогів.

WordPress складається з багатьох локалізацій, для неї розроблено велику кількість додаткових модулів і шаблонів. До недоліків WordPress належать низька швидкість роботи сайтів та можливість збоїв при занадто високій відвідуваності.

Joomla – ще одна популярна в світі CMS, яка складніша в освоєнні та використанні, ніж WordPress, проте має і більш широку сферу використання.

Для CMS Joomla розроблена досить велика кількість різноманітних модулів, завдяки чому її можна назвати універсальною CMS. Також для Joomla є велика кількість шаблонів, які дають необмежені можливості для створення дизайну сайту. Крім того, Joomla має сумісність з різними серверами, зокрема Linux, FreeBSD, MacOSX server, Solaris і AIX. Завдяки цьому її можна широко використовувати, не беручи до уваги сервер. До недоліків Joomla відноситься недостатній захист системи від зломів та можливі складнощі які виникають з індексацією сайтів пошуковими системами.

CMS Drupal – являється одним із кращих варіантів при створенні блогів, онлайн- енциклопедій, інтернет-спільнот, форумів та ін. Drupal притаманні всі

можливості, які необхідні для ефективного управління вмістом сайту. Також для Drupal є багато шаблонів і модулів. З недоліків - дана CMS має менше документації, ніж для Joomla і WordPress, низька швидкість завантаження сторінок та менш дружній інтерфейс.

Якщо порівняти уживаність охарактеризованих CMS на даний час то можна зробити висновок, що найбільш використовуваною згідно діаграми, зображеної на рисунку 2.2, на березень 2021 року CMS є WordPress [6].

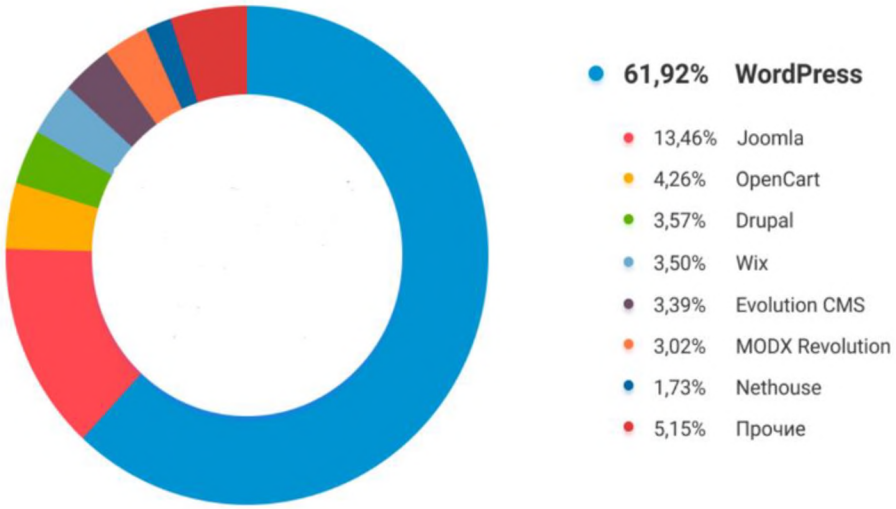


Рисунок 2.2 – Рейтинг безкоштовних CMS на березень 2021 року

Якщо подивитись на відмінні риси платних CMS та безкоштовних, то можна звернути увагу на їх універсальність, наявність кількох версій починаючи від безкоштовних і закінчуючи дорожчими, в яких більш потужний функціонал іможливість створення будь-якого сайту. Серед платних CMS найбільш популярним є Бітрікс, на якій працює безліч різних сайтів. Досить непоганими CMS можна відзначити ABO.CMS, NetCat, Amiro.CMS, UMI.CMS. Досить часто, велика кількість розробників надає перевагу у використання для сайтів CMS власної розробки.

2.2 Проектування структури бази даних

MySQL є швидкою та в той же час стабільною системою. Це основна причина її широкого вжитку. У грудні 2002 року електронне видавництво «eWeek» провело та опублікувало результати тестування основних систем по управлінню базами даних, в тому числі такі як Oracle, Microsoft SQL Server, DB2 I MySQL. За результатами тестування вони вивели MySQL і Oracle 9 на перше місце по продуктивності.

Система MySQL доступна у двох версіях: безкоштовного програмного забезпечення, та в комерційному виконанні. Для цього виробником використовується спеціальна «подвійна» ліцензійна схема.

Все програмне забезпечення MySQL можна отримати безкоштовно із загальнодоступною ліцензією, але в тих випадках, якщо потрібна комерційна версія, то цю систему також можна купити. MySQL може підтримувати більшість можливостей, що є важливими для спільноти користувачів та розробників баз даних. Наприклад, транзакції, блокування на рівні стрічок, зовнішні ключі, підзапити та повнотекстовий пошук.

Система MySQL багато раз перевірена та надійна. Вона використовується багатьма компаніями. Наприклад такими як Yahoo!, Finance, Slashdot. MySQL є чудовим засобом для вивчення баз даних тому що вона проста в інсталяції та використанні, а також, виключно малими вимогами по відношенню до оперативної пам’яті та дискового простору.

Наша база даних яка створена для інформаційно-довідкової системи комп'ютерний клуб зображена на рисунку 2.3.

Дамо опис таблиць що входять в цю базу даних для побудови інформаційно-довідкової системи комп'ютерного клубу.

Записи

$wpdb->posts - таблиця де записуються пости, постійні сторінки, довільні типи записів, вкладення тощо.

$wpdb->postmeta – уточнює таблицю $wpdb->posts. Зберігає додаткові дані записів.

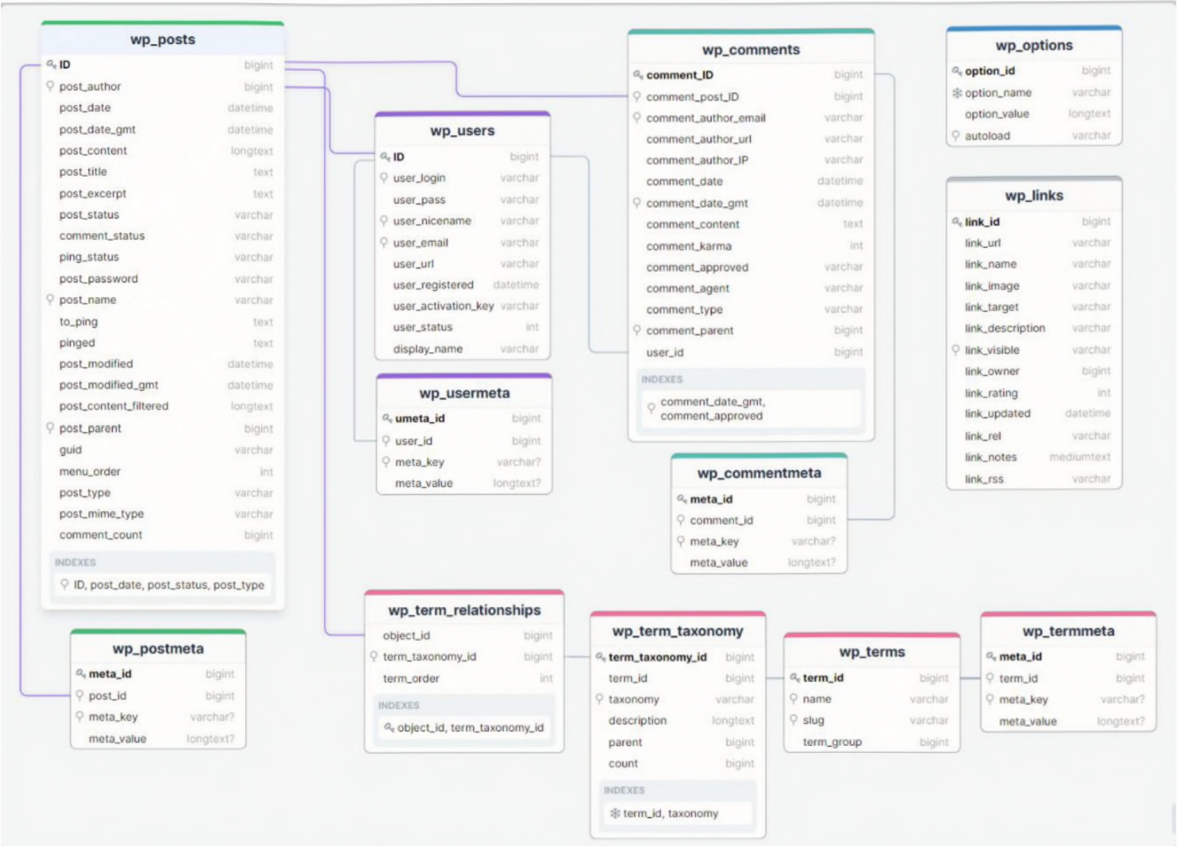


Рисунок 2.3 – База інформаційно-довідкової системи комп'ютерний клуб

Користувачі

$wpdb->users - таблиці з даними про користувачів, що зареєструвались.

$wpdb->usermeta - додаткова інформація про користувачів, така як Ім'я, Нік, права та інші Метаполя для таблиці $wpdb->users.

Коментарі

$wpdb->comments - таблиця з записами коментарів.

$wpdb->commentmeta - метаполя для таблиці $wpdb->comments.

Таксономії

$wpdb->terms - таблиця яка містить інформацію про назви категорій, міток, категорій посилань та інших таксономій.

$wpdb->termmeta - таблиця в якій знаходяться додаткові поля для таблиці $wpdb->terms.

$wpdb->term_taxonomy - таблиця з інформацією про різні таксономії та їх опис.

$wpdb->term_relationships - таблиця яка зв'язує таксономії з контентом

Інші таблиці

$wpdb->links - таблиця в якій зберігаються записи посилань.

$wpdb->options - таблиця опцій (налаштувань).

В даному розділі було спроектовано архітектуру програмного забезпечення та також структуру бази даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Реалізація бази даних

Будь-які дані які створюються повинні десь зберігатися. В програмуванні для цього використовують бази даних.

Проаналізуємо деякі основні типи баз даних.

Типами баз даних, які називаються також моделями бази даних, є шаблони і структури, які використовуються для організації даних в системі управління базами даних (СУБД). Вибір типу впливає на те, які операції зможе виконувати додаток та як будуть представлені дані.

Прості структури даних

В простих структурах даних перший і найпростіший спосіб зберігання це текстові файли. Такий метод застосовується і сьогодні для роботи з невеликими обсягами інформації. Розглянемо це на рисунку 3.1.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
bob:x:1000:1000:Bob Smith,,,:/home/bob:/bin/bash
```

Рисунок 3.1 – Приклад бази даних в текстових файлах

Недоліки таких баз даних полягають в наступному:

– обмежений тип і рівень складності інформації, що зберігається;

– важко встановити зв'язки між компонентами даних;

– відсутність функцій паралелізму.

Тому вони використовуються тільки для систем з невеликими вимогами до читання і запису, та можуть бути призначені для зберігання конфігураційних даних. При їх створенні немає необхідності в сторонньому забезпеченні.

На відміну від текстових таблиць, в ієрархічних базах даних з'являються зв'язки між об'єктами. В них кожен запис має одного «батька», що створює деревоподібну структуру, де записи класифікуються за їхніми стосункам з ланцюжком батьківських записів. Це відображено на рисунку 3.2.

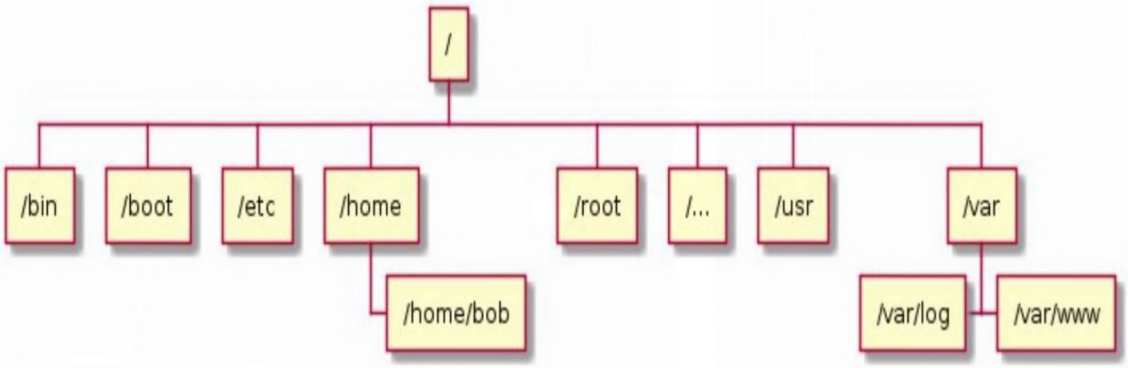


Рисунок 3.2 – Приклад побудови ієрархічних зв’язків

Які ж є особливості цих баз даних:

– інформація організована у вигляді дерева з відносинами «предок-нащадок»;

– кожен запис може мати не більше одного з батьків;

– зв'язки між записами виконані у вигляді фізичних покажчиків;

– неможливо реалізувати відносини «багатьох до багатьох».

В мережевих базах даних розширюють функціональність ієрархічно. Тому записи можуть мати більше одного батька. Це означає, що можна моделювати складні відносини. Розглянемо це на рисунку 3.3.

Особливості цих баз даних полягають в тому, що мережеві бази даних подаються не деревом, а загальним графом але вони обмежені тими ж шаблонами доступу, що ієрархічні бази даних.

Реляційні бази даних – найстаріший тип досі широко використовуваних баз даних загального призначення. Дані та зв'язки в них між ними організовані за допомогою таблиць. Кожен стовпець у таблиці має ім'я і тип. Кожен рядок представляє окремий запис або елемент даних в таблиці, який містить значення для кожного з стовпців.

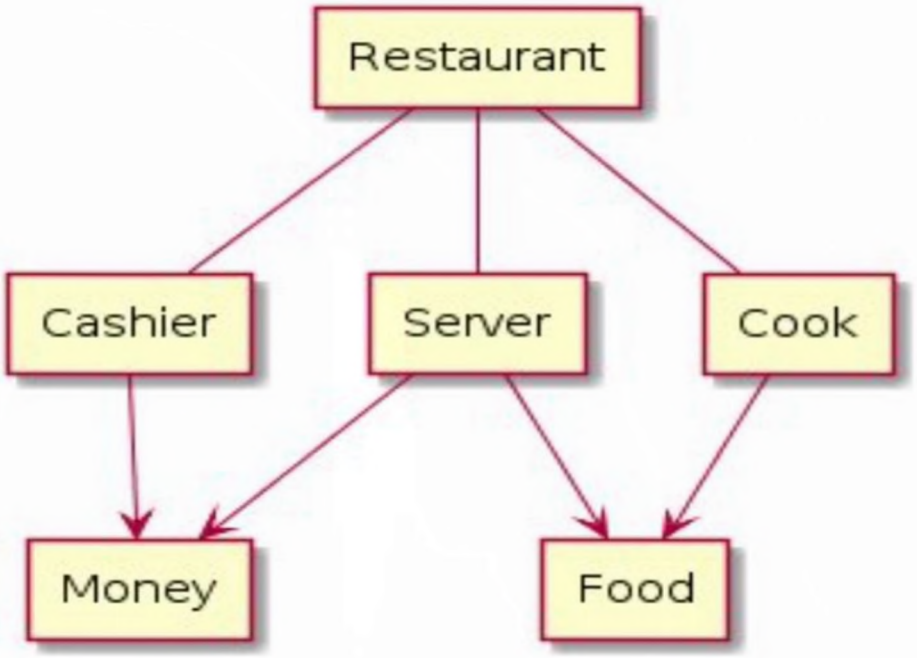


Рисунок 3.3 – Приклад зв’язків в мережевій базі даних

В реляційних базах даних поле в таблиці, назване зовнішнім ключем, може містити посилання на стовпці в інших таблицях, що дозволяє їх з'єднувати. Завдяки високоорганізованій структурі і гнучкості реляційні бази даних є потужними і такими, що адаптуються до різних типів даних. Для доступу до даних в них використовується мова структурованих запитів SQL. Тому ця база даних є надійним вибором для багатьох додатків. Вона зображена на рисунку 3.4.

Документні бази даних (також документоорієнтовані бази даних або сховища документів), спільно використовують базову семантику доступу і пошуку сховищ ключів і значень.

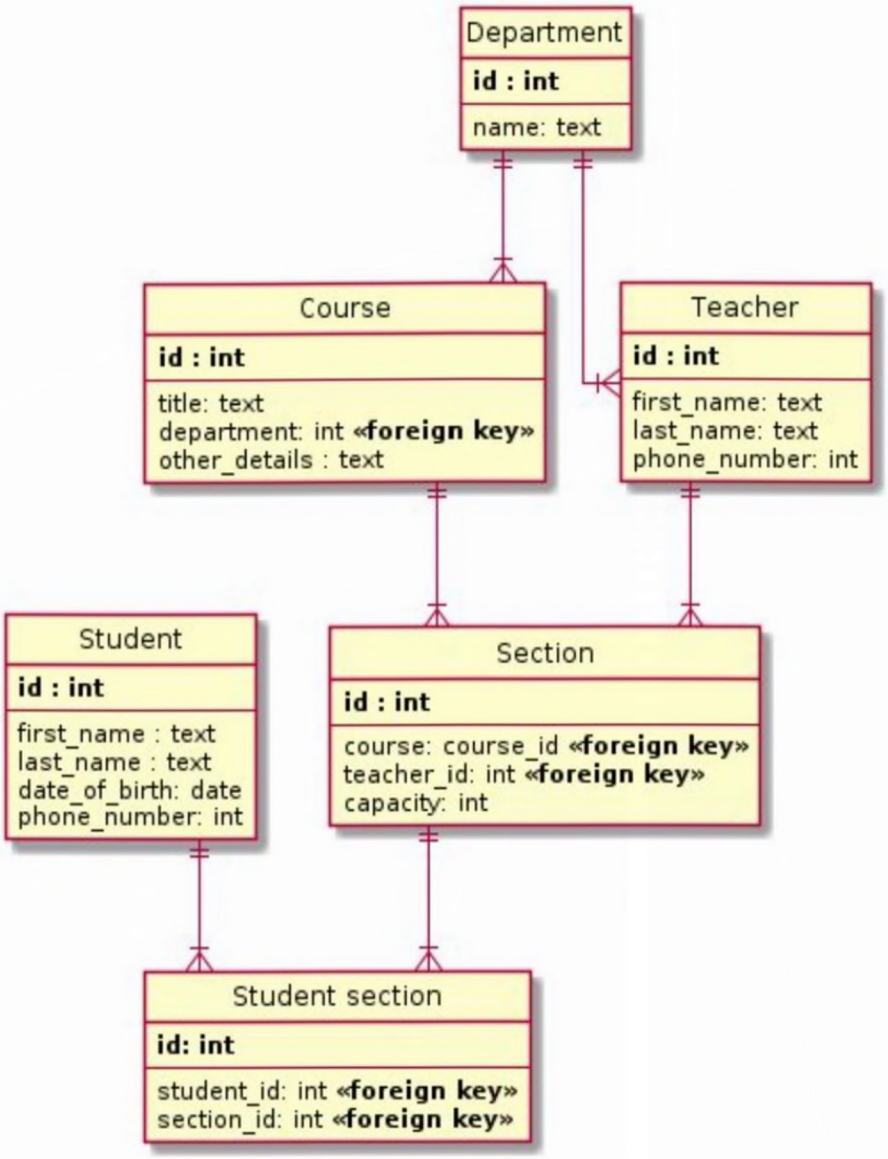


Рисунок 3.4 – Приклад реляційної бази даних

Такі бази даних можуть використовувати ключ для унікальної ідентифікації даних. Різниця між сховищами «ключ-значення» і документними базами даних полягає в тому, що замість зберігання blob-об'єктів, документоорієнтовані бази зберігають дані в структурованих форматах - JSON, BSON або XML. Її можна переглянути на рисунку 3.5.

Особливості документних баз даних полягають в тому, що база даних не виділяє окремий формат або схему, кожен документ може мати свою внутрішню структуру. Документні бази даних є хорошим вибором для швидкої розробки, де у будь-який момент можна змінювати властивості даних, не

змінюючи структуру або самі дані. В графових базах даних замість зіставлення зв'язків з таблицями і зовнішніми ключами, встановлюють зв'язки, використовуючи вузли, ребра і властивості.

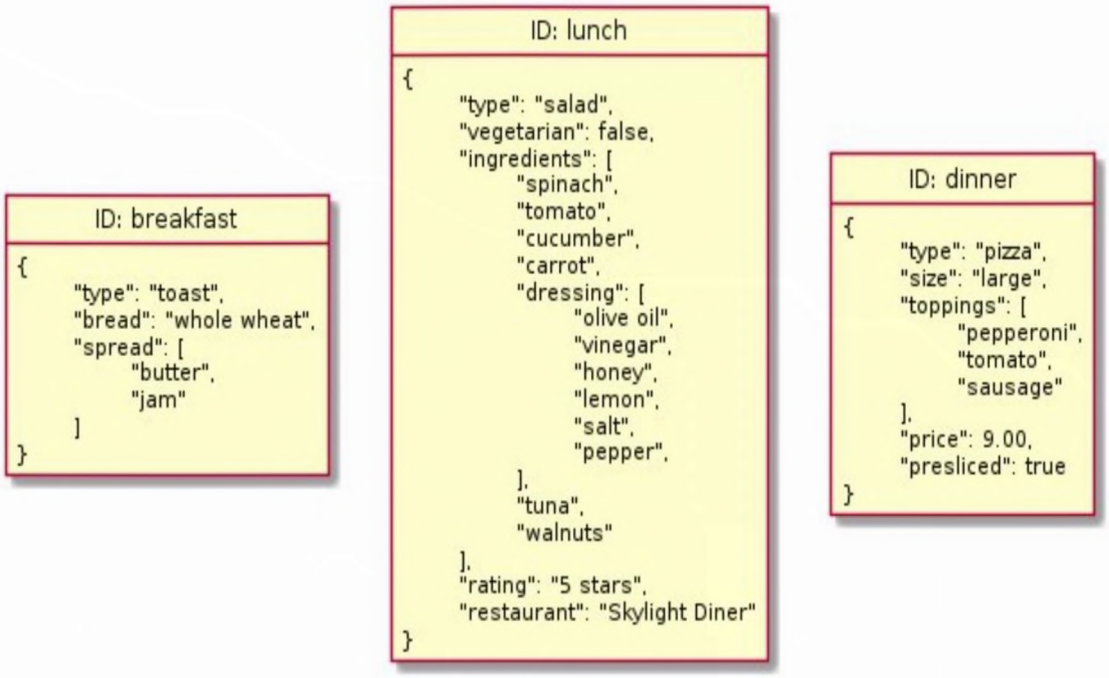


Рисунок 3.5 – Приклад Документної бази даних

Графові бази представляють дані у вигляді окремих вузлів, які можуть мати будь-яку кількість пов'язаних з ними властивостей. Графові бази даних виглядають аналогічно мережевим і фокусуються на зв'язках між елементами. Ці бази даних явно відображають зв'язки між типами даних і не вимагають покрокового обходу для переміщення між елементами. В них немає обмежень в типах зв'язків. Переглянемо її приклад на рисунку 3.6.

Проаналізувавши більшу частину можливих видів баз даних було зроблено висновок, що для створення довідково-інформаційної системи потрібно скористатись реляційною базою даних.

Всі данні довідково-інформаційної системи зберігаються в таблицях реляційної бази даних. Опис таблиць бази даних зображено в таблиці 3.1.

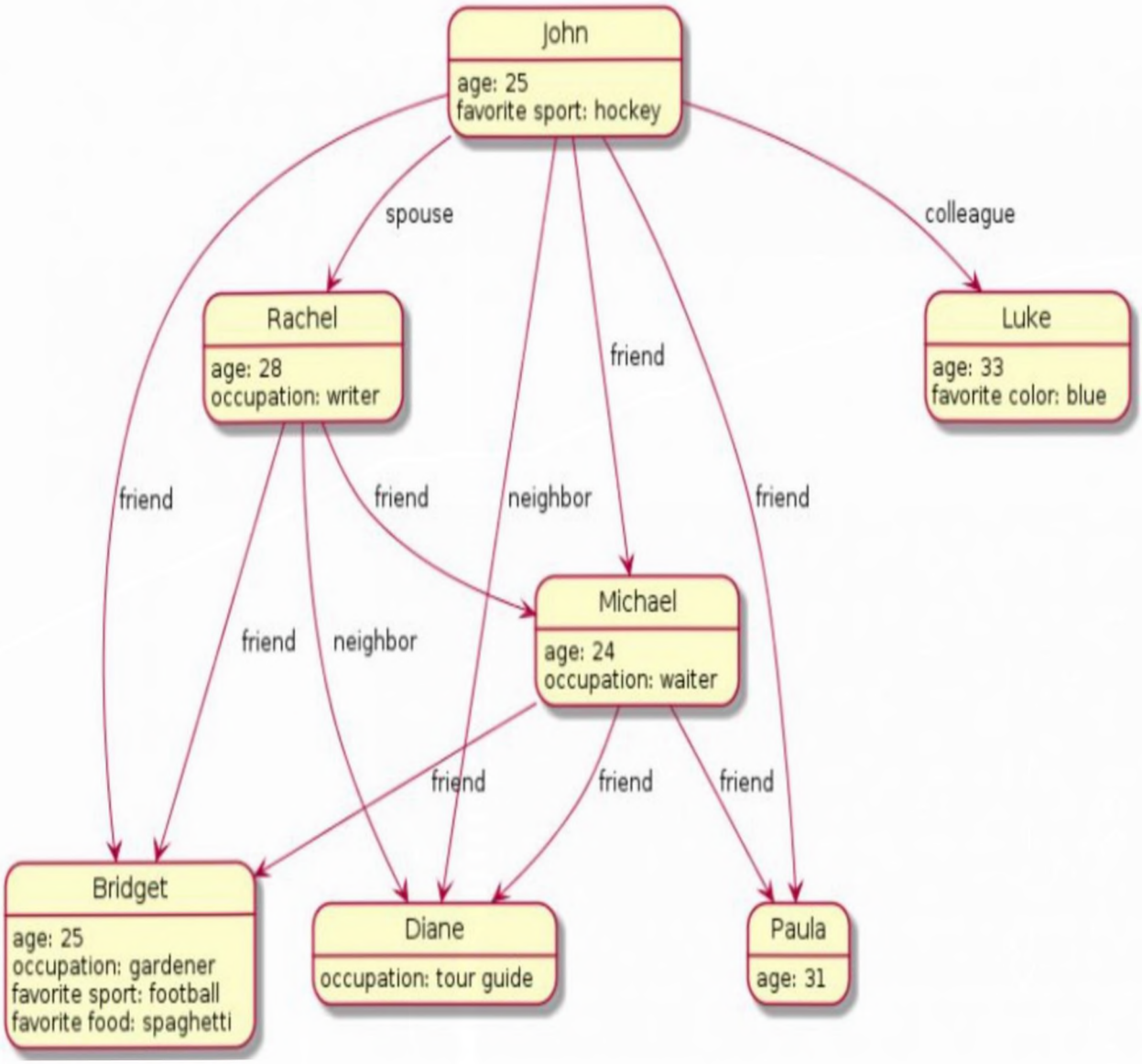


Рисунок 3.6 – Приклад Графової бази даних

Таблиця 3.1 – Опис таблиць бази даних

Ім’я таблиці	Опис
1	2
posts	Основна таблиця в бази де зберігаються всі статі (заголовок, зміст) та вся потрібна службова інформація (дата створення, автор, тип і т.д.).
postmeta	Додаткові данні які відносяться до публікацій, які відсутні в стандартних полях таблиці posts.

Кінець таблиці 3.1

1	2
users	Відомості про користувачів: в якості авторів статей, так і про користувачів з можливістю їх коментувати
usermeta	Додаткові поля, що потрібні для опису користувачів та доповнення до таблиці users.
comments	Зберігаються всі коментарі для всіх наявних статей.
terms	Простий список рубрик, до яких може відноситися стаття. Невелика частина службової інформації про рубрики.
term_taxonomy	Таблиця служить для систематизації рубрик, вона створює дерева рубрик
term_relationships	Тут задається відповідність статті і рубрик , до якої стаття входить
links	Зберігаються посилання та інформація по ним.
options	Загальні налаштування движка.

Розглянемо ці таблиці докладніше. Так, щоб можна було аналізувати вміст таблиць. Підключимося до потрібної бази:

mysql -u user -h host -p database

Виберемо кодування для вмісту (UTF-8):

mysql> set names utf8;

Переглядати всі таблиці можна за допомогою простих запитів, таких як:

mysql> SELECT * FROM posts LIMIT 10;

Переглянемо таблиці які використані для збереження записів, чорновиків та бібліотек медіафайлів та зображення.

Код для створення цієї таблиці в MySQL:

```
CREATE TABLE `posts` (
  `ID` bigint(20) unsigned NOT NULL auto_increment,
  `post_author` bigint(20) unsigned NOT NULL default '0',
  `post_date` datetime NOT NULL default '0000-00-00 00:00:00',
  `post_date_gmt` datetime NOT NULL default '0000-00-00 00:00:00',
  `post_content` longtext NOT NULL,
  `post_title` text NOT NULL,
```

```
`post_category` int(4) NOT NULL default '0',
`post_excerpt` text NOT NULL,
`post_status` varchar(20) NOT NULL default 'publish',
`comment_status` varchar(20) NOT NULL default 'open',
`ping_status` varchar(20) NOT NULL default 'open',
`post_password` varchar(20) NOT NULL default '',
`post_name` varchar(200) NOT NULL default '',
`to_ping` text NOT NULL,
`pinged` text NOT NULL,
`post_modified` datetime NOT NULL default '0000-00-00 00:00:00',
`post_modified_gmt` datetime NOT NULL default '0000-00-00 00:00:00',
`post_content_filtered` text NOT NULL,
`post_parent` bigint(20) unsigned NOT NULL default '0',
`guid` varchar(255) NOT NULL default '',
`menu_order` int(11) NOT NULL default '0',
`post_type` varchar(20) NOT NULL default 'post',
`post_mime_type` varchar(100) NOT NULL default '',
`comment_count` bigint(20) NOT NULL default '0',
PRIMARY KEY (`ID`),
KEY `post_name` (`post_name`),
KEY `type_status_date` (`post_type`,`post_status`,`post_date`,`ID`),
KEY `post_parent` (`post_parent`)
)
```

Детально розглянемо зміст нашої таблиці «Збереження записів, чорновиків, бібліотеки медіа файлів».

Таблиця 3.2 – Зміст таблиці «Збереження записів, чорновиків, бібліотеки медіа файлів»

Поле	Опис
1	2
ID	Ідентифікатор запису або первинний ключ. Значення – від 1 до 263-1. Максимальне значення комірки, що є 20-розрядним числом.
post_author	Автор статті, який позначається ідентифікатором ID таблиці users.
post_date	Дата публікації, час – місцевий.
post_date_gmt	Дата публікації, час – за Гринвічем.
post_content	Вміст статті. При збереженні і виведенні на сторінку вміст проходить через фільтр замін, тому на екрані можна побачити не те, що ми вводили.
post_title	Заголовок статті чи ім'я для зображення в бібліотеці медиафайлів (посилання на них, теж повинні зберігатися в цій таблиці).
post_category	Завжди дорівнює 0
post_excerpt	Коротка витримка, інформація про запис у вигляді цитати. Вводиться самостійно, тому що за замовчуванням відсутня.
post_status	Статус запису. Publish - опублікована, draft – чернетка, inherit - пов'язана (може бути, зображення або резервна копія запису).
comment_status	Чи є можливість провести коментування: open – дозволяється коментувати, closed - не дозволяється.
post_password	Пароль в явному вигляді, якщо стаття з паролем. В іншому випадку – ні.

Кінець таблиці 3.2

1	2
post_name	Назва сторінки для статичного посилання. Записи тут зберігаються автоматично і зміни задають це ім'я автоматом.
post_parent	Для резервних копій та зображень – ID батьківської запису.
Guid	Повна інтернет-посилання на запис (об'єкт).
menu_order	Дорівнє 0.
post_type	Тип запису: post – звичайний запис, page – запис з постійним посиланням (статична), revision – автозбереження, attachment – мультимедійний файл (наприклад, зображення).
post_mime_type	Тип вмісту запису. Важливий і потрібний тільки для прикріплень (наприклад, png–зображення проходить як image / png).
comment_count	Кількість коментів до запису.

Код для створення таблиці в MySQL:

```
CREATE TABLE `postmeta`
(
`meta_id` bigint(20) unsigned NOT NULL auto_increment,
`post_id` bigint(20) unsigned NOT NULL default '0',
`meta_key` varchar(255) default NULL,
`meta_value` longtext,
PRIMARY KEY  (`meta_id`),
KEY `post_id` (`post_id`),
```

KEY `meta_key` (`meta_key`)
)

Таблиця 3.3 – Вміст таблиці posts.

Поле	Опис
meta_id	Первинний ключ.
post_id	До якої записи таблиці posts відносяться наші метадані.
meta_key	Ім'я додаткового параметра. Використовується для прикріплень файлів та інших метаданих до статті.
meta_value	Значення додаткового параметру. Вміст залежить від імені параметра.

Таблиця 3.4 – Вміст таблиці users

Поле	Опис
ID	Ідентифікатор, первинний ключ.
user_login	Ім’я користувача для входу.
user_pass	Пароль користувача для входу, в закодованому вигляді.
user_nicename	Як звертатись до користувача. Як правило, збігається з ім'ям для входу.
user_email	Електронна пошта для зв’язку .
user_url	Адреса сайту користувача.
user_registered	Дата реєстрації користувача.
user_activation_key	Активаційний ключ. Після активації видаляється.
display_name	І’мя для відображення.

Ігноруючи існування таблиці users, всі пов'язані з користувачем дані знаходяться тут.

Код для створення таблиці в MySQL:

```
CREATE TABLE `usermeta` (
  `umeta_id` bigint(20) unsigned NOT NULL auto_increment,
  `user_id` bigint(20) unsigned NOT NULL default '0',
  `meta_key` varchar(255) default NULL,
  `meta_value` longtext,
  PRIMARY KEY  (`umeta_id`),
  KEY `user_id` (`user_id`),
  KEY `meta_key` (`meta_key`)
)
```

Таблиця 3.5 – Вміст таблиці usermeta

Поле	Опис
umeta_id	Ідентифікатор метазапису, первинний ключ.
user_id	Ідентифікатор користувача, який відноситься до запису.
meta_key	Додатковий параметр
meta_value	Значення додаткового параметру.

Код для створення таблиці в MySQL:

```
CREATE TABLE `comments` (
  `comment_ID` bigint(20) unsigned NOT NULL auto_increment,
  `comment_post_ID` bigint(20) unsigned NOT NULL default '0',
  `comment_author` tinytext NOT NULL,
  `comment_author_email` varchar(100) NOT NULL default '',
  `comment_author_url` varchar(200) NOT NULL default '',
  `comment_author_IP` varchar(100) NOT NULL default '',
  `comment_date_gmt` datetime NOT NULL default '0000–00–00 00:00:00',
  `comment_content` text NOT NULL,
  `comment_approved` varchar(20) NOT NULL default '1',
```

```
  `comment_agent` varchar(255) NOT NULL default '',
  `comment_type` varchar(20) NOT NULL default '',
  `comment_parent` bigint(20) unsigned NOT NULL default '0',
  `user_id` bigint(20) unsigned NOT NULL default '0',
  PRIMARY KEY  (`comment_ID`),
  KEY `comment_approved` (`comment_approved`),
  KEY `comment_post_ID` (`comment_post_ID`),
  KEY `comment_approved_date_gmt` (`comment_approved`,`comment_date_gmt`),
  KEY `comment_date_gmt` (`comment_date_gmt`)
)
```

Таблиця 3.6 – Вміст таблиці comments

Поле	Опис
comment_ID	Ідентифікатор коментарів, первинний ключ.
comment_post_ID	Ідентифікатор заміток, до яких відноситься коментар.
comment_author_email	Електронна пошта
comment_author	Автор, використовується з реєстраційних даних або вводиться вручну, якщо реєстрація для коментаря не потрібно.
comment_author_url	Сайт автору коментарів.
comment_author_IP	IP–адреса тих хто коментує.
comment_date	Дата та час
comment_content	Самі коментарі.
comment_approved	Чи схвалено коментар для показу.
comment_agent	З якого браузеру або системи заходили.
comment_parent	Батьківський комент. Потрібен для деревовидної структури.
user_id	Ідентифікатор зареєстрованого користувача.

Код для створення таблиці в MySQL:

```
CREATE TABLE `terms` (
  `term_id` bigint(20) unsigned NOT NULL auto_increment,
  `name` varchar(200) NOT NULL default '',
  `slug` varchar(200) NOT NULL default '',
  `term_group` bigint(10) NOT NULL default '0',
  PRIMARY KEY  (`term_id`),
  UNIQUE KEY `slug` (`slug`),
  KEY `name` (`name`)
)
```

Таблиця 3.7 – Вміст таблиці terms

Поле	Описание
term_id	Ідентифікатор.
name	Назва рубрики. Використовується для визначення рубрики практично скрізь, наприклад, під записом або в віджетах рубрик.
slug	Ярлик рубрики. Зазвичай містить тільки латинські букви в нижньому регістрі, цифри і дефіси.
term_taxonomy	Таблиця для об'єднання рубрик в дерева і для опису взаємозв'язків рубрик.

Код для створення таблиці в MySQL:

```
CREATE TABLE `term_taxonomy` (
  `term_taxonomy_id` bigint(20) unsigned NOT NULL auto_increment,
  `term_id` bigint(20) unsigned NOT NULL default '0',
```

```
  `taxonomy` varchar(32) NOT NULL default '',
  `description` longtext NOT NULL,
  `parent` bigint(20) unsigned NOT NULL default '0',
  `count` bigint(20) NOT NULL default '0',
  PRIMARY KEY  (`term_taxonomy_id`),
  UNIQUE KEY `term_id_taxonomy` (`term_id`,`taxonomy`),
  KEY `taxonomy` (`taxonomy`)
)
```

Таблиця 3.8 – Вміст таблиці term_taxonomy

Поле	Описание
term_taxonomy_id	Ідентифікатор, первинний ключ.
term_id	Ідентифікатор рубрики.
taxonomy	category – рубрика, link_category – рубрика посилань.
description	Короткий опис.
parent	Ідентифікатор батьківської рубрики.
count	Кількість асоційованих записів.
term_relationships	Таблиця для зв'язку статті з рубриками.

Код для створення таблиці в MySQL:

```
CREATE TABLE `term_relationships` (
  `object_id` bigint(20) unsigned NOT NULL default '0',
  `term_taxonomy_id` bigint(20) unsigned NOT NULL default '0',
  `term_order` int(11) NOT NULL default '0',
  PRIMARY KEY  (`object_id`,`term_taxonomy_id`),
  KEY `term_taxonomy_id` (`term_taxonomy_id`)
)
```

Таблиця 3.9 – Вміст таблиці term_relationships

Поле	Опис
object_id	Який об’єкт описуємо
term_taxonomy_id	З яким зв'язком рубрик асоціюємо.

Код для створення таблиці в MySQL:

```
CREATE TABLE `links` (
  `link_id` bigint(20) unsigned NOT NULL auto_increment,
  `link_url` varchar(255) NOT NULL default '',
  `link_name` varchar(255) NOT NULL default '',
  `link_image` varchar(255) NOT NULL default '',
  `link_target` varchar(25) NOT NULL default '',
  `link_category` bigint(20) NOT NULL default '0',
  `link_description` varchar(255) NOT NULL default '',
  `link_visible` varchar(20) NOT NULL default 'Y',
  `link_owner` bigint(20) unsigned NOT NULL default '1',
  `link_rating` int(11) NOT NULL default '0',
  `link_updated` datetime NOT NULL default '0000–00–00 00:00:00',
  `link_rel` varchar(255) NOT NULL default '',
  `link_notes` mediumtext NOT NULL,
  `link_rss` varchar(255) NOT NULL default '',
  PRIMARY KEY  (`link_id`),
  KEY `link_category` (`link_category`),
  KEY `link_visible` (`link_visible`)
)
```

Загальні налаштування движка.

Код для створення таблиці в MySQL:

```
CREATE TABLE `options` (
 `option_id` bigint(20) unsigned NOT NULL auto_increment,
 `blog_id` int(11) NOT NULL default '0',
 `option_name` varchar(64) NOT NULL default '',
 `option_value` longtext NOT NULL,
 `autoload` varchar(20) NOT NULL default 'yes',
 PRIMARY KEY  (`option_id`,`blog_id`,`option_name`),
 KEY `option_name` (`option_name`)
)
```

Створення сторінок є однією з найважливіших і невід'ємних частин роботи зі створення сайту.

Редагувати сторінку і її вміст можна у вікні звичайного текстового редактора, вбудованого в систему, або у вікні редагування html–коду. Це дозволяє людям, які не особливо добре розбираються в html створювати простенькі сторінки, а решті – збільшити їх функціональність і можливості при редагуванні коду вручну. Можна створити материнську сторінку, у яку будуть поміщені підсторінки, або скористатися функцією "Меню" і розмістити на головній сторінці каталоги, в яких будуть розміщені всі інші. Використовуючи функцію «Меню» і публікації сторінок, на сайті «Красилівська районна адміністрація» сторінки були заповнені усілякою інформацією та файлами. Були додані фото, аудіо–і відеофайли.

3.2 Структура та функціональне призначення модулів програми

Якщо ми розглянемо ті функції які повинна виконувати наша система то ми побачимо, що перш за все довідково–інформаційна система розбита на два блоки: це блок адміністратора і блок користувача.

Блок адміністратора включає такі вкладки:

– Адміністрування послуг;
– Користувачі;
– Керування класом;
– Послуги;
– Додаткові налаштування.

У вкладці Адміністрування послуг знаходяться такі розділи, як Черга де Адміністратор бачить послідовність замовлень та їх виконання, в підсистемі Створити – формується саме замовлення.

Вкладка Користувача розбита на Переглянути, де можна переглянути всіх хто працює в даний часу комп'ютерному клубі, їх електронну пошту, ім'я, по–батькові, надіслати їх повідомлення, побачити загальну суму по послугам, видалити користувача і Додати де додаються послуги та користувачі. Вікно із розділу Переглянути зображено на рисунку 3.7.

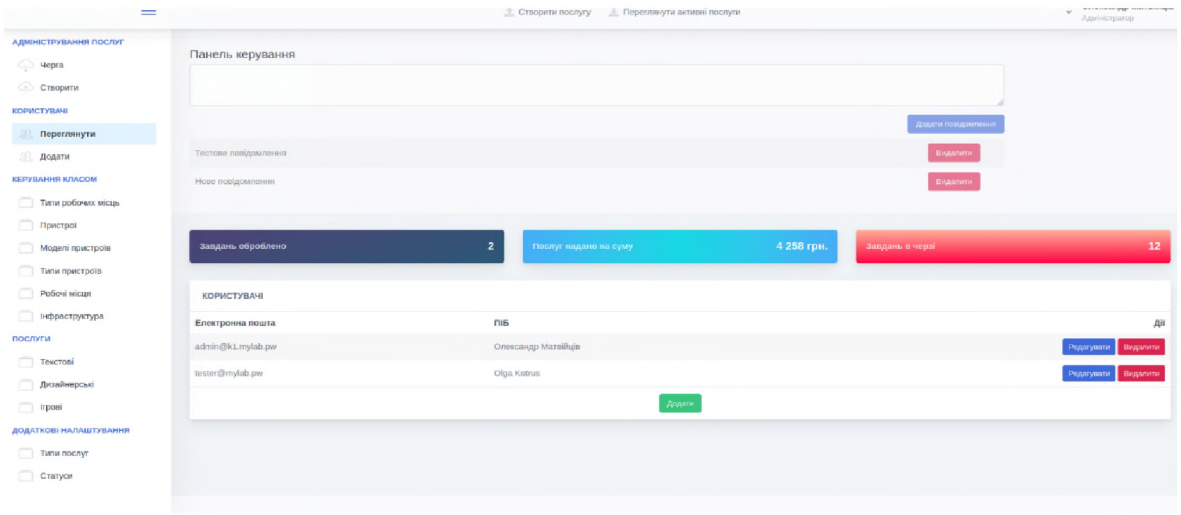


Рисунок 3.7 – Розділ «Переглянути» вкладки «Користувачі»

У вкладці Керування класом знаходяться розділи такі як Типи робочих місць, Пристрої, Моделі пристроїв, Типи пристроїв, Робочі місця, Інфраструктура.

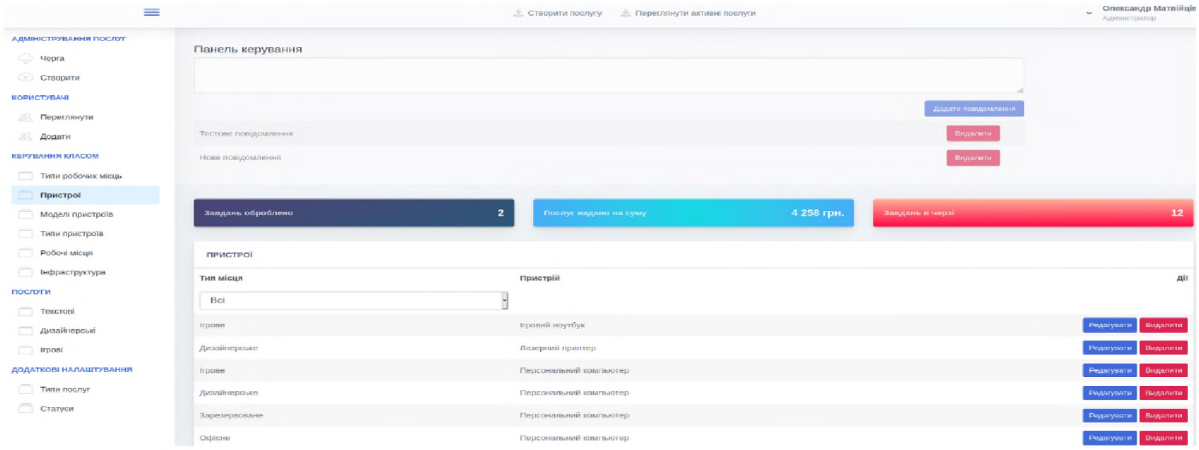


Рисунок 3.8 – Вкладка «Пристрої»

Розглянемо розділ Пристрої більш детально.

В цьому розділі адміністратор може вибрати тип місця і пристрій яким буде користуватись. Одночасно беручи до уваги епідеміологічну ситуацію він може розміщувати користувачів на безпечній для них відстані.

У вкладці Послуги є такі розділи як Текстові, Дизайнерські та Ігрові.

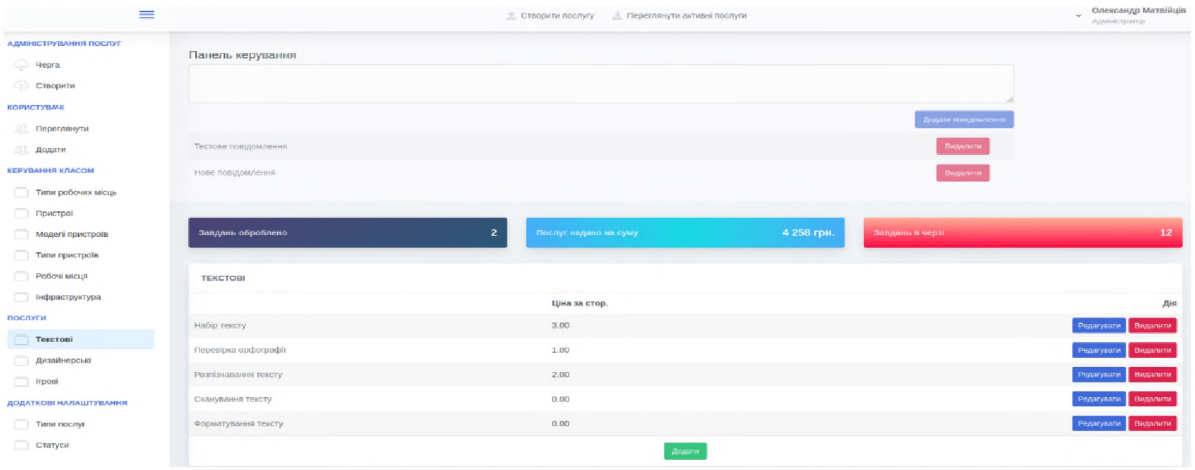


Рисунок 3.9 – Вкладка «Послуги» розділ «Текстові»

Тут адміністратор бачить які з послуг обрав користувач. Наприклад, як зображено на Рисунок у розділі Текстові видно що адміністратор відслідковує які послуги обрав користувач. Це може бути і набір тексту. І перевірка орфографії, і розпізнавання тексту та багато іншого.

У вкладці Додаткові налаштування є розділи Типи послуг та Статуси. Де можна слідкувати які типи послуг обрав користувач і статус користувача.
Блок Користувача призначений для відвідувачів клубу.

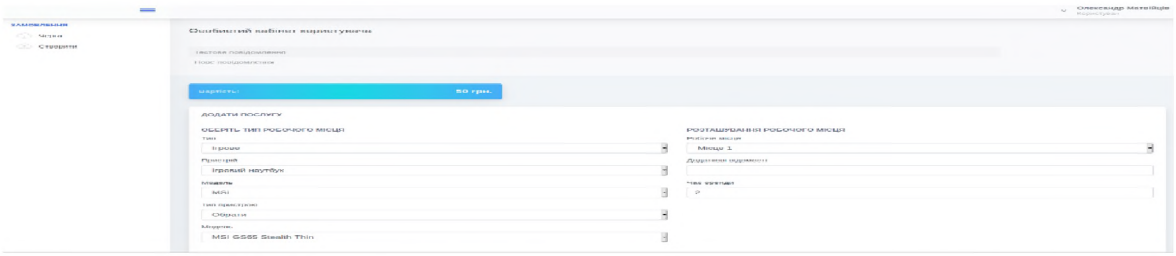


Рисунок 3.10 – «Особистий кабінет користувача»

Тут вони можуть вибрати тип робочого місця, тип комп'ютера відповідно до їх робочих потреб, місце розташування комп'ютера, зарезервувати час оренди. Зразу побачити вартість вибраних послуг.

3.3 Інструкція користувача

Наведемо невеличку інструкцію для користувача для того щоб можна було користуватись даною довідково–інформаційною системою.

Для тієї частини яка видна користувачеві нам необхідно дати перш за все наступні вказівки.

Спочатку потрібно зайти на Веб–сайт довідково–інформаційної системи. Далі для кращого розуміння та перегляду всіх можливостей нашої системи бажано зареєструватися. Після цього він заходить в кабінет користувача де бачить всі послуги які він використовує в даний час.

У вікні «Замовити» може переглянути доступний час для відвідування, що досить важливо в даний час, коли діють карантинні обмеження. Тут відвідувач бачить вільний час та номер робочого місця.

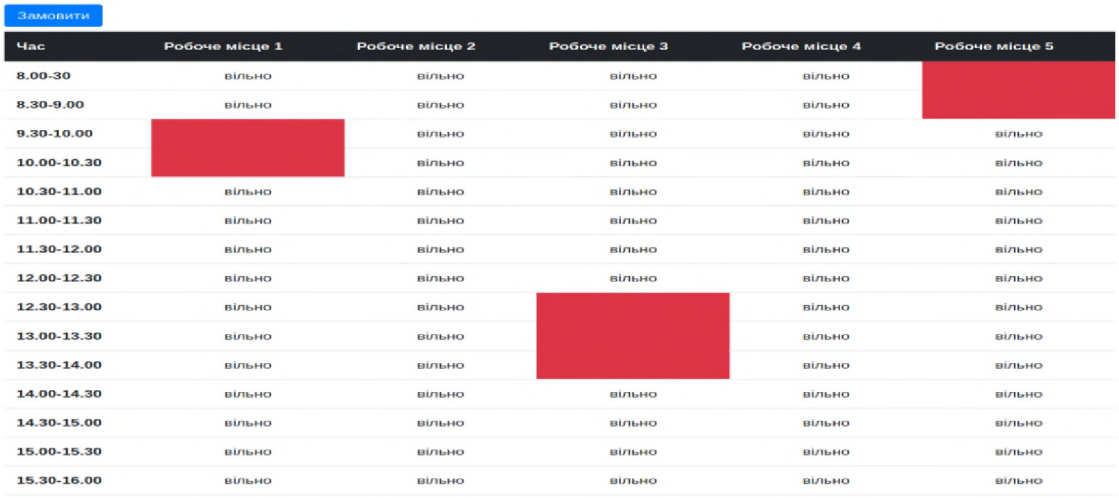


Рисунок 3.11 – Фрагмент сторінки «Замовити»

В цей час якщо м перейдемо до того доступу, який є у адміністратора то побачимо, що так само для того щоб зайти на сторінку йому потрібно зареєструватися і після цього він може бачити всю роботу нашої довідково–інформаційної системи. Де має можливість переглянути замовлення відвідувачів, їх вартість, розташування людей в залі та черговість виконання різних послуг комп'ютерного клубу в даний час.

3.4 Технічні характеристики програмного забезпечення

Для правильної та якісної роботи системи необхідно, щоб комп'ютер відповідав таким характеристикам:

– тактова частота процесора – не менше 1 ГГц;
– об'єм ОП – мінімум 1024 Мбайт;
– об'єм вільного місця на жорсткому диску – мінімум 400 Мбайт;
– мінімальний об'єм відео пам'яті – 512 Мбайт;
– операційна система – Windows (Windows 7, або пізніша версія до Windows 10);
– розширення екрану 1024 x 768;

– наявність CD/DVD – ROM (для встановлення);

– периферія (клавіатура, миша).

Також для запуску на персональному комп'ютері потрібно Framework 4.5 або вище, для забезпечення функціонування Windows додатків, створених на Microsoft Visual Studio. Даний пакет надається із більшістю сучасних версій Windows, та є стандартним, але в окремих випадках потрібно встановлювати та докомплектовувати цю версію.

Код програми подано у додатку Б.

У даному розділі було здійснено програмну реалізацію системи. Подано структуру та функціональне призначення модулів програми, показано реалізацію бази даних, спроектовано інтерфейс користувача. Було здійснено програмну реалізацію системи. Подано структуру та функціональне призначення модулів програми, показано реалізацію бази даних, спроектовано інтерфейс користувача.

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Вибір та обґрунтування методів тестування програмного забезпечення

Після того як довідково–інформаційна система комп'ютерного клубу готова, вона повинна пройти ряд тестів для того щоб зрозуміти чи працює вона коректно. Розглянемо які ж є класифікації та види тестування і якими ми будемо користуватись для перевірки нашого продукту. Перш за все розглянемо класифікацію за рівнем тестування програмного забезпечення

Модульне тестування (Unit Testing). Таке тестування перевіряє функціональність і шукає дефекти в тих частинах додатку, які доступні і можуть бути протестовані окремими частинами. Це можуть бути модулі програм, об'єкти, класи, функції і т.д.

Інтеграційне тестування, як правило, перевіряє взаємодію між компонентами системи після проведення його компонентного тестування.

Системне тестування перевіряє функціональні та не функціональні вимоги в даній системі в цілому. При цьому виявляється перелік дефектів, таких як неправильне використання ресурсів даної системи, непередбачені комбінації даних користувача певного рівня, несумісність з його оточенням, непередбачені сценарії використання, відсутня або неправильна відповідна функціональність та незручність використання.

Операційне тестування полягає в тому, що навіть, якщо система може задовольняти всім вимогам то важливо переконатися в тому, що вона задовольняє всім потребам користувача і виконує свою певну роль в середовищі своєї експлуатації, так як це було визначено в бізнес–моделі даної системи. Слід врахувати, що і бізнес модель може містити деякі помилки. Тому так важливо провести повне операційне тестування як фінальний крок даної валідації. Крім цього, дане тестування в середовищі експлуатації дозволяє виявити і нефункціональні проблеми, наприклад: конфлікт з іншими системами, суміжними в області бізнесу або в інших програмних і електронних

середовищах та недостатня продуктивність їх системи в середовищі експлуатації. Очевидно, що знаходження таких подібних речей на стадії впровадження – це буде критична і дорога проблема. Тому так важливо проведення не тільки верифікації, а й валідації, на початкових етапах розробки програмного забезпечення.

Приймальне тестування це такий формальний процес тестування, який перевіряє відповідність даної системи певним вимогам і проводиться з метою:
визначення чи задовольняє система заявленим критеріям;
винесення рішення замовником або іншою уповноваженою особою чи приймається додаток.

Види тестування за об'єктом тестування:

– функціональне тестування яке розглядає наперед вказану поведінку і ґрунтується на аналізі специфікацій даної функціональності компоненту або системи в цілому;

– тестування продуктивності, підвидами якого є:

– навантажувальне тестування;

– стрес–тестування;

– тестування стабільності.

– тестування безпеки, що використовується для перевірки безпеки системи;

– тестування інтерфейсу користувача;

– тестування зручності використання. Воно направлене на встановлення ступеню зручності використання, навчання, зрозумілості та привабливості для користувача в заданому контексті умов;

– тестування локалізації – це процес тестування локалізованої версії програмного продукту;

– тестування сумісності перевіряє здатність програмного забезпечення працювати в певному середовищі;

За знанням системи існують наступні види тестування:

– тестування чорного ящика – відомі вхідні та вихідні дані, як відбувається перетворення невідомо;

– тестування білого ящика – відомі вхідні та вихідні дані та як відбувається дане перетворення;

– тестування сірого ящика об’єднує у собі два попередні види.

За ступенем автоматизації поділяють тестування на:

– ручне тестування;

– автоматизоване тестування;

– напівавтоматизоване тестування.

За ступенем ізольованості компонентів:

– компонентне тестування полягає в тестуванні окремо кожного модуля системи;

– інтеграційне тестування – це тестування взаємодії скомбінованих в одне ціле всіх частин програми;

– системне тестування тестує дану інтегровану систему.

За часом проведення тестування поділяють:

а. альфа–тестування – імітація реальної роботи з системою штатними розробниками або реальна робота з системою потенційними користувачами/замовником.

1) Розділяється на:
2) тестування при прийманні замовлення або «димове» тестування.
3) тестування нової функціональності.
4) регресивне тестування.
5) тестування при здачі

б. бета–тестування – в деяких випадках виконується поширення версії з обмеженнями для певної групи осіб, для того щоб переконатися, що продукт містить небагато помилок. Іноді бета–тестування виконується для того, щоб отримати відгук про продукт від його майбутніх користувачів.

Можна визначити такі види тестування за ознакою позитивності сценаріїв:

– позитивне тестування та негативне тестування.

За ступенем підготовленості до тестування виділяють:

– тестування по документації;

– тестування без документації, воно базується на методиці передбачення помилки на особистому досвіді тестувальника;

– дослідницьке тестування, це тестування де одночасно виконується дизайн тестів, виконання тестування та навчання.

Дивлячись на розвиток інтернет технологій в розрізі минулих років ми бачимо, що надійність Веб-додатків є необхідністю. Але тестування Веб-додатків має свої певні особливості.

Ми бачимо, що сфера Веб-додатків швидко розвивається. Не всі підходи і методи, які застосовуються для тестування стандартних додатків можуть бути застосовні для тестування Веб-додатків. Веб-додаток – це клієнт–серверний додаток, в якому клієнтом виступає браузер, а сервером – Веб–сервер, а це вже являється по суті двома окремими програмами, які потрібно тестувати як окремо, так і разом взяті. Розглянемо структуру Веб-додатку на рисунку 4.1.

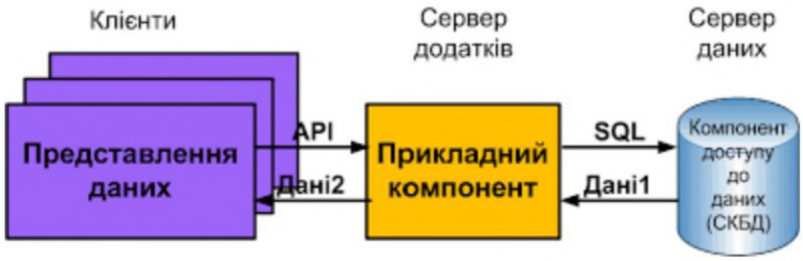


Рисунок 4.1 – Структура Веб-додатку

Більшість сучасних програми орієнтовано на роботу з мережею. Збереження даних Веб-додатків відбувається, переважно, на сервері, обмін інформацією проходить по мережі. Коли користувач виявляє помилку в мережевому середовищі, то часто важко точно вказати, де саме вона відбулася, і тому режим роботи, або попередження про помилку, яке ми отримуємо, може бути результатом тих помилок, які сталися в різних частинах мережевої

системи. Маючи багато спільного з тестуванням класичних програм, тестування Веб–орієнтованих додатків має свої особливості, пов'язані насамперед із середовищем функціонування. Маючи компонентні, структурні та технологічні особливості, Веб-додаткам характерні особливості режимів роботи, інсталяції, запуску, зупинки і видалення, а також створення інтерфейсів. Працюючи як правило з мережею і з великою кількістю користувачів, Веб-додатки розмежовують права доступу для різних користувачів. Логіка Веб-додатку розподілена між сервером і клієнтом, зберігання даних у нього здійснюється на сервері, обмін інформацією проходить по мережі. Одним з переваг підходу може бути той факт, що клієнти не залежать від конкретної операційної системи користувача, це свідчить проте, що Веб-додатки є міжплатформними сервісами. В таблиці 4.1 приведено особливості, відмінності тестування Веб-додатків

Таблиця 4.1 – Відмінності в тестуванні додатків

Відмінності	Класичний додаток	Веб-додаток
1	2	3
Технологічні відмінності	Працює з використанням однієї або сімейства споріднених технологій	Працює з використанням принципово різних технологій
Структурні відмінності	Додаток є «монолітним» та складається з одного або невеликої кількості модулів. Не використовує сервери бази даних та Веб–сервери	«Багатокомпонентний». Він складається з багатьох модулів. Обов'язково використовує сервери баз даних, Веб–сервери та сервери додатків

Продовження таблиці 4.1

1	2	3
Відмінності в режимах роботи	Працює, як правило, в режимі реального часу, а це означає, що відомо про дії користувача відразу ж, як тільки вони виконані	Працює в режимі "запит–відповідь", що говорить про те що відомо про деякий набор дій тільки після запиту на сервері
Відмінність формування інтерфейсу	Використовує для формування інтерфейсу користувача усталені та стандартизовані технології	Використовує для формування інтерфейсу технології, що стрімко розвиваються та безліч яких конкурує між собою
Відмінності роботи з мережею	Практично не використовує мережеві канали передачі даних.	Активно використовує мережеві канали щоб передавати дані.
Відмінності запуску і зупинки	Запускається і зупиняється не часто	Запускається і зупиняється тільки за фактом надходження кожного запиту, а це означає дуже часто
Різниця в кількості користувачів	Кількість користувачів, що одночасно використовують додаток, піддається контролю, вона обмежена і є легко прогнозованою	Кількість користувачів, що одночасно використовують додаток, важко прогнозується і вона може змінюватися в широких діапазонах

Закінчення таблиці 4.1

1	2	3
Особливості збоїв і відмов	Вихід з ладу тих чи інших компонентів додатку відразу стає очевидним	Вихід з ладу деяких компонентів надає непередбачуваний вплив на працездатність всієї програми в цілому
Відмінності інсталяції	Процес інсталяції стандартизований та максимально орієнтований на широку аудиторію користувачів. Він не вимагає різних специфічних знань. Додавання компонентів програми виконується стандартним способом та з використанням одного і того ж інсталятора	Процес інсталяції досить часто недоступний кінцевому користувачеві. Інсталяція вимагає деяких специфічних знань. Процес зміни компонент програми не передбачається або ж потребує специфічної кваліфікації користувачів. Інсталятор відсутній
Відмінності в деінсталяції	Процес деінсталяції стандартизований та виконується автоматично або ж напівавтоматично.	Процес деінсталяції вимагає втручання адміністратора та пов'язаний із зміною коду в середовищі функціонування програми.

4.2 Розробка тестових наборів даних

Для перевірки навантажуальної стійкості програми використовуються спеціальні програми, які допомагають перевірити можливості системи. Одною із них ми і перевірили нашу довідково–інформаційну систему. Після того як перевірка на навантажування пройшла програма видала графік, згідно якого ми бачимо, що вона може працювати з великою кількістю відвідувачів одночасно. Сам графік можна розглянути на рисунку 4.1.

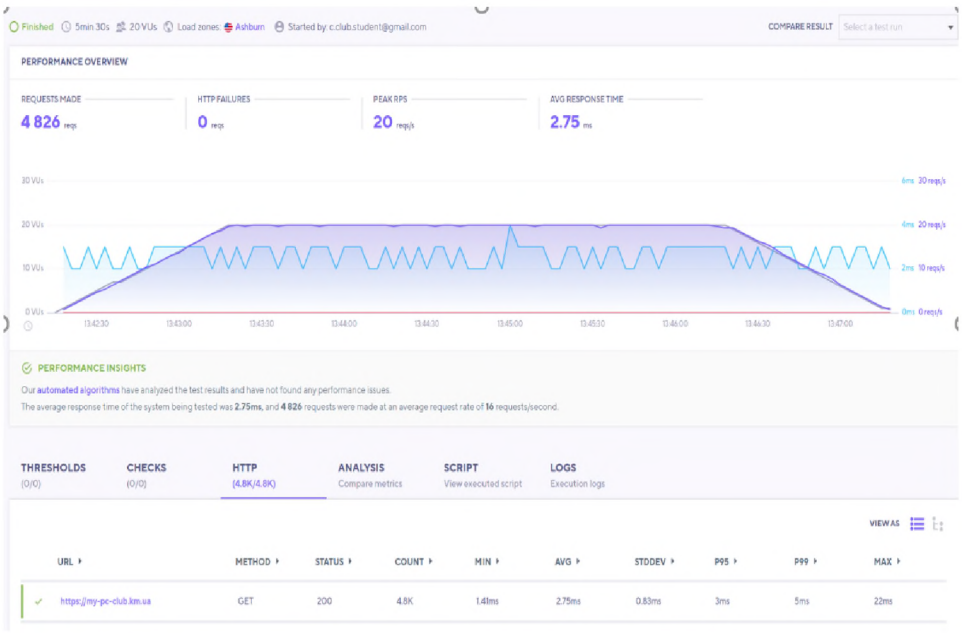


Рисунок 4.1 – Вікно результату навантажувального тесту

На малюнку розміщено графік, де по осі y – кількість одночасних користувачів на сайті, а по осі x – час проходження тесту.

Тестовий сценарій – це послідовність дій по перевірці системи і відповідних результатів на її виході. Дана інформація може подаватися у вигляді таблиць Excel документа, в якому кожен рядок є стан системи, вплив на неї та отриманий результат. Найбільш складним, творчим та неконтрольованим є процес пов’язаний з інтерпретацією вимог, специфікаціями до відповідних тестових послідовностей.

Показниками якості альтернатив на підприємстві визначаються такі критерії як необхідний час для створення програмного продукту для нової технології написання тестових сценаріїв. Цей критерій можна назвати, як ступінь трудовитрат з розробки програмного забезпечення. Процес створення такого тестового сценарію повинен мати чіткі послідовності дій, переходів та умов завершення.

Таблиця 4.2 – Сценарій для тестування інтерфейсу користувача

Дія	Реакція програми	Результат
1. Натиснути на кнопку «Реєстрація»	Появиться діалогове вікно, де потрібно вводити логін та пароль	Вірно
2. У діалоговому вікні ввести ім'я користувача user та пароль 000000. Потім натиснути кнопку «Вхід»	Вікно «Вхід» повинно закритися	*Невірно.* Вікно входу повинно відкритись
3. Завершити сеанс роботи з вкладкою, натиснувши кнопку «Замовити»	Вкладка «Замовити» повинно закритися, і повинно відкритись наступне вікно	Вірно

У даному розділі ілюструється тестування розроблюваного програмного забезпечення та проводиться його аналіз. Довідково-інформаційна система пройшла тестування.

ВИСНОВКИ

Під час виконання курсового проекту ми досягли поставленої мети, а саме було розроблено довідково–інформаційну систему комп'ютерного клубу.

У першому розділі нашої роботи було здійснено дослідження предметної області та виділення функціональних та нефункціональних вимог для розроблюваної довідково–інформаційної системи, та окреслено його технічне завдання та постановку задачі. В цьому розділі був даний аналіз наявноих довідково–інформаційних систем даного типу та виявлено як позитивні так і негативні їх сторони.

У другому розділі було проведено проектування довідково–інформаційної системи, зокрема архітектура, структура, база даних та інтерфейс користувача. Було проведено аналіз та вибрано технології та методи реалізації довідково–інформаційної системи.

У третьому розділі проведена програмна реалізація довідково–інформаційної системи. Було подано структуру та функціональне призначення модулів програми, показано реалізацію баз даних та спроектовано інтерфейс.

У четвертому розділі було проведено тестування створеного програмного засобу в можливому діапазоні.

В останньому розділі роботи було виведено висновки та сформовано список використаної для розробки проекту літератури.

ЛІТЕРАТУРА

1. Jason Gregory. Game Engine Architecture. / A.K. Peters. – Wellesley: Massachusetts, 2009. – 853 с.

2. В. Молочков. WordPress з нуля. / В. Молочков. – Новгород: БХВ, 2020. – 303 с.

3. Дизайн сайту [Електронний ресурс] // Вебsaituz. – Режим доступу до ресурсу: https://Вебsait.uz.ua/dyzajn–saitu/ (дата звернення – 13.04.2021). – Назва з екрану.

4. Изучение OpenGL: VBO, VAO и шейдеры URL [Електронний ресурс] // EAX. – Режим доступу до ресурсу: https://eax.me/opengl-vbo-vao-shaders/ (дата звернення – 10.03.2021). – Назва з екрану.

5. Интернет–маркетинг с нуля [Електронний ресурс] // Tilda. – Режим доступу до ресурсу: https://tilda.education/courses/marketing/social–networks (дата звернення –04.04.2021). – Назва з екрану.

6. Исследование популярности CMS за 2021 год [Електронний ресурс] // ITrack. – Режим доступу до ресурсу: https://itrack.ru/research/cmsrate/ (дата звернення – 30.04.2021). – Назва з екрану.

7. История киберспортивной организации [Електронний ресурс] // Navi. – Режим доступу до ресурсу: https://navi.gg/static/history/ (дата звернення – 05.03.2021). – Назва з екрану.

8. История развития и эволюция видеоигр в цифрах и картинках URL[Електронний ресурс] // Hype.Tech. – Режим доступу до ресурсу: https://hype.tech/@id103/istoriya–razvitiya–i–evolyuciya–videoigr–v–cifrah–i–kartinkahw4e9feon (дата звернення – 04.03.2021). – Назва з екрану.

9. История развития компьютерных игр – от первых игр до виртуальной реальности URL [Електронний ресурс] // Stevsky. – Режим доступу до ресурсу: http://stevsky.ru/starie–igri/istoriya–razvitiya–igr–ot–pervich–igr–dovirtualnoy–realnosti (дата звернення – 04.03.2021). – Назва з екрану.

10. Компютерний клуб [Електронний ресурс] // PlayHub. – Режим доступу до ресурсу: https://playhub–internetcafe.business.site (дата звернення – 25.04.2021). – Назва з екрану.

11. Курс WordPress – с нуля до Профи [Електронний ресурс] // Beonmax. – Режим доступу до ресурсу: https://beonmax.com/courses/wordpress/ (дата звернення – 01.04.2021). – Назва з екрану.

12. Л. П. Бедратюк. Дипломний проект: методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення» / Л. П. Бедратюк, Г. І. Радельчук, Ю. В. Форкун, О. М. Яшина. – Хмельницький: ХНУ, 2020. – 77 с.

13. Мережа кібер хабів [Електронний ресурс] // ItLand. – Режим доступу до ресурсу: https://itland.kiev.ua (дата звернення – 25.04.2021). – Назва з екрану.

14. Полный обзор Unity 5 URL [Електронний ресурс] // Devgam. – Режим доступу до ресурсу: http://devgam.com/polnyj–obzor–unity–5 (дата звернення – 20.03.2021). – Назва з екрану.

15. Что такое киберспорт [Електронний ресурс] // GoSport. – Режим доступу до ресурсу: https://go–sport.ru/article/chto–takoe–cybersport/ (дата звернення – 04.03.2021). – Назва з екрану.

ДОДАТОК А
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках дипломного проекту «Довідково–інформаційна система комп'ютерного клубу».

1 Підстава для розробки

Підставою для розробки є «Завдання на дипломний проект» затверджене завідувачем кафедри інженерії програмного забезпечення.

Найменування розробки: «Довідково–інформаційна система комп'ютерного клубу»

2 Призначення розробки

Довідково–інформаційна система комп'ютерного клубу призначена для узагальнення та отримання інформації по роботі даного закладу та можливість вести облік відвідувачів та їх види діяльності з однієї сторони та надання послуг їм з іншої.

Користувачами програми є адміністратор та відвідувачі клубу.

Адміністратор може контролювати роботу всього закладу. Він може бачити та керувати всіма послугами комп'ютерного клубу, а саме бачити що за обладнання замовив відвідувач, на якому місці він сидить, який час він використовує та одразу бачить загальну вартість послуг. Також він може проводити зміни в замовленнях безпосередньо з свого робочого місця.

Відвідувач може вибирати робоче місце та його тип, бачити графік зайнятості та місцерозташування в комп'ютерному клубі відвідувачів та вільні робочі місця на протязі цілого дня.

3 Вимого до програми

3.1 Вимого до функціональних характеристик

Довідково–інформаційна система комп'ютерного клубу повинна забезпечувати виконання наступних функцій:

– введення, зберігання, пошук і обробку інформації по замовленням
– ведення журналу реєстрації;
– своечасне формування звітів по оплаті послуг;

– формування інформації по наповненості клубу.

У програмі необхідно передбачити: можливість коригування налаштувань системи; резервне збереження даних; можливість зміни пароля входу в систему; наявність вбудованої довідкової системи; швидкий пошук необхідних документів та довідкової інформації тощо;

3.2 Вимоги до надійності

Розроблюване ПЗ повинно мати:

– можливість самовідновлення після збоїв (відключення електроживлення, збої в операційній системі тощо);

– парольний захист при запуску програми;

– обмеження несанкціонованого доступу до даних;

– можливість резервного копіювання інформаційної бази;

– розмежування прав користувача;

– контроль інформації, що вводиться, та блокування некоректних дій користувача при роботі з системою.

3.3 Вимоги до складу та параметрів технічних засобів

Системні вимоги для роботи ПЗ повинні бути наступними:

– тактова частота процесора – 1200 Гц;

– обсяг оперативної пам’яті – 64 Мб;

– обсяг вільного дискового простору – 50 Мб;

– роздільна здатність монітора 1024×768.

3.4 Вимоги до інформаційної та програмної сумісності

Програма повинна працювати в операційних системах Windows XP/7/10. Все звіти, що формуються, повинні мати можливість експортування в редактор електронних таблиць MS Office Excel.

3.5 Вимоги до транспортування та зберігання

Умови експлуатації програмного забезпечення збігаються з умовами експлуатації ПК.

3.6 Спеціальні вимоги

Програма повинна мати дружній інтерфейс, розрахований на користувача середньої кваліфікації (з точки зору комп'ютерної грамотності).

З огляду на обсяг проекту завдання передбачається вирішувати поетапно. При цьому модулі ПЗ, створені в різний час, повинні передбачати можливість нарощування системи і бути сумісні один з одним; тому документація на прийняте експлуатаційне ПЗ повинна містити повну інформацію, необхідну для роботи з ним програмістів.

Мова програмування визначається вибором виконавця, при цьому він повинен забезпечувати можливість інтеграції програмного забезпечення з пакетом MS Office 2003/2007.

4 Вимоги до програмної документації

В ході розробки програми повинні бути підготовлені:

– текст програми;
– опис програми;
– програма;
– методика випробувань;
– керівництво користувача.

5 Техніко–економічне обґрунтування

Для роботи комп'ютерного клубу, який завдяки розвитку кіберспорту переживає своє відродження доцільно мати довідково–інформаційну систему свого клубу з максимально дружнім інтерфейсом до користувача та простотою в роботі. На даний час можна виділити дві категорії людей які можуть відвідувати комп'ютерний клуб. Це молодь яка захоплюється комп'ютерними іграми та старше покоління яке користується переважно соціальними мережами. В той же час пандемія внесла первні корективи у спілкування людей і досить актуально в даний час завдяки довідково–інформаційній системі розмежовувати відвідувачів не тільки в часі але і в місцеположенні у самому клубі. Використання таких обмежень суттєво збільшить відвідуваність людьми та збільшить економічну ефективність

6 Стадії та етапи розробки

Стадії та етапи розробки відповідно створеного графіку.

7 Порядок контролю та приймання

Контроль і приймання розробки здійснюються на основі розробленої методики випробувань. При цьому перевіряється виконання всіх функцій програми. Хід проведення випробувань документується у «Протоколі проведення випробувань». На основі цього протоколу підписується акт приймання–здачі програми в експлуатацію.

ДОДАТОК Б

(обов'язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

```
<?php

class Devices
{
   public function __construct()
   {
      parent::__construct();


      do_action('dashboard.device_controller_init_before');

      if (!current_user_can('access_device|manage_device')) {
         sp_not_permitted();
      }

      breadcrumb_add('dashboard.device', __('П р и с т р і й '), url_for('dashboard.device'));

      view_set('device__active', 'active');


      do_action('dashboard.device_controller_init_after');
   }


   public function index()
   {

      $app = app();

      $clubModel = new ClubModel;

      $currentPage = (int) $app– >request– >get('page', 1);
```

```
$itemsPerPage = 12;

$sort = $app–>request–>get('sort', null);

if (!$clubModel–>isSortAllowed($sort)) {
    $sort = 'newest';
}

$mime = $app–>request–>get('type', null);

$sortRules = $clubModel–>getAllowedSorting();
$mimeRules = $clubModel–>mimeSortType;
array_unshift($mimeRules, 'everything');

$filters = [
    'sort' => e_attr($sort)
];

$filters['where'][] = ['club_type', '=', 'attachment'];


if ($mime && $clubModel–>isValidSortMimeType($mime)) {
    $mimeRegex = $clubModel–>getMimeTypeRegex($mime);
    $filters['where'][] = ['club_mimetype', 'REGEXP', $mime];
} else {
    $mime = 'everything';
}

$totalCount = $clubModel–>countRows(null, $filters);

$queryStr = request_build_query(['page', 'sort', 'type']);

$pagination = new Pagination($totalCount, $currentPage, $itemsPerPage);
$pagination–>setUrl("?page=@id@&sort={$sort}&type={$mime}{$queryStr}");

$paginationHtml = $pagination–>renderHtml();

$offset = $pagination–>offset();
```

```
$entries = $clubModel– >readMany(
    ['*'],
    $offset,
    $itemsPerPage,
    $filters
);

foreach ($entries as $key => $_entry) {
    $entry = $_entry;
    $entry['club_ext'] = pathinfo($_entry['club_path'], PATHINFO_EXTENSION);
    $entry['club_file_type'] = $clubModel– >getFileType($entry['club_ext'], $_entry['club_mimetype']);
    $entry['club_url'] = uploads_uri($_entry['club_path']);
    $entry['club_rel_path'] = trailingslashit(UPLOADS_DIR) . $_entry['club_path'];
    $entry['club_readable_size'] = format_size_units($_entry['club_size']);
    $entry['club_filename'] = $_entry['club_title'] . '.' . $entry['club_ext'];

    if (preg_match('#^image/#i', $_entry['club_mimetype'])) {
        $entry['club_thumbnail'] = sp_thumbnail_uri($entry['club_rel_path']);
    } else {
        $entry['club_thumbnail'] = site_uri("assets/img/media/{$entry['club_file_type']}.png");
    }

    $entries[$key] = $entry;
}

$data = [
    'list_entries'    => $entries,
    'max_upload_size' => format_bytes(get_max_upload_size(), 'M', 0),
    'allowed_filetypes' => $clubModel– >getAllowedFileTypes(),
    'total_items'     => $totalCount,
    'offset'          => $offset === 0 ? 1 : $offset,
    'current_page'    => $currentPage,
    'items_per_page'  => $itemsPerPage,
    'current_items'   => $itemsPerPage * $currentPage,
    'sort_type'       => $sort,
    'mime_type'       => $mime,
    'pagination_html' => $paginationHtml,
    'sorting_rules'   => $sortRules,
    'mime_rules'      => $mimeRules,
    'query_str'       => $queryStr
```

```
    ];
    return view('admin::device/index.php', $data);
}


public function create()
{
    if (!current_user_can('manage_device')) {
        response_status(403);
        return response_body(__("Д о с т у п  з а б о р о н е н о "));
    }

    $uploadPath = uploadspath(date('Y/M/d'));

    if (!is_dir($uploadPath)) {
        mkdir($uploadPath, 0755, true);
    }

    $nonAbsoluteUploadPath = date('Y/M/d');

    $clubModel = new ClubModel;

    $storage = new FileSystem($uploadPath);

    $allowedExtensions = $clubModel– >getAllowedFileTypes();

    $maxSize = format_bytes(get_max_upload_size(), 'M') . 'M';

    $file = new File('file', $storage);

    $originalFileName = $file– >getName();
    $fileName = $originalFileName;

    $i = 1;
    while (file_exists($uploadPath . '/' . $fileName . '.' . $file– >getExtension())) {
        $i++;
        $fileName = $originalFileName . " ({$i})";
    }

    $file– >setName($fileName);
```

```
    $fileData = [
        'name'       => $file->getNameWithExtension(),
        'rawName'    => $file->getName(),
        'extension'  => $file->getExtension(),
        'mime'       => $file->getMimetype(),
        'size'       => $file->getSize()
    ];

    $file->addValidations([
        new Extension($allowedExtensions),
        new Size($maxSize)
    ]);

    try {
        $file->upload();
    } catch (\Exception $e) {
        $errors = $file->getErrors();
        response_status(403);
        return response_body(join($errors, "\n"));
    }

    $filePath = trailingslashit($nonAbsoluteUploadPath) . $fileData['name'];

    $data = [
        'club_title' => $fileData['rawName'],
        'club_size' => $fileData['size'],
        'club_mimetype' => $fileData['mime'],
    ];

    $clubModel->addAttachment($filePath, $data);

    $jsonData = $data;
    $jsonData['success'] = true;
    $jsonData['club_url'] = uploads_uri($filePath);
    $jsonData['club_relative_url'] = leadingslashit(UPLOADS_DIR) . leadingslashit($filePath);

    return json($jsonData);
}
```

```
public function update($id)
{

    breadcrumb_add('dashboard.device.update', __('П о н о в и т и '));

    $clubModel = new ClubModel;

    $device = $clubModel– >read($id);

    if (!$device) {
        flash('device– danger', __('П р и с т р о ї  в  н е  з н а й д е н о '));
        return redirect_to('dashboard.device');
    }

    $data = [];
    return view('admin::device/update.php', $data);
}


public function deletePOST($id)
{
    if (!current_user_can('manage_device')) {
        return sp_not_permitted();
    }

    $clubModel = new ClubModel;

    $filters = [];
    $filters['where'][] = ['club_type', '=', 'attachment'];
    $deviceItem = $clubModel– >read($id, ['club_path'], $filters);

    if (!$deviceItem) {
        flash('device– danger', __('П р и с т р о ї  в  н е  з н а й д е н о '));
        return;
    }

    $clubModel– >deleteAttachment($id);

    flash('device– success', __('П р и с т р і й  в и д а л е н о '));
```

```
    }
}
<?php

class DashboardUsersController
{
    public function __construct()
    {
        parent::__construct();

        do_action('dashboard.users_controller_init_before');
        breadcrumb_add('dashboard.users', __('К о р и с т у в а ч і '), url_for('dashboard.users'));
        view_set('users__active', 'active');
        do_action('dashboard.users_controller_init_after');
    }

    public function index()
    {
        if (!current_user_can('add_user|edit_user|delete_user')) {
            sp_not_permitted();
        }
        $app = app();

        $userModel = new UserModel;

        $roleModel = new RoleModel;

        $usersTable = $userModel- >getTable();
        $rolesTable = RoleModel::getTable();

        $currentPage = (int) $app- >request- >get('page', 1);

        $search = $app- >request- >get('s', null);

        $itemsPerPage = (int) $app- >config('dashboard.items_per_page');

        $sort = $app- >request- >get('sort', null);

        if (!$userModel- >isSortAllowed($sort)) {
            $sort = 'oldest';
```

```
}
$sortRules = $userModel->getAllowedSorting();


$filters = [
    'sort' => e_attr($sort)
];

$roleID = (int) $app->request->get('role_id', 0);

$role = [
    'role_id'     => $roleID,
    'role_name'   => null,
    'users_count' => 0
];

if ($roleID) {

    $roleQuery = $roleModel->read($roleID, ['role_name']);

    if (!$roleQuery) {

        $roleID = 0;
    } else {

        $role['role_name'] = $roleQuery['role_name'];

        $filters['where'][] = ["{$usersTable}.role_id", '=', $roleID];
    }
}

if ($filter) {
    $filters['where'][] = ["{$usersTable}.full_name", 'LIKE', "%$search%"];
    $filters['where'][] = ["{$usersTable}.email", 'LIKE', "%$search%", 'OR'];
    $filters['where'][] = ["{$usersTable}.usemame", 'LIKE', "%$search%", 'OR'];
    $filters['where'][] = ["{$usersTable}.user_ip", 'LIKE', "%$search%", 'OR'];
}

$totalCount = $userModel->countRows(null, $filters);
```

```
$role['users_count'] = $totalCount;
$queryStr = request_build_query(['page', 'sort']);

$pagination = new Pagination($totalCount, $currentPage, $itemsPerPage);
$pagination–>setUrl("?page=@id@&sort={$sort}{$queryStr}");

$paginationHtml = $pagination–>renderHtml();

$offset = $pagination–>offset();


$fields[] = "{$usersTable}.*";
$fields[] = "{$rolesTable}.role_name";

$sql = $userModel–>select($fields)
–>leftJoin(
    $rolesTable,
    "{$usersTable}.role_id",
    '=',
    "{$rolesTable}.role_id"
);

$sql = $sql–>limit($itemsPerPage, $offset);

$sql = $userModel–>applyModelFilters($sql, $filters);
$stmt = $sql–>execute();

$entries = $stmt–>fetchAll();

$roleList = [];

$roleList[] = [
    'role_id' => 0,
    'users_count' => $userModel–>countRows(),
    'role_name' => __('All'),
];

$protectedRoles = $roleModel
            –>select(['role_id', 'role_name'])
```

```
            – >where('is_protected', '=', 1)
            – >execute()
            – >fetchAll();

    $currentRoleIsProtected = false;
    foreach ($protectedRoles as $key => $_role) {

        $_role['users_count'] = $roleModel– >getUsersCountUnderRole($_role['role_id']);
        $roleList[] = $_role;

        if ($roleID === $_role['role_id']) {
            $currentRoleIsProtected = true;
        }
    }

    if (!$currentRoleIsProtected && $roleID) {
        $roleList[] = $role;
    }

    $data = [
        'page_subheading' => __('К о р и с т у в а ч і '),
        'list_entries'    => $entries,
        'total_items'     => $totalCount,
        'offset'          => $offset === 0 ? 1 : $offset,
        'current_page'    => $currentPage,
        'items_per_page'  => $itemsPerPage,
        'current_items'   => $itemsPerPage * $currentPage,
        'sort_type'       => $sort,
        'pagination_html' => $paginationHtml,
        'sorting_rules'   => $sortRules,
        'query_str'       => $queryStr,
        'role'            => $role,
        'search'          => $search,
        'role_list'       => $roleList
    ];
    return view('admin::users/index.php', $data);
}
public function getList()
{
    if (!current_user_can('add_user|edit_user|delete_user')) {
```

```
    sp_not_permitted();
}
$app = app();
$req = $app->request;
$action = $req->post('action');
$postedIDs = (array) $req->post('item_multi', []);
$userIDs = [];
foreach ($postedIDs as $value) {
    if ((int) $value === current_user_ID()) {
        continue;
    }
    $userIDs[] = $value;
}
if (empty($userIDs)) {
    flash('users_warning', __('Не вибрано'));
    return redirect_to_current_route();
}
$userModel = new UserModel;
switch ($action) {
    case 'delete':
        if (!current_user_can('delete_user')) {
            sp_not_permitted();
        }
        $i = 0;
        foreach ($userIDs as $id) {
            $userModel->deleteUser($id);
            $i++;
        }
        flash('users_success', sprintf(__('%s був видалений'), $i));
        return redirect_to_current_route();
        break;
    case 'verify':
        if (!current_user_can('edit_user')) {
            sp_not_permitted();
        }
        $i = 0;
        foreach ($userIDs as $id) {
            $userModel->update($id, ['is_verified' => 1]);
            $i++;
        }
```

```
            flash('users- success', sprintf(__('%d %s б у в  д о д а н и й '), $i));
            return redirect_to_current_route();
        case 'unverify':
            if (!current_user_can('edit_user')) {
                sp_not_permitted();
            }
            $i = 0;
            foreach ($userIDs as $id) {
                $userModel- >update($id, ['is_verified' => 0]);
                $i++;
            }
            flash('users- success', sprintf(), $i));
            return redirect_to_current_route();
        case 'block':
            if (!current_user_can('change_user_status')) {
                sp_not_permitted();
            }
            $i = 0;
            foreach ($userIDs as $id) {
                $userModel- >update($id, ['is_blocked' => 1]);
                $i++;
            }
            flash('users- success', sprintf(__('%d'), $i));
            return redirect_to_current_route();
        case 'unblock':
            if (!current_user_can('change_user_status')) {
                sp_not_permitted();
            }
            $i = 0;
            foreach ($userIDs as $id) {
                $userModel- >update($id, ['is_blocked' => 0]);
                $i++;
            }
            flash('users- success', sprintf(__('%d'), $i));
            return redirect_to_current_route();
        default:
            flash('users- warning', __('П о м и л к а , с п р о б у й т е  п і  з н і  ш е '));
            return redirect_to_current_route();
    }
}
```

```
public function create()
{
    if (!current_user_can('add_user')) {
        sp_not_permitted();
    }



    breadcrumb_add('dashboard.users.create', __('Д о д а т и '));
    $roles = [];
    if (current_user_can('change_user_role')) {
        $roleModel = new RoleModel;
        $roles = $roleModel–  >readMany(['role_id', 'role_name'], 0, 50);
    }
    $data = [
        'role_list' => $roles
    ];
    return view('admin::users/create.php', $data);
}

public function createUser()
{
    if (!current_user_can('add_user')) {
        sp_not_permitted();
    }

    $app = app();
    $req = $app–  >request;
    $data = [
        'email'     => $req–  >post('email'),
        'password'  => $req–  >post('password'),
        'username'  => $req–  >post('username'),
        'role_id'   => $req–  >post('role_id'),
        'full_name' => $req–  >post('full_name'),
        'user_ip'   => $req–  >post('user_ip'),
    ];
    $userModel = new UserModel;
    $roleModel = new RoleModel;
    $v = new Validator($data);
```

```
    $v– >labels([
        'email'     => __('Електронна пошта'),
        'password'  => __('Пароль'),
        'username'  => __('Користувач'),
        'full_name' => __('ПІБ'),
    ])– >rule('required', ['email', 'password'])
      – >rule('email', 'email')
      – >rule('uniqueEmail', 'email')
      – >rule('uniqueUsername', 'username')
      – >rule('lengthMin', 'password', (int) config('internal.password_minlength'))
      – >rule('lengthMax', 'full_name', 200)
      – >rule('ip', 'user_ip')
      – >rule('username', 'username');
    if (!$v– >validate()) {
        $errors = sp_valitron_errors($v– >errors());
        flash('users– danger', $errors);
        sp_store_post($data);
        return redirect_to_current_route();
    }
    if (!current_user_can('change_user_role') || !$roleModel– >exists($data['role_id'])) {
        $data['role_id'] = RoleModel::TYPE_USER;
    }
    $data['full_name'] = sp_strip_tags($data['full_name'], true);
    $data['is_verified'] = 1;
    if (!$id = $userModel– >addUser(
        $data['email'],
        $data['password'],
        $data
    )) {
        flash('users– danger', UserModel::DB_ERROR);
        return redirect_to_current_route();
    }
    flash('users– success', __("Користувача додано"));
    return redirect_to('dashboard.users');
}

public function update($id)
{
    if (!current_user_can('edit_user')) {
        sp_not_permitted();
```

```
    }

    breadcrumb_add('dashboard.users.update', __('Update User'));
    $userModel = new UserModel;
    $user = $userModel->read($id);
    if (!$user) {
        flash('users-danger', __('К о р и с т у в а ч і  в  н е  з н а й д е н о '));
        return redirect_to('dashboard.users');
    }
    if ((int) $user['user_id'] === current_user_ID()) {
        flash('users-danger', __('Р е д а г у в а н н я  з а б о р о н е н о '));
        return redirect_to('dashboard.users');
    }
    $roleModel = new RoleModel;
    $roleName = $roleModel->read($user['role_id'], ['role_name'])['role_name'];
    $user['role_name'] = $roleName;
    $roles = [];
    if (current_user_can('change_user_role')) {
        $roleModel = new RoleModel;
        $roles = $roleModel->readMany(['role_id', 'role_name'], 0, 50);
    }
    $data = [
        'role_list' => $roles,
        'user' => $user
    ];
    return view('admin::users/update.php', $data);
}

public function updateUser($id)
{
    if (!current_user_can('edit_user')) {
        sp_not_permitted();
    }

    $userModel = new UserModel;
    $roleModel = new RoleModel;
    $app = app();
    $req = $app->request;
    $user = $userModel->read($id, ['user_id', 'username', 'email', 'role_id']);
    if (!$user) {
```

```
    flash('users_ danger', __('К о р и с т у в а ч і  в  н е  з н а й д е н о '));
    return redirect_to('dashboard.users');
}

if ((int) $user['user_id'] === current_user_ID()) {
    flash('users_ danger', __('Р е д а г у в а н н я  з а б о р о н е н о '));
    return redirect_to('dashboard.users');
}
$data = [
    'email'     => $req_ >post('email'),
    'password'  => $req_ >post('password'),
    'username'  => $req_ >post('username'),
    'role_id'   => $req_ >post('role_id', 0),
    'full_name' => $req_ >post('full_name'),
    'user_ip'   => $req_ >post('user_ip'),
    'is_blocked'   => sp_int_bool($req_ >post('is_blocked')),
    'is_verified'   => sp_int_bool($req_ >post('is_verified')),
];
$v = new Validator($data);
$v_ >labels([
    'email'     => __('Е л е к т р о н н а  п о ш т а '),
    'password'  => __('П а р о л ь '),
    'username'  => __('К о р и с т у в а ч '),
    'full_name' => __('П І  Б '),
])_ >rule('required', ['email', 'user_ip'])
  _ >rule('email', 'email')
  _ >rule('uniqueEmail', 'email', $user['email'])
  _ >rule('uniqueUsername', 'username', $user['username'])
  _ >rule('lengthMin', 'password', (int) config('internal.password_minlength'))
  _ >rule('lengthMax', 'full_name', 200)
  _ >rule('ip', 'user_ip')
  _ >rule('username', 'username');
if (!$v_ >validate()) {
    $errors = sp_valitron_errors($v_ >errors());
    flash('users_ danger', $errors);
    sp_store_post($data);
    return redirect_to_current_route();
}

if (current_user_can('change_user_role')) {
```

```
        if (!$roleModel->exists($data['role_id'])) {
            unset($data['role_id']);
        }
    } else {
        unset($data['role_id']);
    }
    if (!current_user_can('change_user_status')) {
        unset($data['is_blocked']);
    }
    if (empty($data['password'])) {
        unset($data['password']);
    }
    $data['full_name'] = sp_strip_tags($data['full_name'], true);
    $userModel->updateUser($id, $data);
    flash('users-success', __('Дані користувача поновлено'));
    return redirect_to_current_route();
}

public function delete($id)
{
    if (!current_user_can('delete_user')) {
        sp_not_permitted();
    }

    sp_enqueue_script('parsley', 2, ['dashboard-core-js']);

    breadcrumb_add('dashboard.users.delete', __('Видалити користувача'));
    $userModel = new UserModel;
    $user = $userModel->read($id, ['full_name', 'email', 'avatar', 'last_seen', 'user_id', 'created_at']);
    if (!$user) {
        flash('users-danger', __('Користувача не знайдено'));
        return redirect_to('dashboard.users');
    }
    if ((int) $user['user_id'] === current_user_ID()) {
        flash('users-danger', __("Видалення заборонено"));
        if (is_ajax()) {
            return;
        }
        return redirect_to('dashboard.users');
    }
```

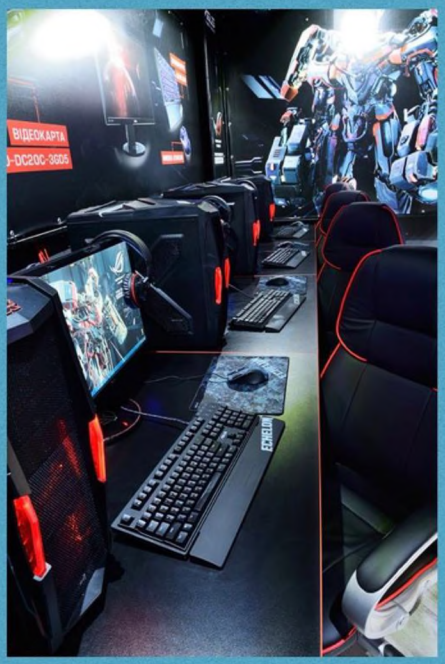
```
        $data = [
            'user' => $user
        ];
        return view('admin::users/delete.php', $data);
    }

    public function deleteUser($id)
    {
        if (!current_user_can('delete_user')) {
            sp_not_permitted();
        }

        $userModel = new UserModel;
        $user = $userModel->read($id, ['user_id']);
        if (!$user) {
            flash('users-danger', __('Видалити користувача'));
            if (is_ajax()) {
                return;
            }
            return redirect_to('dashboard.users');
        }
        if ((int) $user['user_id'] === current_user_ID()) {
            flash('users-danger', __("Ви не можете видалити власний аккаунт"));
            if (is_ajax()) {
                return;
            }
            return redirect_to('dashboard.users');
        }
        $userModel->deleteUser($id);
        flash('users-success', __('Користувача видалено'));
        if (is_ajax()) {
            return;
        }
        return redirect_to('dashboard.users');
```

ДОДАТОК В

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Довідково-інформаційна система комп’ютерного клубу



До недавнього часу захоплення комп'ютерними іграми викликало лише незадоволення батьків, а вже сьогодні дохід професійних кіберспортсменів сягає мільйонів доларів.

Останні десять років ознаменовані появою нового спортивного спрямування, в якому українці займають найвищі місця у світових змаганнях.

На підйомі популярності кіберспорту розвивається і напрямок комп’ютерних клубів. Для них це основний вектор розвитку. Ця тенденція спостерігається у всьому світі.

Зважаючи на все вище викладене, можна зробити висновок про доцільність та актуальність створення довідково-інформаційної системи комп'ютерного клубу, яка повинна забезпечити злагоджену та ефективну взаємодію відвідувачів та власників клубу.

Мета проекту – розробка довідково-інформаційної системи комп'ютерного клубу.

Об'єкт дослідження – процес організації роботи та взаємодії відвідувачів та адміністратора клубу за допомогою довідково-інформаційної системи комп'ютерного клубу.

Предмет – система створення довідково-інформаційної системи комп'ютерного клубу.

Що ж повинно бути в нашій довідково-інформаційній системі?

- Перш за все вона повинна бути розбита на два інтерфейси: для відвідувачів та адміністратора клубу.
- При входженні в систему відвідувач повинен зареєструватися та створити свій обліковий запис.
- В інтерфейсі користувачі можуть поповнювати свій баланс, вибирати тарифи і абонементи, переглядати необхідні сайти і запускати різноманітні ігри, перевіряти новини вашого клубу.

Для управління комп'ютерним клубом довідково-інформаційна система має включати в себе наступні можливості:

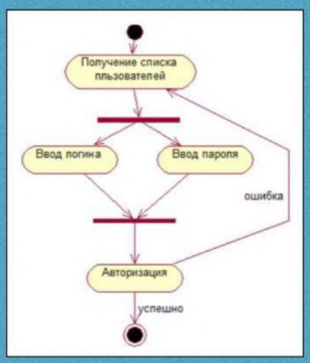
- інтерфейс для клієнта і адміністратора,
- карту управління ПК,
- модуль резервування ПК,
- цінові пропозиції та програму лояльності.

З свого боку, адміністратори клубу можуть керувати всіма комп'ютерами і їх резервами, розміщувати новини, оновлювати список доступних ігор, відстежувати кількість доступних ПК, обробляти замовлення клієнтів, а також вивантажувати звіти.

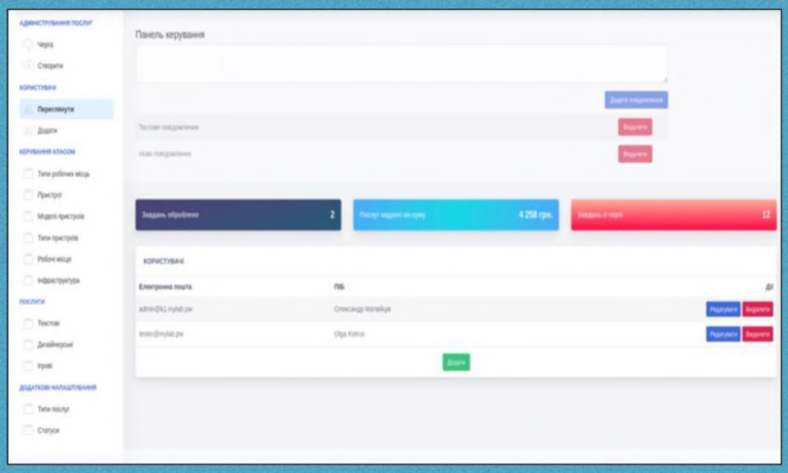
Варто також врахувати, що починаючи з 2020 року під час пандемії COVID-19 комп'ютерні клуби повинні дотримуватися встановлених епідеміологічних правил, а програмне забезпечення повинно бути достатньо гнучким, щоб допомогти бізнесу адаптуватися до нових норм.

Для довідково-інформаційної системи комп'ютерний клуб було створено базу даних

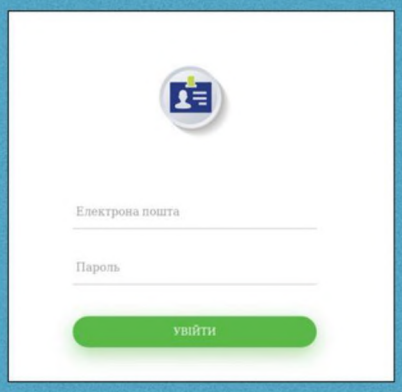
Для кращої роботи довідково-інформаційної системи комп'ютерного клубу бажаним є забезпечення можливості реєстрації та авторизації користувачів. На рис. зображено вигляд авторизації користувача схематично за допомогою UML діаграми діяльності.



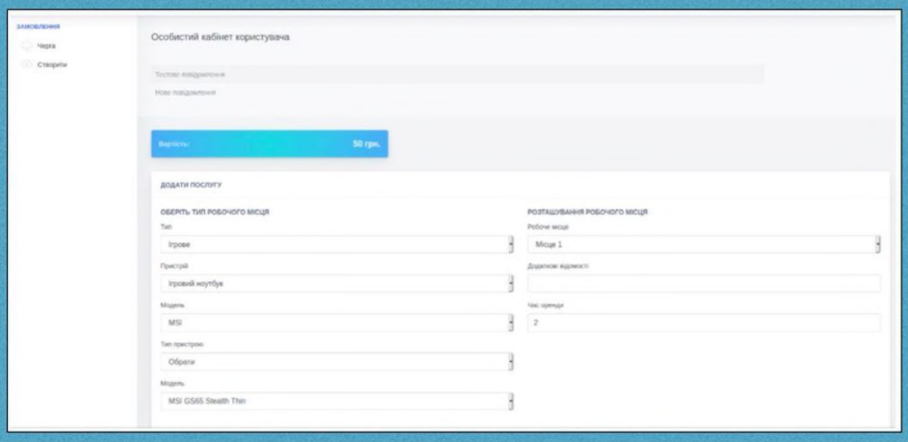
В доступі для адміністратора відкривається вікно з такими можливостями як адміністрування, користувачі, керування класом, послуги, додаткові налаштування. Видно завдання до обробки, рахунок за послуги, кількість завдань в черзі



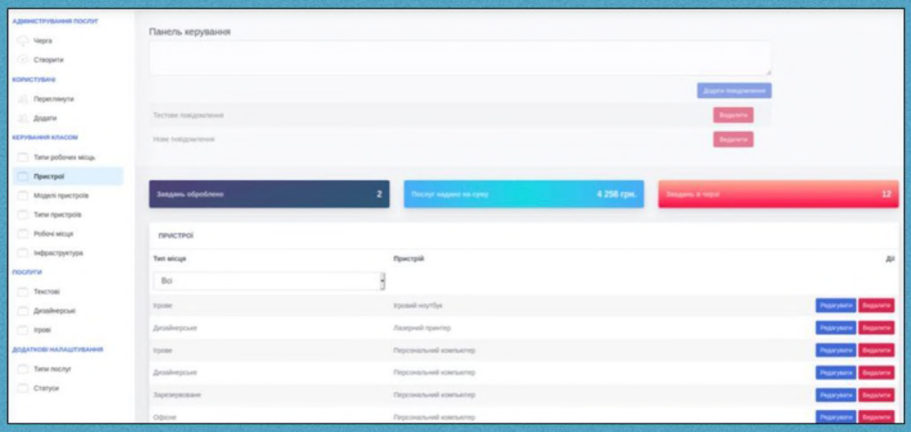
В довідково інформаційній системі проводиться авторизація відвідувачів.



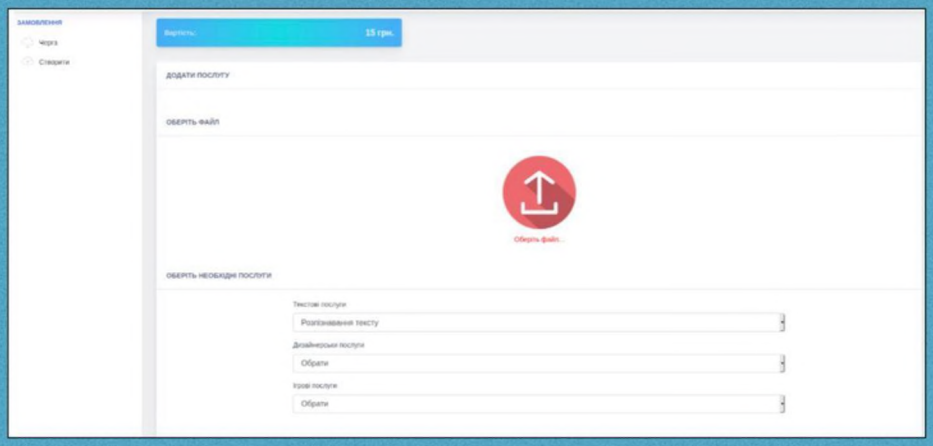
Якщо розглянути вікна які доступні користувачеві при користу-ванні довідково-інформаційною системою, то особистий кабінет має такий вигляд.



У цьому вікні можна побачити задіяні пристрої та їх призначення



Користувач одразу може обрати послугу, побачити її вартість, завантажити необхідний файл



Дякую за увагу

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Зайцева І. О.

Прізвище, ініціали

факультет ПКТС, 4 курс, група ІПЗ-17-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02 червня 2021

дата

підпис

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1008255602

Дата перевірки:
10.06.2021 11:51:55 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
10.06.2021 11:52:35 EEST

ID користувача:
100005589

Назва документа: Записка Диплом Зайцев без додатків

Кількість сторінок: 64 Кількість слів: 11334 Кількість символів: 83302 Розмір файлу: 5.78 MB ID файлу: 1008327152

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

14.7%
Схожість

Найбільша схожість: 7% з Інтернет-джерелом (http://www.it-simple.ru/?p=2322)

12.6% Джерела з Інтернету | 199 ..... Сторінка 66

2.37% Джерела з Бібліотеки | 42 ..... Сторінка 67

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 9

Підозріле форматування | 12 сторінок

Wed Jun 09 20:13:36 EEST 2021, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 11.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 16%**

ID: 92960 Назва: Довідково-інформаційна система комп`ютерного клубу Додано в БД: 2021-06-09 Автора: І. О. Зайцев Керівники: О. Г. Онишко Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	86217	860	17867 (21%)	198 (23%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
90212	Назва: Звіт з переддипломної практики Зайцев І.О. Додано в БД: 2021-05-11 Автора: Зайцев І.О. Керівники: Онишко О.Г Консультанти: Опоненти:	9154 (11.0%)	81 (9.0%)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Довідково-інформаційна система комп'ютерного клубу»

Автор: Зайцев Ігор Олександрович

Спеціальність: 121 – Інженерія програмного забезпечення

Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»

Науковий керівник: Онишко Оксана Григорівна, канд. пед. наук, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломного проекту системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень в бланках (титульного аркушу, бланку завдання, в структурі підрозділів, ВСТУПУ);
2) В якості запозичень системою було зафіксовано стандартні конструкції та послідовності коду, а також посилання на використання бібліотек які є спільними для великої кількості мобільних додатків та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;
3) усі запозичення фрагментарні, або мають належним чином оформленні посилання.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності / схожості, складає 14,7% і адресується до 241 першоджерел, що з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь дипломного проекту.

Керівник _____ О.Г. Онишко

Гарант ОП _____ Л.П. Бедратюк

Завідувач кафедри _____ Л.П. Бедратюк

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНИЙ ПРОЕКТ
освітнього ступеня «Бакалавр»

Дипломник Зайцев Ігор Олександрович

Тема Довідково-інформаційна система комп`ютерного клубу

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломного проекту:

Кількість листів креслень _____; кількість сторінок записки 88

1.Короткий зміст пояснювальної записки та прийнятих рішень У дипломному проекті проаналізовано предметну область, обрано інструменти для вирішення поставленого питання, а також визначено недоліки існуючих на ринку рішень. Була створена система комп`ютерного клубу, проведено її тестування і сформовано висновки.

2. Висновок про відповідність проекту поставленому завданню Дипломна робота повністю відповідає поставленому завданню та виконана із дотриманням всіх вимог.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовується актуальність теми роботи, дається аналіз досліджуваної проблеми та обґрунтовується застосовуваний підхід до її вирішення, формулюються цілі і завдання дослідження, описується практична значимість отриманих результатів. У першому розділі проведено змістовний аналіз предметної області та дослідження цільової аудиторії людей, для яких реалізується інформаційна система. Наступні розділи присвячені проектуванню довідково-інформаційної системи, розробці, а також тестуванню програмного забезпечення.

4. Позитивні сторони проекту Тематика дипломної роботи є актуальною, визначено існуючі проблеми та незручності, завдяки чому було модернізовано й усучаснено підхід до роботи з клієнтами комп'ютерного клубу, застосувавши при цьому існуючі необхідні технології.

5. Негативні сторони проекту Система розроблена таким чином, що користувачі зможуть дізнаватись інформацію про новини та список оновлених ігор клубу лише після обов'язкової ідентифікації, способом введеня логіну і паролю.

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконане відповідно до теми дипломної роботи з дотриманням вимог стандартів, виконане на достатньому рівні. Пояснювальна записка відповідає вимогам стандартів до її оформлення.

7. Відгук про дипломний проект в цілому В цілому дипломна робота заслуговує позитивної оцінки. Весь матеріал дипломної роботи структурований, чіткий та послідовний. Усі розділи роботи є послідовними та логічними, що дозволяє чітко розуміти викладений матеріал у рамках тематики дипломної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для вирішення поставленої задачі.

8. Інші зауваження

9. Оцінка дипломного проекту Робота виконана в повному обсязі. Розглянувши позитивні та негативні сторони представленої дипломної роботи, можна зробити висновок, що вона заслуговує на оцінку «добре» (4,00/С).

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи) Лисенко Сергій Миколайович, доктор технічних наук, доцент кафедри комп'ютерної інженерії та системного програмування (КІСП) ХНУ.

"01" червня 2021 р. _____
(підпис)