НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

на тему: «Web-система з організації благодійних аукціонів»

Виконала:
студентка IV курсу, групи ТВ-71
Вахромова Надія Олександрівна _____

Керівник:
к.т.н., доц.
Шалденко Олексій Вікторович _____

Рецензент:

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2021р.

ЗАВДАННЯ
на дипломну роботу студенту

Вахромовій Надії Олександрівні
(прізвище, ім’я, по батькові)

1. Тема роботи Web-система з організації благодійних аукціонів

керівник роботи к.т.н., доц. Шалденко Олексій Вікторович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”_24_” травня 2021р. № 1267-с_

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи __Мова програмування PHP, фреймворк Laravel, веб-система для проведення благодійних аукціонів

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Провести аналіз благодійної діяльності в Україні та наявних систем, що надають можливість долучитись до благодійності. Спроектувати систему проведення благодійних аукціонів. Підібрати стек технологій. Розробити візуальну частину веб-застосунку. Реалізувати потрібний вигляд системи та відповідний функціонал. Протестувати програмний продукт.

5. Перелік ілюстративного матеріалу

Існуючі способи надання благодійної допомоги, модель MVC, інтерфейс Figma та PhpStorm, опис стторінок веб-системи, діаграма прецедентів, діаграма класів, схема організації БД, приклади програмного коду, сторінки веб-системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”10” жовтня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	Захист програмного продукту	13.05.2021	
8.	Передзахист	27.05.2021	
9.	Захист	15.06.2021	

Студент _____ _____Вахромова Н.О._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Шалденко О.В._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Дипломну роботу виконано на 49 аркушах, вона містить 3 додатки та перелік посилань на використані джерела з 11 найменувань. У роботі наведено 37 рисунків.

Метою даної дипломної роботи є розробка системи для проведення благодійних аукціонів, яка є інтуїтивно зрозумілою та зручною у використанні, щоб залучити більше людей до благодійної діяльності.

У роботі проведено аналіз існуючих способів надання благодійної допомоги, виділені їх переваги та недоліки. Для вирішення недоліків було розроблено систему проведення аукціону для залучення більшої кількості людей.

Спроектовано та розроблено веб-систему для організації благодійних аукціонів із зрозумілим інтерфейсом та відповідним функціоналом. Здійснено мануальне тестування, що допомогло виявити певні недоліки в системі і вчасно їх виправити.

Ключові слова: благодійність, благодійна діяльність, благодійні аукціони, адміністративна панель, панель прогресу.

ABSTRACT

Thesis is done on 49 sheets, it contains 3 appendices and a list of references to the sources used from 11 items. The work shows 37 figures.

The aim of this thesis is to develop a system for charity auctions, which is intuitive and easy to use to attract more people to charity.

The paper analyzes the existing ways of providing charitable assistance, highlights their advantages and disadvantages. To address the shortcomings, an auction system has been developed to attract more people.

A web system for organizing charity auctions with a clear interface and appropriate functionality has been designed and developed. Manual testing was performed, which helped to identify certain shortcomings in the system and correct them in time.

Key words: charity, charitable activity, charity auctions, administrative panel, progress panel.

ЗМІСТ

АНОТАЦІЯ 4
ABSTRACT 5
ЗМІСТ 6
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8
ВСТУП 9
1. ЗАДАЧА РОЗРОБКИ ВЕБ-СИСТЕМИ З ОРГАНІЗАЦІЇ БЛАГОДІЙНИХ АУКЦІОНІВ 11
2. АНАЛІЗ ПРОБЛЕМИ ОРГАНІЗАЦІЇ БЛАГОДІЙНОЇ ДІЯЛЬНОСТІ В УКРАЇНІ 13
2.1. Аналіз існуючих способів надання благодійної допомоги 13
2.2. Необхідність створення веб-системи з проведення благодійних аукціонів 15
3. МЕТОДИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ 17
3.1. Вибір архітектурного рішення 17
3.2. Опис інструментів розробки 18
3.2.1. Сервіс розробки інтерфейсів Figma 18
3.2.2. Середовище розробки JetBrains PhpStorm 19
3.2.3. Мова розмітки HTML 20
3.2.4. Таблиці стилів CSS 21
3.2.5. Фреймворк Bootstrap 22
3.2.6. Фреймворк Laravel 23
3.2.7. Система управління MySQL 24
4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 25
4.1. Загальна інформація 25
4.2. Опис сторінок 25
4.3. Діаграма прецедентів 26
4.4. Діаграма класів 27
4.5. Робота з базою даних 28
4.6. Маршрутизація 31
4.7. Контролер AuctionController 32
4.8. Допоміжні функції 34

4.9. Контролер панелі адміністратора LotController 35
5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ 38
5.1. Системні вимоги 38
5.2. Сценарій роботи користувача з системою 38
5.3. Сценарій роботи адміністратора з системою 42
ВИСНОВКИ 48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 49
ДОДАТОК А 50
ДОДАТОК Б 52
ДОДАТОК В 72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних.

HTTP – HyperText Transfer Protocol.

HTML – Hyper Text Markup Language.

CSS – Cascading Style Sheets.

ВСТУП

Благодійність є добровільною сферою діяльності, що направлена на допомогу іншим особам, державі, підприємствам тощо. Особами, що надають добровільну допомогу можуть бути окремі фізичні особи або компанії. Враховується будь-яка допомога, така як, наприклад, допомога послугами, матеріальна допомога, фінансова або надання компанією деяких потрібних товарів.

На щастя, благодійність в Україні останнім часом набирає обертів. Більше людей хочуть стати причасними до допомоги іншим. Згідно з думкою експертів, благодійна діяльність у нашій країні почала стрімко розвиватись через події 2014 року. Люди зрозуміли, що треба більше приділяти уваги людям навколо й допомагати, адже так ми зможемо домогтися змін.

Проте, згідно з дослідженнями, проведеними нещодавно, експертна оцінка благодійної діяльності в Україні була досить суперечливою й становила лише 5,5 балів із десяти можливих. Це пов’язано з тим, що часто люди, які мають намір зайнятись благодійністю, просто не мають необхідної інформації про те, як це правильно зробити.

В даний момент часу існує проблема недостатньої систематизації благодійної роботи, тобто немає достатньої кількості знань або ресурсів задля автоматизації та структуризації благодійної діяльності. Багато ресурсів, які використовують тільки переказ на рахунок коштів, що не створює у людей відчуття того, що вони беруть участь у благодійності.

Саме тому актуальною задачею є створення веб-застосунку, що дозволить людям приймати участь у благодійних аукціонах, адже це дуже розвинена практика у всьому світі, яка зарекомендувала себе як досить дієвий спосіб долучити громадян до допомоги тим, хто цього потребує.

Веб-застосунок з організації благодійних аукціонів доцільно розробити у вигляді веб-сайту, використавши мову програмування PHP та середовище розробки PhpStorm.

Пояснювальна записка складається з п’яти розділів.

У першому розділі розглянуто питання постановки задачі розробки веб-системи з проведення благодійних аукціонів, виділено основні задачі.

У другому розділі наведено аналіз проблеми благодійності в Україні та уже існуючих способів долучитись до благодійної роботи.

Третій розділ розглядає питання використаної архітектури та обраних для реалізації технологій.

Четвертий розділ присвячено опису програмної реалізації, де наведено відповідні схеми та діаграми.

П’ятий розділ пояснює те, як користувач може взаємодіяти з веб-системою з супроводженням рисунками. Наведено роботу і просто користувача, і користувача з правами адміністратора.

1. ЗАДАЧА РОЗРОБКИ ВЕБ-СИСТЕМИ З ОРГАНІЗАЦІЇ БЛАГОДІЙНИХ АУКЦІОНІВ

Метою дипломної роботи є розробка зручної й інтуїтивно-зрозумілої веб-системи, що є легкою у використанні, щоб залучити більшу кількість людей до благодійності. Система має надати можливість проводити благодійні аукціони з ціллю збору коштів для допомоги дітям.

Щоб досягнути поставленої мети потрібно:

— Провести аналіз благодійної діяльності в Україні та наявних систем, що надають можливість долучитись до благодійності. Виявити у присутніх способах переваги та недоліки.

— Спроектувати систему проведення благодійних аукціонів з урахуванням недоліків, знайдених в уже існуючих системах.

— Підібрати необхідний для реалізації стек технологій.

— Розробити візуальну частину веб-застосунку.

— Реалізувати потрібний вигляд системи та її функціонал, тобто Font-end Back-end частини системи, за допомогою обраних раніше технологій.

Веб-система повинна мати наступні основні можливості:

— Можливість переглянути лоти, доступні у даний момент, та прийняти участь в аукціоні, тобто зробити ставку на обраний лот.

— Реєстрація у веб-застосунку із створенням особистого кабінету користувача.

— Купити лот за сталою ціною без необхідності робити ставку.

— Пожертвувати гроші незареєстрованому користувачеві без участі в аукціоні.

— Надання визначеним користувачам ролі адміністратора з можливістю входу в панель адміністратора.

— Для адміністратора: можливість додавання, редагування та видалення лотів за допомогою зручного інтерфейсу.

— Відображення панелі прогресу, що показує відношення коштів, які уже зібрано за місяць, до запланованої кількості (виставляється адміністратором у панелі адміністратора).

Веб-система побудована за принципом дворівневої клієнт-серверної архітектури, тобто є клієнт — веб-браузер користувача, що приймає дії користувача, база даних, що зберігає дані про користувачів, аукціони, лоти тощо, та сервер, що отримує запити від клієнта, зв’язується з базою даних і бере звідти потрібну інформацію, та передає клієнту відповідь на отриманий від користувача запит.

Потенційними користувачами даного веб-застосунку є всі люди, які мають доступ до мережі інтернет та веб-браузер. Наявність комп’ютера є необов’язковою, можна користуватись і з планшета, із мобільного телефона. Це люди, в яких є намір, бажання та можливість долучитись до благодійної діяльності й змінити чиєсь життя на краще.

2. АНАЛІЗ ПРОБЛЕМИ ОРГАНІЗАЦІЇ БЛАГОДІЙНОЇ ДІЯЛЬНОСТІ В УКРАЇНІ

Благодійна діяльність в Україні на даний момент є не дуже поширеною, що є наслідком багатьох факторів, таких як, наприклад, складна юридична сторона питання, або недовіра людей до благодійних заходів, адже багато хто мав негативний досвід зустрічей з шахраями.

В зв'язку з подіями 2014 року набагато більше людей почали займатись благодійністю, особливо багато коштів було зібрано на допомогу військовим у зоні АТО, проте тенденція поки не дуже задовільна[2]. Відсоток людей, що займаються благодійною діяльністю, залишається досить низьким.

2.1. Аналіз існуючих способів надання благодійної допомоги

У ході свого дослідження та аналізу наявних способів долучитись до благодійності онлайн, я знайшла сайти благодійних фондів, на яких є можливість зробити пожертву.

Наприклад, нижче наведені одні з найпопулярніших:

— «Благо / Допомагати просто» — благодійних фонд від ПриватБанку.

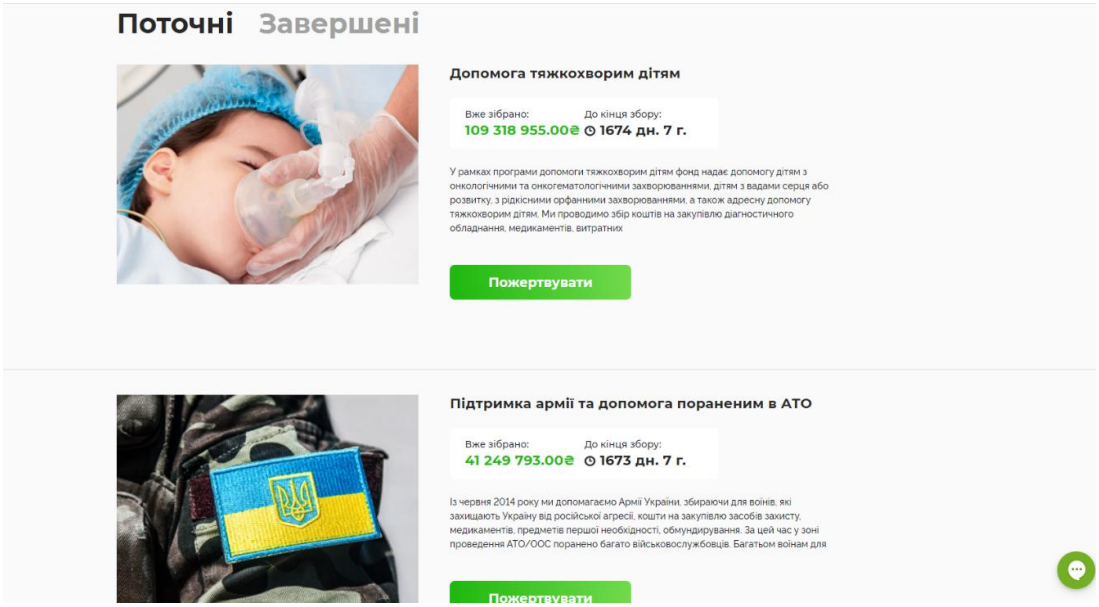

Рисунок 2.1 – Інтерфейс сайту благодійного фонду «Благо/ Допомагати просто»

Цей фонд наразі є одним із найбільших фондів — за кількістю зібраних на допомогу коштів— в Україні. Допомагає дана організація тяжкохворим дітям, воєнним, дитячим будинкам тощо.

У звіті фонду вказано, що за 2020 рік було надано допомоги на суму приблизно сімдесяти п’яти мільйонів гривень.

— «Таблеточки» — благодійний фонд, заснований відомою жінкою-лідером Ольгою Кудіненко.

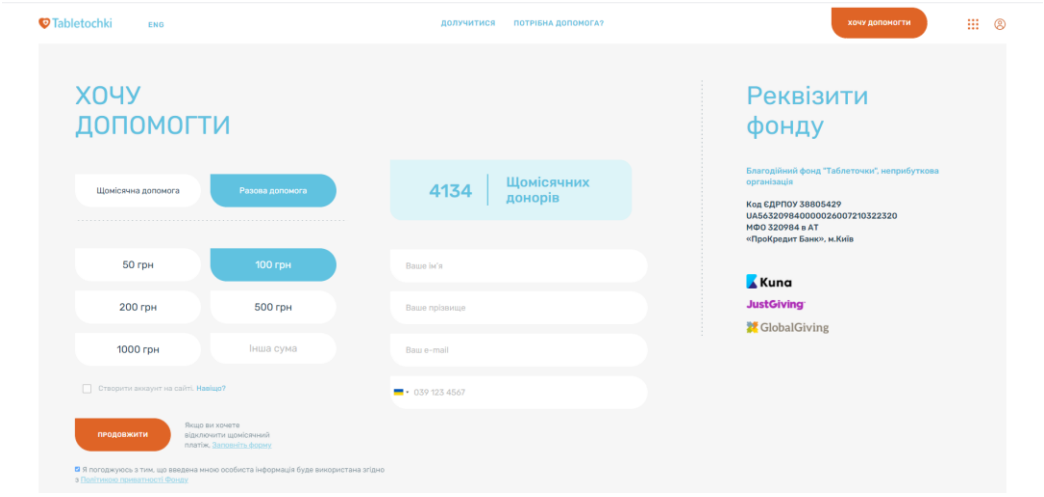

Рисунок 2.2 — Інтерфейс сайту благодійного фонду «Таблеточки»

Сам фонд займається допомогою онкохворим дітям і часто влаштовує співробітництва з відомими українськими бізнес-компаніями. За результатами звітності самого фонду, за 2019 рік було зібрано близько вісімдесяти чотирьох мільйонів гривень.

— «Помагаєм» — фонд, заснований волонтерами у 2009 році.

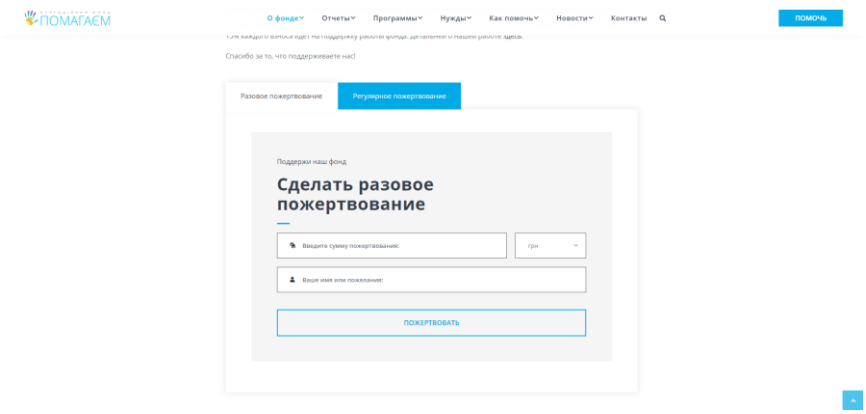

Рисунок 2.3 — Інтерфейс сайту благодійного фонду «Помагаєм»

Цей фонд поєднав дві групи небайдужих людей. По-перше, це волонтери, що працювали в дитячих лікарнях. По-друге, це прийомні батьки, що взяли на себе роль батьків для дітей-сирот. Основне направлення діяльності фонду — це допомога важкохворим дітям і дітям-сиротам. За 2020 рік було зібрано близько дванадцяти мільйонів гривень.

Переваги даних способів долучення до благодійності:

— Гарна звітність. Людям не потрібно здогадуватись, чи попадуть їхні гроші на допомогу.

— Досить зрозумілі користувацькі інтерфейси.

— Дуже легко знайти як зробити пожертвування.

Недолік:

Проаналізувавши ці сайти я знайшла єдиний, на мою думку, недолік. Я вважаю, що дані інструменти хороші, щоб вже замотивовані люди зробили пожертву. Проте, вони не можуть саме зацікавити людей у здійсненні благодійної діяльності.

2.2. Необхідність створення веб-системи з проведення благодійних аукціонів

Для того, щоб залучити більше людей для благодійності треба додати якусь «родзинку» або момент гри, щоб люди зацікавились і захотіли спробувати. Так виникла необхідність створення саме аукціону. Цей метод збору коштів у благодійних цілях є досить популярним у світі.

Дослідження показують, що аукціони з доходами, переданими на благодійні цілі, призводять до суттєво вищих продажних цін, що є результатом вищих ставок учасників з благодійними мотивами, а не збільшення кількості учасників торгів.

Також аукціони з 25% доходу, переданого на благодійність, мали більший чистий дохід, ніж недоброчинні аукціони. Отже, компанії можуть мати змогу використовувати благодійні аукціони як частину стратегії корпоративної соціальної відповідальності та одночасно підвищувати прибутковість, навіть незважаючи на те, що частину виручки вони віддають на благодійність.

Запорукою такого успіху на аукціонах є певний момент азарту. Те, що всі люди люблять вигравати — це загально-відомий факт. У випадку благодійних аукціонів ми можемо використати це задля доброї справи.

На благодійному аукціоні виграшний платіж приносить користь справі, яка, ймовірніше усього, оцінюється як учасником, так і конкуруючими учасниками. Таким чином, учасник тендеру отримує вигоду від власного платежу — як виграний предмет, так і цінність наданої допомоги, — так само й інші учасники торгів, оскільки їх благодійність підтримується. Отже, учасники тендеру мають дві основні цілі: виграти речі, які вони цінують, а також підтримати благодійну справу, частково підвищуючи ціну, що вони готові її заплатити.

Отже, система з організації благодійних аукціонів може стати ще одним інструментом для збору коштів для благодійних фондів, щоб зацікавити більше людей у допомозі тим, хто опинився у складній життєвій ситуації.

3. МЕТОДИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Спроектувавши систему та оцінивши задачу, було вирішено приділити особливу увагу засобам розробки, адже вони є основою успішної побудови системи.

3.1. Вибір архітектурного рішення

У якості архітектури було обрано паттерн MVC[8]. Він являє собою архітектурне рішення, що складається з трьох взаємопов'язаних частин. Включає в себе: модель або Model (дані застосунку або бізнес-логіка), представлення або View (інтерфейс користувача) та контролер або Controller (процеси й методи, що займаються обробкою вхідних даних).

Модель MVC часто використовується саме для розробки сучасних інтерфейсів, адже забезпечує компоненти для розробки як для мобільних пристроїв, так і для веб-застосунків. Також перевагою такої моделі є те, що вона добре співпрацює з об'єктно-орієнтованим підходом програмування, адже загалом різні моделі, представлення та контролери можна розглядати як об'єкти, тобто їх можна використовувати у програмі повторно.

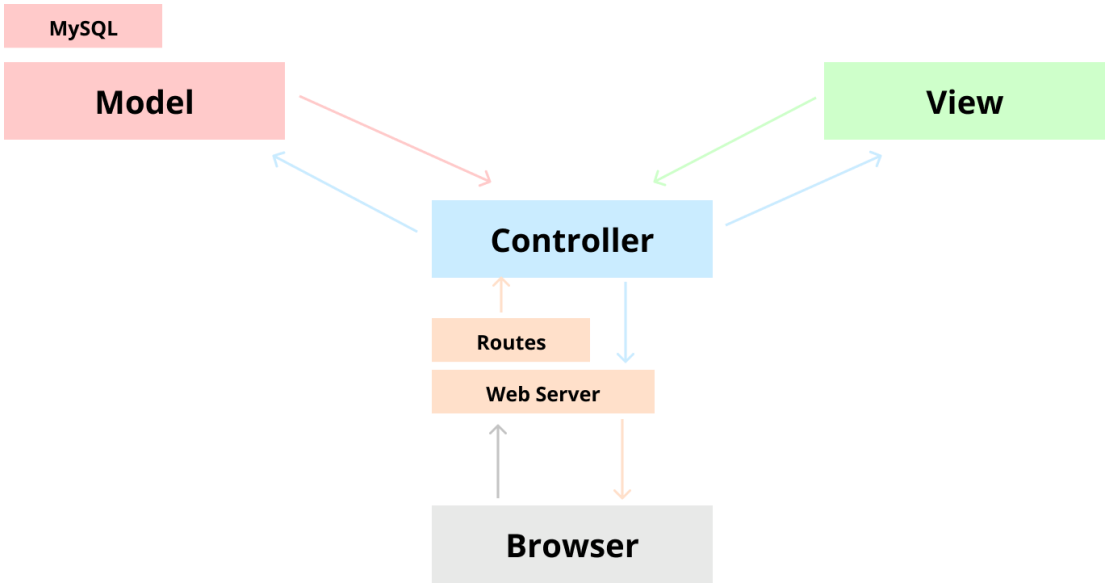

Рисунок 3.1 — Модель MVC

Компоненти паттерну MVC:

— Модель (Model). Це дані, що програма використовує для своєї роботи. Наприклад, база даних, об’єкти або файли.

— Представлення (View). Це спосіб відображення об’єктів програми. Включає в себе відображення кнопок, вікон або текстів. Тобто відповідає за все, що бачить користувач під час використання програми.

— Контролер (Controller). Контролер займається оновленням моделей та представлень. Він приймає дії користувача та виконує відповідне оновлення або дії.

На рисунку 3.1 показано, що всі частини моделі пов’язані між собою і постійно взаємодіють для надання певної функціональності.

Хоча даний паттерн був потрібний для розробки додатків, багато мов програмування та середовищ розробки підтримує цю архітектуру, а отже вона стала розповсюдженим вибором серед розробників.

3.2. Опис інструментів розробки

Для розробки даного програмного продукту були використані такі засоби:

— Figma
— Середовище розробки JetBrains PHPStorm.
— HTML
— CSS
— Bootstrap
— jQuery
— PHP (Laravel)
— MySQL

3.2.1. Сервіс розробки інтерфейсів Figma

Сервіс Figma[9] є веб-програмою для створення та дизайну користувацьких інтерфейсів та редагування графіки. Цей інструмент можна використовувати для

створення усіх видів графічного та веб-дизайну, для розробки інтерфейсів для мобільних додатків, прототипування, підготовки публікацій для соціальних мереж тощо.

Ця програма відрізняється від іншим інструментів насамперед тим, що може працювати безпосередньо у браузері. Це надає можливість отримувати доступ до своїх проектів з будь-якого пристрою, що є дуже зручним.

Ще одною перевагою саме цього сервісу є його достатньо широкий безкоштовний план, якого вистачить для роботи над невеликими проектами.

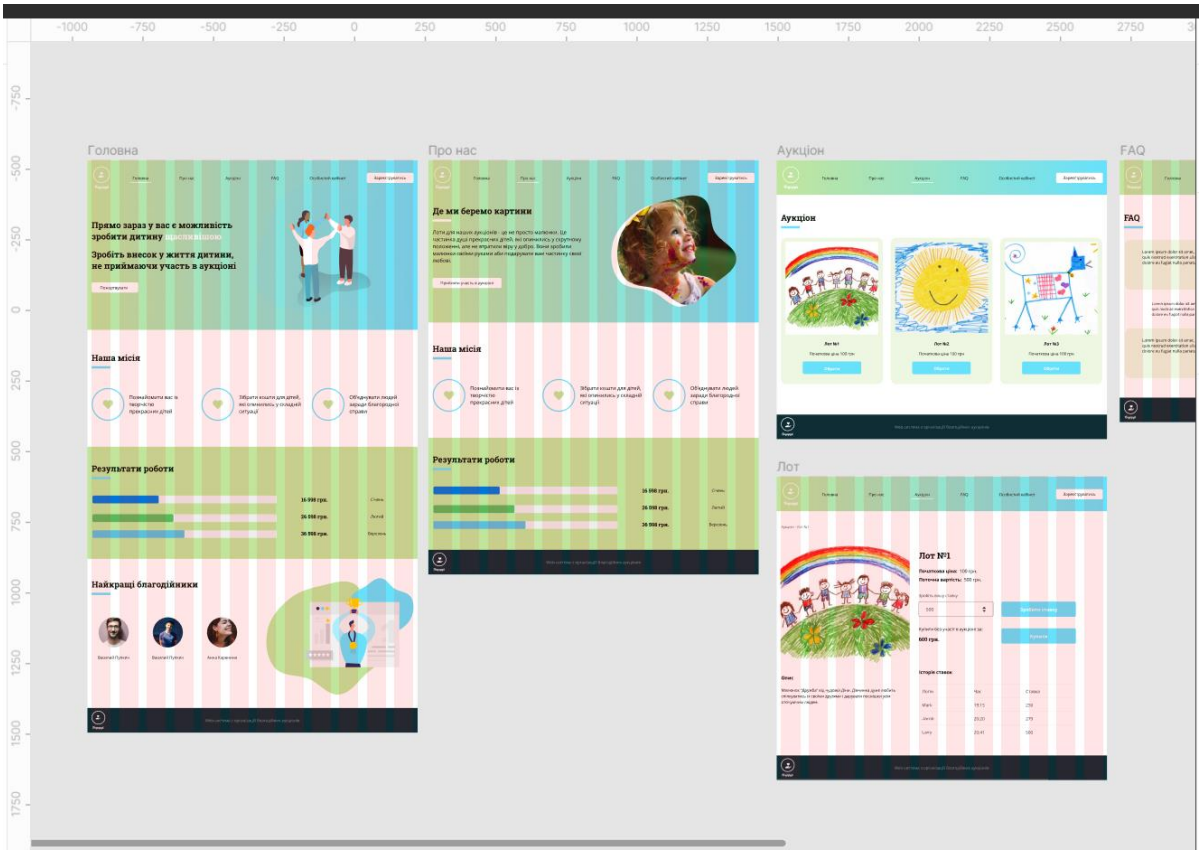

Рисунок 3.2 — Інтерфейс програми Figma

3.2.2. Середовище розробки JetBrains PhpStorm

Середовище JetBrains PhpStorm[3] є інноваційним та крос-пратформним інтегрованим середовищем розробки. Воно чудово підходить для роботи з таким технологіями як: Symfony, Laravel, WordPress тощо.

Можливості PhpStorm IDE досить широкі, воно може забепечувати автоматичне заповнення та підказки, рефакторнг коду, попередження про помилки ще в момент

написання самого коду та виступати у якості розширеного редактора для HTML, CSS та JavaScript.

Інтегроване середовище також надає інструменти роботи з базами даних у проектах. Воно може з'єднуватись із базою даних, редагувати таблиці, відправляти запити та навіть генерувати діаграми.

Вся функціональність WebStorm є включеною в PhpStorm.

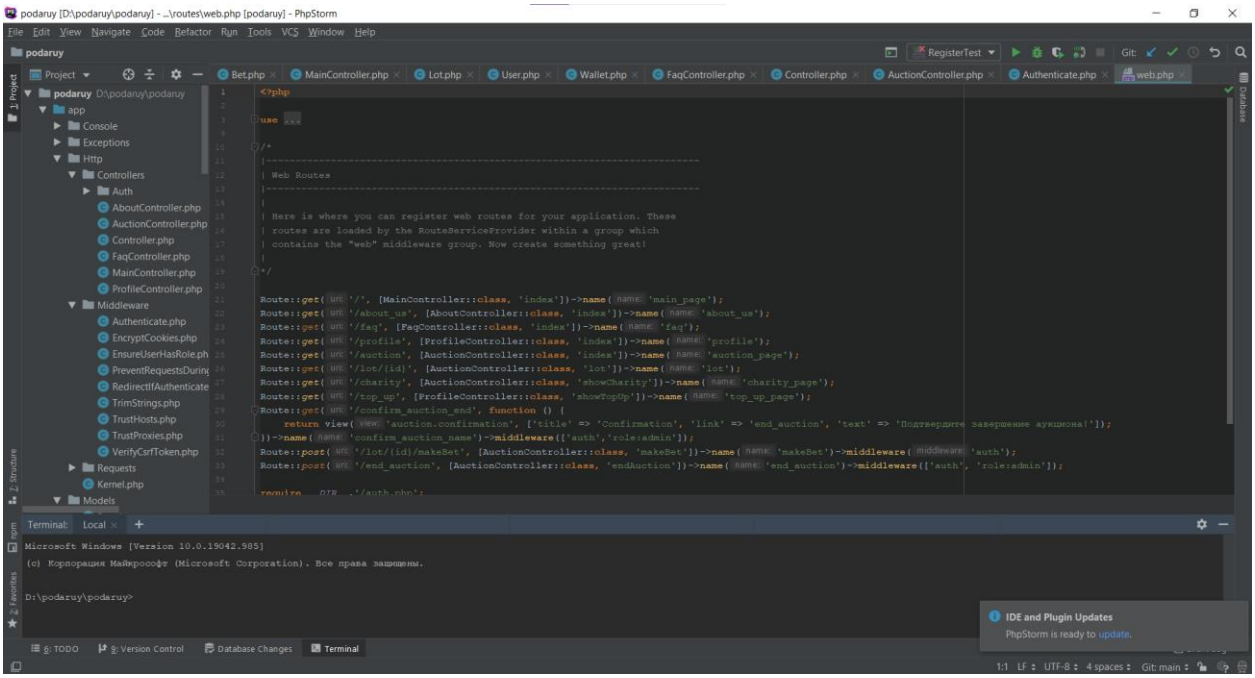

Рисунок 3.3 Інтерфейс PHPStorm

3.2.3. Мова розмітки HTML

Абревіатура HTML[7] розшифровується як Hyper Text Markup Language, тобто мова гіпертекстової розмітки. Ця мова використовується для створення кожної веб-сторінки або веб-додатку.

Що містить назва HTML:

— Гіпертекст або Hyper Text. Буквально це означає, що текст містить у собі посилання, отже є гіпертекстом. Загалом, це є способом пов'язування двох або більше веб-сторінок між собою. Тобто по натисканню на гіпертекст на одній сторінці, користувач попадає на іншу сторінку.

— Мова розмітки або Markup Language. Це комп’ютерна мова, що використовується для пояснень про розмітку та форматування відносно текстового документа. Мова розмітки здатна зробити текст більш інтерактивним і динамічним. Може перетворювати текст на таблиці, посилання та зображення.

— Веб-сторінка або Web page є документом, який частіше за все пишуть у форматі HTML. Потім цей документ перекладається браузером і відображається у ньому.

Сам документ складається з багатьох тегів, що мають різні правила, синтаксичний та логічний вміст.

3.2.4. Таблиці стилів CSS

Каскадні таблиці стилів[6], які всі називають просто CSS, є достатньо простою й зрозумілою мовою стилю або дизайну, яка була створена для того, щоб робити веб-сторінки більш красивими для користувача та презентабельнішими.

Мова CSS безпосередньо відповідає за зовнішній вигляд веб-сторінок. За допомогою цього інструменту можна керувати стилем шрифту, кольорами, розміщенням елементів, розмірами картинок, відображенням самої сторінки на екранах різних розмірів та додаванням деякої інтерактивності й анімації.

Даний інструмент чудово співпрацює з HTML та дозволяє повністю контролювати його відображення.

Перевагами використання CSS є:

— Зберігання часу розробника. Написавши лише один раз певні правила відображення, їх можна використовувати безліч разів. Немає ніякої потреби у тому, щоб писати одні і ті самі правила відображення для кожного схожого тегу. Це тим самим зменшує навантаження на браузер і робить відображення швидшим.

— Легке редагування. Для того, щоб змінити зовнішній вигляд великої кількості тегів потрібно лише змінити одне правило, яке застосоване до них. В результаті стиль відображення усіх елементів зміниться автоматично.

— Зручна сумісність деяких пристроїв. Прописавши розмітку та стилі для одного типу пристроїв, можна легко оптимізувати її й для інших пристроїв. При цьому не потрібно дублювати увесь написаний код, можна використовувати ті самі тексти розмітки та стилю.

— Широкий набір атрибутів. Порівняно зі стилями, що можуть бути використані безпосередньо у HTML документі, CSS надає набагато більший функціонал для забезпечення потрібного вигляду сторінки.

3.2.5. Фреймворк Bootstrap

Фреймворк Bootstrap[4] є досить потужним набором інструментів HTML, CSS та JavaScript для створення звучних веб-сторінок та веб-додатків. Він являє собою фрейморк з відкритим кодом, що з самого початку був створеним для Twitter.

Даний інструмент дуже швидко набрав популярність після свого виходу, адже він є простим та гнучким у роботі. Його основні переваги у тому, що він є сумісним з багатьма браузерами, пропонує можливість створення дизайнів за допомогою повторно використовуваних компонентів та є швидким для освоювання.

Чому цей інструмент є гарним вибором для розробників:

— Адаптивна сітка. Дизайнерам дуже часто треба створити правильне відображення для екранів різних розмірів. Створювати кожного разу свою сітку приносить незручності, тому тут на допомогу приходить Bootstrap. Тут система сітки уже є чітко визначеною, а отже можна зосередитись безпосередньо на вмісті контейнерів.

— Наявність компонентів. Сам фреймворк поставляється з великою кількістю готових компонентів, які можна або просто використати у своєму проекті, або змінити під власні потреби. Вибір досить великий, тут і випадаючі меню, і навігаційні панелі і різні індикатори тощо.

— Вичерпна документація. Кожен фрагмент коду можна переглянути та розібрати на офіційному веб-сайті. Тут же представлені пояснення з прикладами.

— Взаємодія з JavaScript. Якщо вам не вистачає певної інтерактивності або функцій, то на допомогу приходить бібліотека JQuery. Bootstrap дає можливість скористатись перевагами багатьох плагінів JQuery. У ній можна знайти багато рішень для налаштування інтерактивності, наприклад, спливаючі вікна, переходи, галереї зображень тощо.

3.2.6. Фреймворк Laravel

Фреймворк Laravel[5] є фреймворком з відкритим кодом, основою якого є мова програмування PHP. Він наслідує принципи архітектури MVC. Цей фреймворк використовує уже існуючі компоненти різних фреймворків, таким чином ставши максимально структурованим та прагматичним.

Сам Laravel містить досить багатий набір функцій та готових рішень, які прискорюють та полегшують процес веб-розробки[10].

Переваги даного інструмента розробки:

— Масштабування веб-програми. Так як Laravel використовуж компоненти з інших середовищ. То дозволяє значно економити час. Він включає в себе інтерфейси та простори імен, що в свою чергу дозволяє чітко організовувати ресурси й керування ними.

— Composer. Це інструмент, що включає всі бібліотеки та залежності. Він дає можливість легко і швидко встановлювати навіть сторонні бібліотеки для розробки.

— Випробовуваність. Фрейморк надає функції для зручного тестування. Таким чином можна підтримувати відповідність свого коду до заданих вимог.

— Маршрутизація. Забезпечується гнучкий підхід до визначення маршутів у програмі. Це допомагає у задачі масштабування веб-додатку та підвищенні його продуктивності.

— Конструктор запитів та ORM. Laravel включає конструктор запитів, що дозволяє легко будувати свої запити до бази даних, використовуючи різні ланцюгові методи.

— Автентифікація. Ця функціональність є загальною для більшості веб-застосунків. Тому такі функції як реєстрація, автентифікація, відновлення пароля є включеними.

— Зрозуміла та повна документація, яка дозволить використовувати усі можливості даного інструменту як досвідченим розробника, так і новачкам.

3.2.7. Система управління MySQL

Система управління MySQL[1] — це система управління реляційними базами даних (СУБД) із використання моделі клієнт-сервер. СУБД є програмним забезпеченням, яке створене й використовується для управління базами даних реляційного типу.

База даних — це місце, де упорядковано зберігаються дані. «Реляційний» означає, що всі дані впорядковані й організовані у вигляді таблиць, кожна з яких певним чином пов’язана з іншими.

MySQL може працювати на різних платформах, його можна встановити на сервері або на робочому столі. Крім цього, MySQL є швидким і масштабованим. Ця система є чудовими вибором при розробці веб-додатків або веб-сайтів.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Робота почалась з проектування. Адже перед тим як писати код, спочатку треба зрозуміти як організувати систему в цілому. Після проведення аналізу було виділено й враховано основний недолік існуючих систем.

4.1. Загальна інформація

На сайті влаштовуються аукціони, на яких продаються дитячі малюнки (виставляються на аукціон). Усі гроші йдуть у фонд для дітей, які того потребують. Фонд може вирішувати на які потреби відправляти фінансову допомогу. Сайт повинен викликати довіру, бути не дуже яскравим, але у той же момент повинно бути зрозуміло, що мова йде про дітей, яким ми допомагаємо.

4.2. Опис сторінок

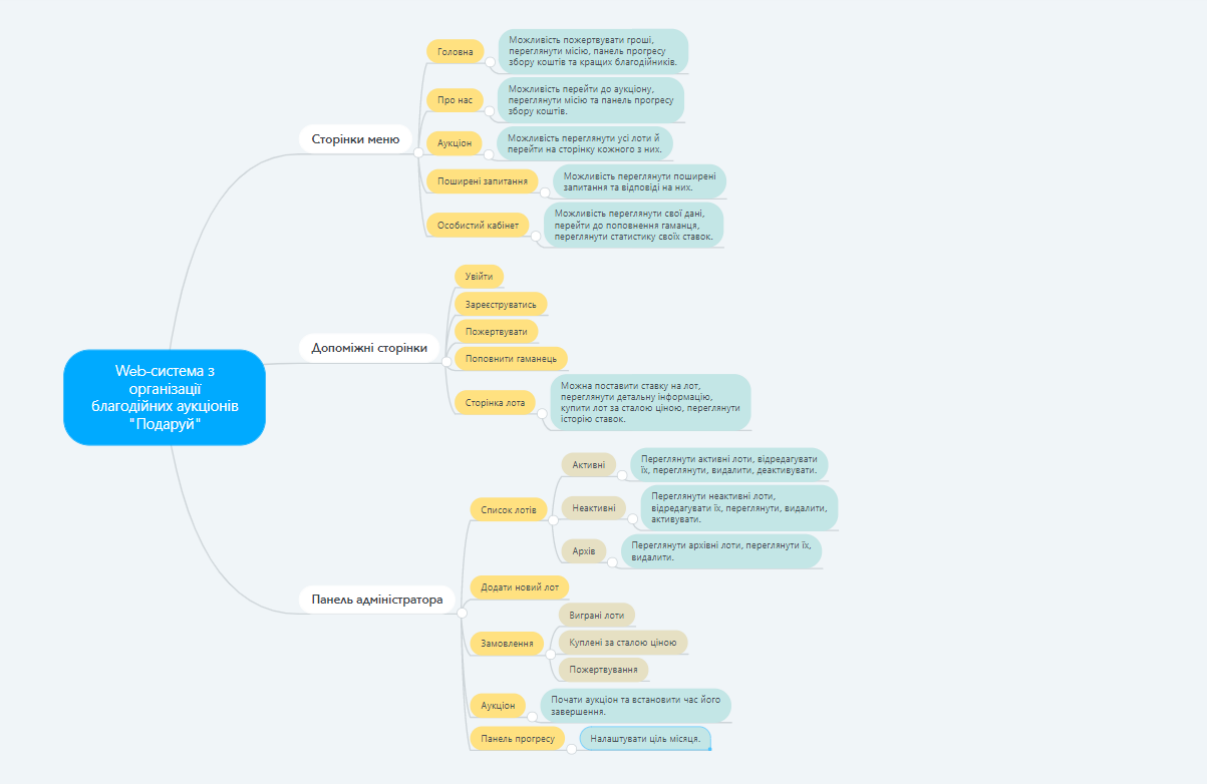

Рисунок 4.1 — Опис сторінок веб-системи

На рисунку 4.1 представлено ієрархію та опис сторінок веб-системи за допомогою MindMap.

4.3. Діаграма прецедентів

Діаграма прецедентів або, як її ще називають, діаграма випадків використання — це основна форма упорядкування програмних вимог до системи. Кожній випадок містить очікувану поведінку системи, тобто «що» зробить система, а не спосіб здійснення[11]. Основним завданням даної діаграми є те, що вона має показати систему з точки зору користувача.

Зазвичай, даний вид діаграми є достатньо прости, вона лише узагальнює зв’язки між певними випадками використання, сторонніми системами та об’єктами.

Основне в діаграмі — це актори, тобто хто виконує певну дію, та самі прецеденти, тобто самі дії.

На рисунку 4.2 представлена діаграма прецедентів моєї дипломної роботи.

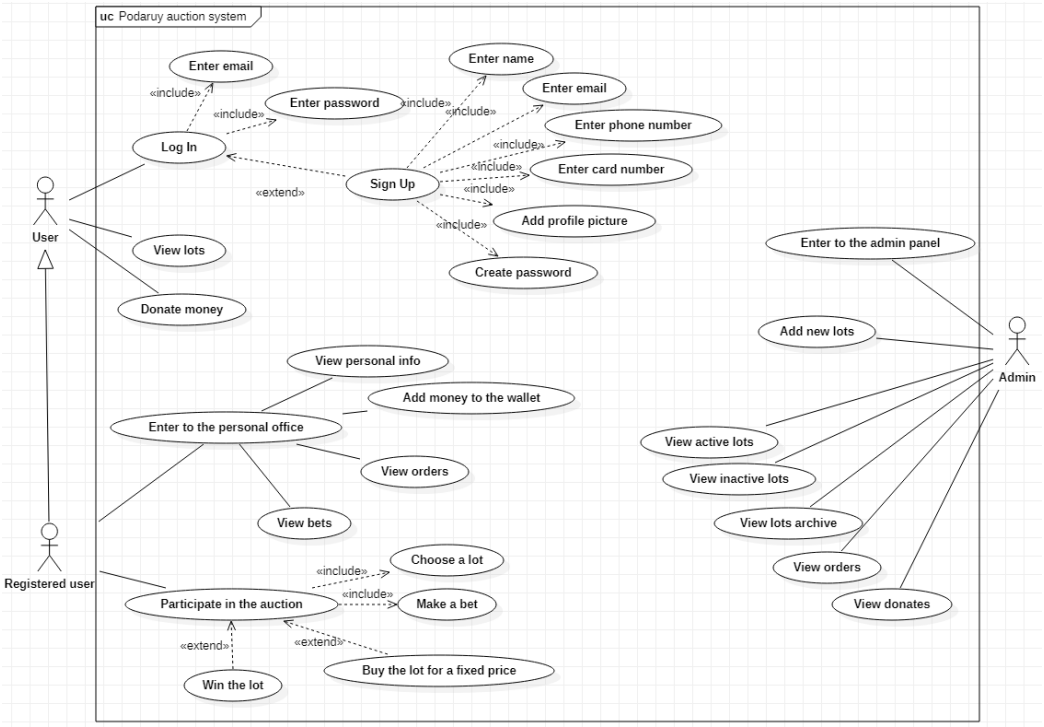

Рисунок 4.2 — Діаграма прецедентів веб-системи «Подаруй»

Акторами моєї системи є користувач, зареєстрований користувач, що наслідує

можливості незареєстрованого користувача та адміністратор.

Згідно з рисунком 4.2 також можна виділити основний сценарій роботи користувача із системою:

— Користувач заходить на сайт та виконує вхід за допомогою логіна і пароля. Якщо він ще не зареєстрований, то потрібно спочатку зареєструватись, щоб мати можливість брати участь у аукціоні.

— Користувач заходить на сторінку «Аукціон» і бачить усі лоти, які наразі є доступними. Він дивиться лоти і обирає ті, які його цікавлять.

— При натисканні на кнопку «Обрати» користувач попадає на сторінку обраного лоту, де у нього з’являється можливість поставити свою ставку. На цій же сторінці відображається історія всіх ставок на даний лот. Основна логіка сторінки полягає у тому, що йде перевірка і запис усіх ставок, і користувач не може поставити менше грошей, аніж уже поставила остання людина. Більше того, користувач має можливість купити лот за сталою ціною, якщо не хочу приймати участь у змаганні.

— Після закриття аукціону обирається переможець для кожного лоту.

4.4. Діаграма класів

Діаграма класів призначена для того, щоб визначати типи класів системи та існуючі зв’язки між ними[11]. На ній зображують класи, їх атрибути, операції та можливі обмеження, що накладаються на зв’язки. Діаграма може виглядати по-різному в залежності від обраного рівня абстракції.

Класи в діаграмі характеризуються своїми атрибутами та операціями.

Атрибут являє собою властивість об’єкта класу, а операція є функцією або методом, тож може мати якісь параметри та повертати певні значення.

Якщо узагальнити, то мета діаграми класів — це аналіз і проектування статичного подання програми та описання обов’язків системи.

На рисунку 4.3 представлено діаграму класів даної системи.

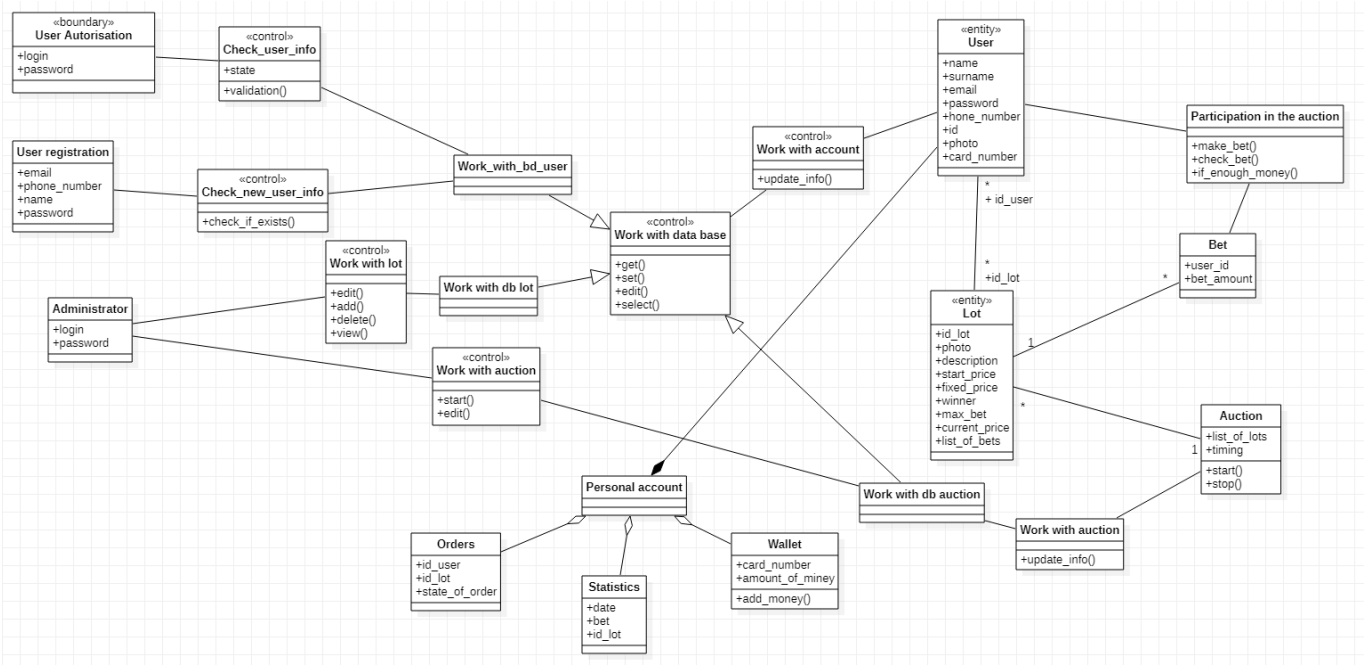

Рисунок 4.3 — Діаграма класів веб-системи «Подаруй»

Так як проект є веб-сайтом, то в основному робота проводиться із базою даних, де розміщено інформацію про користувачів, лоти тощо.

Основними сутностями є Користувач (User) та Лот (Lot). Об’єкти цих класів будуть відповідати за окремих користувачів та окремі лоти в системі

4.5. Робота з базою даних

На рисунку 4.4 представлені таблиці, використані у роботі, та зв’язки між ними.

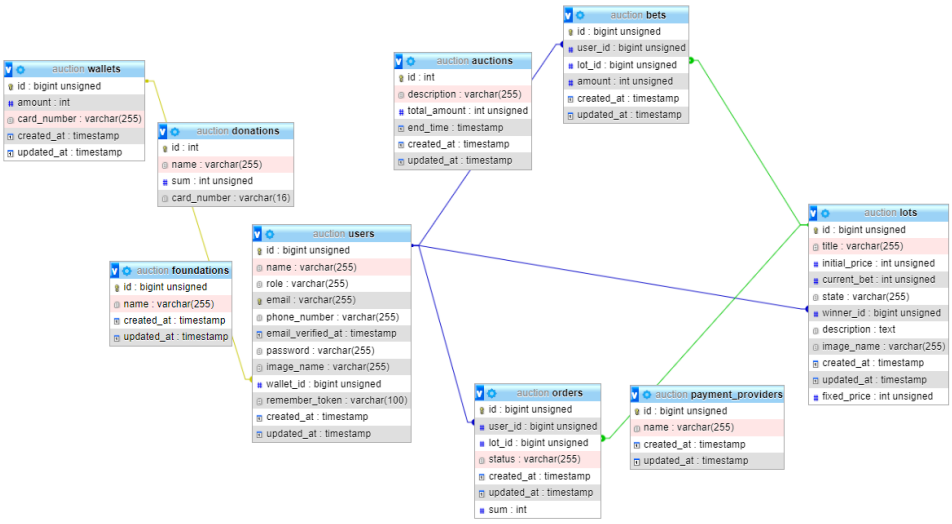

Рисунок 4.4 — Схема організації бази даних

З огляду на те, що веб-сайт в основному зав’язаний на отриманні та відправленні різноманітних даних, то потрібно дуже уважно і ретельно підійти до питання створення правильних таблиць та зв’язків між ними задля коректної роботи застосунку.

Сам фреймворк Laravel включає у себе ORM (object-relational mapper) Eloquent, який робить роботу з даними та базами даних набагато приємнішою. Коли використовується Eloquent, то кожна таблиця у базі даних повинна мати відповідну модель, яка використовується для управління відповідною таблицею. Моделі Eloquent мають широкі можливості, вони можуть отримувати записи з таблиць, оновлювати дані, вставляти або видаляти їх.

Для наглядності в таблицях 4.1 – 4.2 представлено структуру основних таблиць бази даних.

Таблиця 4.1 — Структура сутності «Users»

Назва поля	**Тип поля**	**Опис**
id	bigint	Ідентифікатор
name	varchar	Ім’я користувача
role	varchar	Роль
email	varchar	Поштова адреса
phone_number	varchar	Номер телефону
password	varchar	Пароль
image_name	varchar	Аватар
wallet_id	bigint	Ідентифікатор гаманця
created_at	timestamp	Час створення
updated_at	timestamp	Час оновлення

Таблиця 4.1 повністю відповідає за всю інформацію про користувача і пов’язана з відповідною моделлю «User» у програмному коді.

Таблиця 4.2 — Структура сутності «Lots»

Назва поля	**Тип поля**	**Опис**
id	bigint	Ідентифікатор
title	varchar	Заголовок
initial_price	int	Початкова ціна
current_bet	int	Поточна ціна
state	varchar	Стан
winner_id	bigint	Ідентифікатор переможця
description	text	Опис
image_name	varchar	Шлях до зображення
fixed_price	int	Стала ціна покупки
created_at	timestamp	Час створення
updated_at	timestamp	Час оновлення

Аналогічно до таблиці користувачів, таблиця лотів 4.2 відповідає за інформацію про лоти та пов’язана з відповідною моделлю.

```php
public function up()
{
    Schema::create( table: 'users', function (Blueprint $table) {
        $table->id();
        $table->string( column: 'name');
        $table->string( column: 'role')->default( value: 'user');
        $table->string( column: 'email')->unique();
        $table->string( column: 'phone_number')->nullable();
        $table->timestamp( column: 'email_verified_at')->nullable();
        $table->string( column: 'password');
        $table->string( column: 'image_name');
        $table->unsignedBigInteger( column: 'wallet_id')->nullable();
        $table->foreign( columns: 'wallet_id')->references( columns: 'id')->on( table: 'wallets');
        $table->rememberToken();
        $table->timestamps();
    });

}
```

Рисунок 4.5 — Приклад створення таблиці

Усі таблиці в БД були створені за допомогою спеціальних міграцій Laravel

(рисунок 4.5). Для кожної таблиці в базі даних було створено модель та прописано зв’язки.

4.6. Маршрутизація

За архітектурою MVC для кожного маршруту встановлюється контролер, який його обробляє, та дія, яка представлена функцією в даному контролері.

```
Route::get( uri: '/', [MainController::class, 'index'])->name( name: 'main_page');
Route::get( uri: '/about_us', [AboutController::class, 'index'])->name( name: 'about_us');
Route::get( uri: '/faq', [FaqController::class, 'index'])->name( name: 'faq');
Route::get( uri: '/profile', [ProfileController::class, 'index'])->name( name: 'profile');
Route::get( uri: '/auction', [AuctionController::class, 'index'])->name( name: 'auction_page');
Route::get( uri: '/lot/{id}', [AuctionController::class, 'lot'])->name( name: 'lot');
Route::get( uri: '/charity', [AuctionController::class, 'showCharity'])->name( name: 'charity_page');
Route::get( uri: '/top_up', [ProfileController::class, 'showTopUp'])->name( name: 'top_up_page');
Route::get( uri: '/confirm_auction_end', function () {
    return view( view: 'auction.confirmation', ['title' => 'Confirmation', 'link' => 'end_auction', 'text' => 'Підтвердіть закінчення аукціону!']);
})->name( name: 'confirm_auction_name')->middleware(['auth','role:admin']);
Route::post( uri: '/lot/{id}/makeBet', [AuctionController::class, 'makeBet'])->name( name: 'makeBet')->middleware( middleware: 'auth');
Route::post( uri: '/end_auction', [AuctionController::class, 'endAuction'])->name( name: 'end_auction')->middleware(['auth', 'role:admin']);

//admin
Route::prefix( prefix: 'admin')->middleware(['auth', 'role:admin'])->group(function () {
    Route::get( uri: '', [CommonController::class, 'index'])->name( name: 'admin_index');
    Route::get( uri: '/lot/active', [LotController::class, 'indexActive'])->name( name: 'active_lots');
    Route::get( uri: '/lot/inactive', [LotController::class, 'indexInactive'])->name( name: 'inactive_lots');
    Route::get( uri: '/lot/create', [LotController::class, 'create'])->name( name: 'create_lot');
    Route::get( uri: '/lot/archive', [LotController::class, 'indexArchive'])->name( name: 'archive_lots');
    Route::post( uri: '/lot/store', [LotController::class, 'store'])->name( name: 'store_lot');
    Route::post( uri: '/lot/update/{id}', [LotController::class, 'update'])->name( name: 'update_lot');
    Route::get( uri: '/lot/{id}', [LotController::class, 'showLot'])->name( name: 'lot_info');
    Route::get( uri: '/lot/edit/{id}', [LotController::class, 'edit'])->name( name: 'edit_lot');
    Route::post( uri: '/lot/delete/{id}', [LotController::class, 'delete'])->name( name: 'delete_lot');
    Route::post( uri: '/lot/deactivate/{id}', [LotController::class, 'deactivate'])->name( name: 'deactivate_lot');
    Route::post( uri: '/lot/activate/{id}', [LotController::class, 'activate'])->name( name: 'activate_lot');

    Route::get( uri: '/order/won', [OrderController::class, 'indexWon'])->name( name: 'won_lots');
    Route::get( uri: '/order/sold', [OrderController::class, 'indexSold'])->name( name: 'sold_lots');
    Route::get( uri: '/donation', [OrderController::class, 'showDonations'])->name( name: 'donations');
    Route::get( uri: '/start-auction', [CommonController::class, 'indexStart'])->name( name: 'index_auction_start');
    Route::post( uri: '/start-auction', [CommonController::class, 'startAuction'])->name( name: 'start_auction');
    Route::get( uri: '/progress-options', [CommonController::class, 'indexProgressBar'])->name( name: 'index_progress_options');
    Route::post( uri: '/progress-options', [CommonController::class, 'setProgressOptions'])->name( name: 'set_progress_options');
    Route::post( uri: 'lot/buy/{id}', [AuctionController::class, 'buyLot'])->name( name: 'buy_lot')->middleware( middleware: 'auth');
});
```

Рисунок 4.6 — Маршрути програми

При обробці маршруту викликається відповідна функція контролера. Контролери використовують моделі, в яких проводяться маніпуляції з даними: вони змінюються або просто беруться , або записуються. Далі дані з моделі передаються у представлення, тобто шаблони відображення, які написані за допомогою шаблонізатора blade, який є встроєним у Laravel. Далі представлення вертаємо на контролер, а він уже повертає відповідну сторінку з потрібними даними з моделі.

4.7. Контролер AuctionController

За основний функціонал проведення аукціону відповідає AuctionController.

Функції контролера:

— index(). Бере всі активні лоти і відображає їх на екрані за допомогою модуля view, в якому знаходиться логіка відображення.

— lot(). Виконує аналогічні дії, але виводить один лот і ставки на ньому. Також даний метод перевіряє, чи сплив час аукціону та відповідає за виведення лічильника.

```
public function lot($id)
{
    $remainingTime = 0;
    $lot = Lot::find($id);
    $bets = Bet::where( string: 'lot_id', $lot->id)->orderBy('created_at', 'DESC')->get();
    $collection = Auction::all();
    if (1 === \count($collection)) {
        $auction = $collection->first();
        $remainingTime = Carbon::createFromFormat( format: CommonController::LARAVEL_FORMAT, $auction->end_time)->diffInRealSeconds(Carbon::
        now());
    }

    return view( view: 'auction.lot_page', ['lot' => $lot, 'bets' => $bets, 'remainingTime' => $remainingTime]);
}
```

Рисунок 4.7 — Метод lot()

— makeBet(). По вибраному лоту створює об’єкт моделі «Ставка», перевіряє наявність потрібної суми на рахунку, списує гроші з балансу користувача та зберігає зроблену ставку до бази даних.

```
public function makeBet($id, Request $request): RedirectResponse
{
    $lot = Lot::find($id);
    $wallet = User::find(Auth::id())->wallet;
    $amount = $request->amount;
    if ($lot->current_bet >= $amount) {

        return Redirect::back()->withErrors(['Занадто маленька ставка!']);
    }
    if ($amount > $wallet->amount) {
        return Redirect::back()->withErrors(['Недостатньо коштів на балансі!']);
    }
    $wallet->amount = $wallet->amount - $amount;
    $wallet->save();

    $lot->current_bet = $amount;
    $lot->save();

    $bet = new Bet();
    $bet->setAttribute('lot_id', $id);
    $bet->setAttribute('user_id', Auth::id());
    $bet->setAttribute('amount', $amount);
    $bet->save();

    return Redirect::back()->with('message', "Дякую за вашу ставку!");
}
```

Рисунок 4.8 — Метод makeBet()

— finish(). Бере всі лоти, по яким були зроблені ставки. Обирає найвищу ставку, створює екземпляр моделі «Замовлення» з статусом «Виграно», і присвоює це замовлення користувачу з найвищою ставкою. Зберігає замовлення в базу даних. Під час виконання функції всі лоти, по яким було визначено переможця, переходять в архів, де їх може побачити адміністратор. Для лотів, на які не було поставлено ставку передбачено стан «Неактивний», їх також може потім подивитись адміністратор.

```php
public function finish(): void
{
    $lots = Lot::getAllActiveWithBets();

    foreach ($lots as $lot) {
        $highestBet = $lot->getHighestBet();

        if (null === $highestBet) {
            $lot->state = 'conflict';
            $lot->save();
            continue;
        }

        $user = $highestBet->user;

        $order = new Order();
        $order->user_id = $user->id;
        $order->lot_id = $lot->id;
        $order->status = 'won';
        $order->sum = $highestBet->amount;
        $order->save();

        $lot->winner_id = $user->id;
        $lot->state = 'archive';
        $lot->save();
    }

    $lotsWithoutBets = Lot::getAllActiveWithoutBets();

    foreach ($lotsWithoutBets as $lot) {
        $lot->state = 'inactive';

        $lot->save();
    }
```

Рисунок 4.9 — Метод finish()

— buyLot(). Відповідає за можливість покупки лота за сталою ціною. Перевіряє чи вистачає у користувача на балансі коштів. Якщо ні, то повідомляє про те, що він не може купити це лот. Якщо все в порядку, і коштів вистачає, то створює екземпляр класу «Замовлення» та зберігає його до БД. Також проводить операцію зняття коштів

з балансу користувача. Сам лот відправляється до архіву, де зберігаються продані лоти.

```php
public function buyLot($id)
{
    $lot = Lot::find($id);
    $wallet = User::find(Auth::id())->wallet;

    if ($lot->fixed_price > $wallet->amount) {
        return Redirect::back()->withErrors(['Недостатньо коштів на балансі!']);
    }

    $wallet->amount = $wallet->amount - $lot->fixed_price;
    $wallet->save();

    $order = new Order();
    $order->user_id = Auth::id();
    $order->lot_id = $lot->id;
    $order->status = 'sold';
    $order->sum = $lot->fixed_price;
    $order->save();

    $lot->state = 'archive';
    $lot->save();

    return redirect(route( name: 'auction_page'))->with('message', 'Дякую за покупку!');
}
```

Рисунок 4.10 — Метод buyLot()

4.8. Допоміжні функції

Для обчислення статистики було розроблено метод getSumOfBetsByLastMonths() у MainController, який повертає загальні суми всіх коштів, що були використані у ставках по місяцям.

```php
public static function getSumOfBetsByLastMonths(Collection $bets): array
{
    $amountByMonth = [];

    foreach ($bets as $bet) {
        if (!isset($amountByMonth[self::$months[Carbon::parse($bet->created_at)->month]]))
            $amountByMonth[self::$months[Carbon::parse($bet->created_at)->month]] = 0;
        $amountByMonth[self::$months[Carbon::parse($bet->created_at)->month]] += $bet->amount;
    }

    return $amountByMonth;
}
```

Рисунок 4.11 — Метод getSumOfBetsByLastMonths()

Також було створено файл допоміжних функцій, що доступні у всьому проекті.

Наприклад, функція getColorByPercent() повертає кольори в залежності від відсотка забраної суми відносно цілі на місяць.

```php
function getColorByPercent(int $number): string
{
    $percent = getPercentOfGoal($number);
    $color = '';

    switch ($percent) {
        case (0 < $percent && 25 > $percent):
            $color = '#FF0000';
            break;
        case 50 > $percent:
            $color = '#FFA500';
            break;
        case 75 > $percent:
            $color = '#F5EF42';
            break;
        default:
            $color = '#6BFC03';
    }

    return $color;
}

function getPercentOfGoal($amount) {
    $goal = (int) Option::where('name', 'goal')->first()->value;
    return ($amount / $goal) * 100;
}
```

Рисунок 4.12 — Метод getColorByPercent()

Отже, адміністратор може виставити яку кількість грошей ми плануємо зібрати у кожному місяці, і в залежності від цієї цифри буде малюватись прогрес. Колір змінюється від червоного (забрали менше 25 відсотків від запланованої суми) до зеленого (зібрали 75 відсотків і більше).

4.9. Контролер панелі адміністратора LotController

Даний контролер відповідає за всі маніпуляції з лотами у адміністративній панелі. Має такі методи:

— indexActive(). Відображає сторінку активних лотів, які зараз виставлені на аукціоні.

— indexInactive(). Аналогічно попередньому, але сторінку неактивних лотів, які зараз не беруть участь в торгах.

— indexArchive(). Аналогічно попереднім, але сторінку лотів, що в архіві, тобто уже виграні або куплені лоти.

— showLot(). Відповідає за відображення інформації про обраний лот.

— edit(). Виконує перехід на сторінку редагування.

— update(). Проводить валідацію даних, що були введені на сторінці редагування та записує нову інформацію у відповідні поля.

```php
public function update($id, Request $request)
{
    $request->validate([
        'title' => 'required|string|max:255',
        'description' => 'required|string|max:255',
        'initial_price' => 'required|integer',
        'fixed_price' => 'required|integer',
        'image' => 'image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        'activation' => 'bool'
    ]);

    $imageName = null;

    if (null !== $request->image) {
        $imageName = time() . '.' . $request->image->extension();
        $request->image->move(public_path( path: 'lots'), $imageName);
    }

    $lot = Lot::find($id);

    $lot->title = $request->title;
    $lot->description = $request->description;
    $lot->initial_price = $request->initial_price;
    $lot->fixed_price = $request->fixed_price;
    null === $imageName ?: $lot->image_name = $imageName;
    true != $request->activation ?: $lot->state = 'active';


    $lot->save();

    return back()->withInput()->with('message', 'Успішно оновлено!');
}
```

Рисунок 4.13 — Метод update()

— deactivate(). Відправляє активний дот до неактивних.

— activate(). Аналогічно попередньому, проте зворотна операція.

— delete(). Видаляє певний лот.

— create(). Виконує перехід на сторінку додавання нового лота.

— store(). Схожий за функціональністю з методом update().

```php
public function store(Request $request)
{
    $request->validate([
        'title' => 'required|string|max:255',
        'description' => 'required|string|max:255',
        'initial_price' => 'required|integer',
        'fixed_price' => 'required|integer',
        'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        'activation' => 'bool'
    ]);

    $imageName = time() . '.' . $request->image->extension();
    $request->image->move(public_path( path: 'lots'), $imageName);

    $activation = true == $request->activation;

    $lot = new Lot();

    $lot->title = $request->title;
    $lot->description = $request->description;
    $lot->initial_price = $request->initial_price;
    $lot->current_bet = $request->initial_price;
    $lot->fixed_price = $request->fixed_price;
    $lot->image_name = $imageName;
    $lot->state = $activation ? 'active' : 'inactive';


    $lot->save();

    $redirectRoute = $activation ? 'active_lots' : 'inactive_lots';

    return redirect()->route($redirectRoute)->withInput()->with('message', 'Успішно створено!');
}
```

Рисунок 4.14 — Метод store()

Він проводить певну валідацію даних: перевіряє, щоб всі поля були заповнені, щоб картинки мали правильне розширення та щоб типи введених даних були коректними. Також записує нову інформацію до бази даних та повідомляє адміністратора про те, що операцію проведено успішно.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Щоб користувач зміг безперебійно користуватись програмою треба дотримуватись основних вимог.

5.1. Системні вимоги

Для коректної роботи застосунку у користувача локально на комп’ютері має бути папка з усіма файлами, необхідними для роботи. На даний момент веб-застосунок працює тільки локально, але надалі розглядається можливість його розгортання на сервері й додання в мережу на відповідних хостингу та домені.

5.2. Сценарій роботи користувача з системою

Користувач має змогу взаємодіяти з системою за допомогою зручного та зрозумілого графічного інтерфейсу. Інтерфейс являє свою веб-сторінку, за замовчування робота починається зі сторінки «Головна».

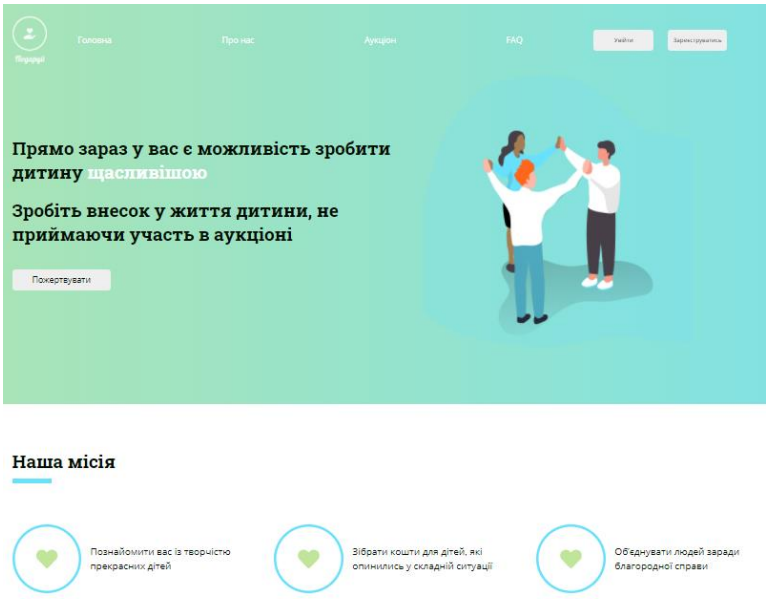

Рисунок 5.1 — Головна сторінка

На головній сторінці користувач може перейти до сторінки «Пожертвувати», переглянути інформацію про місію веб-системи, подивитись панель прогресу та кращих благодійників.

У хедері розміщені основні сторінки сайту, на які можна перейти натисканням відповідного пункту меню, такі як:

— «Про нас», де можна прочитати основну інформацію та перейти до сторінки аукціону.

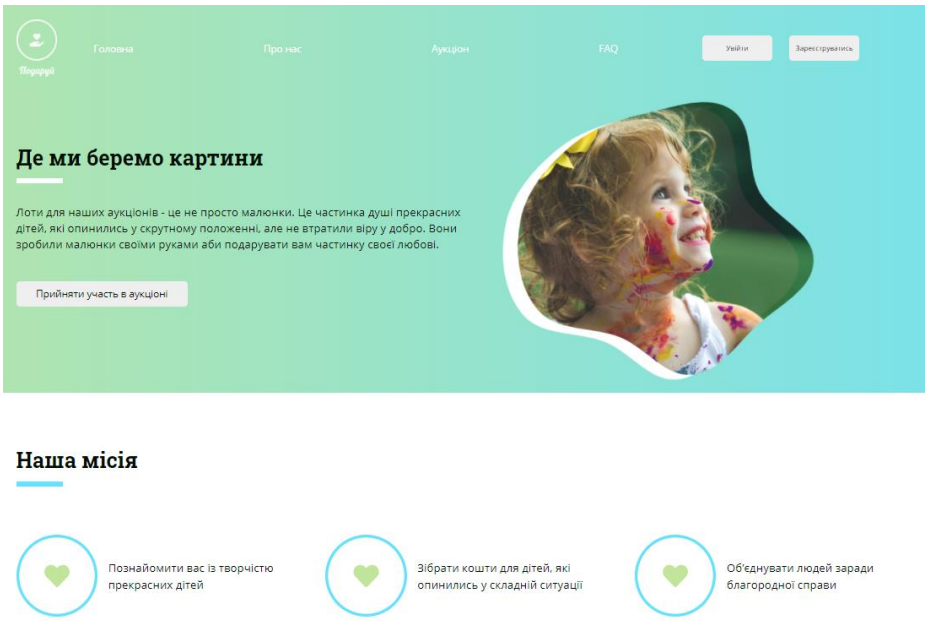

Рисунок 5.2 — Сторінка «Про нас»

— «Аукціон», де представлені всі лоти, доступні на даний момент.

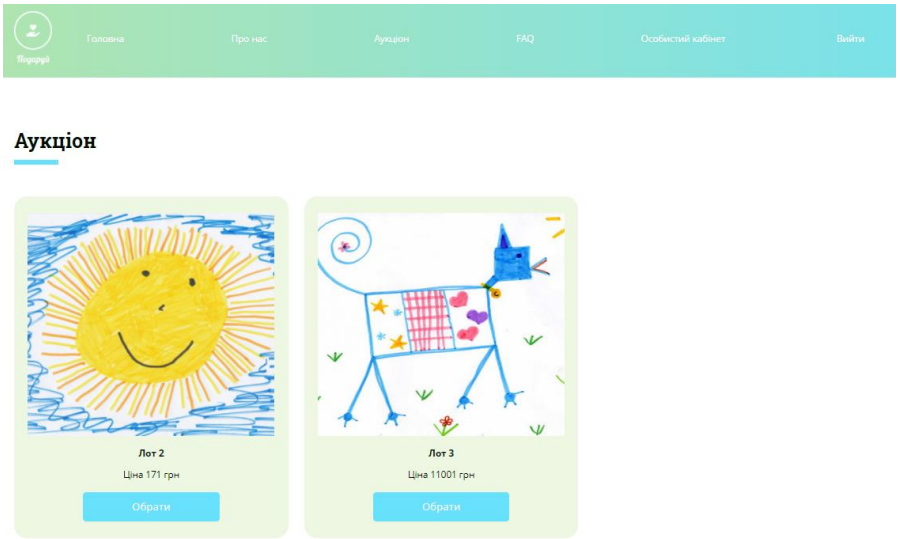

Рисунок 5.3 — Сторінка «Аукціон»

При натисканні кнопки «Обрати», що є на кожному лоті, ми можемо перейти на сторінку відповідного лоту, де в свою чергу можна поставити ставку (вписати суму у відповідне поле і натиснути кнопку «Зробити ставку») або купити лот за фіксованою ціною (натиснути кнопку «Купити»).

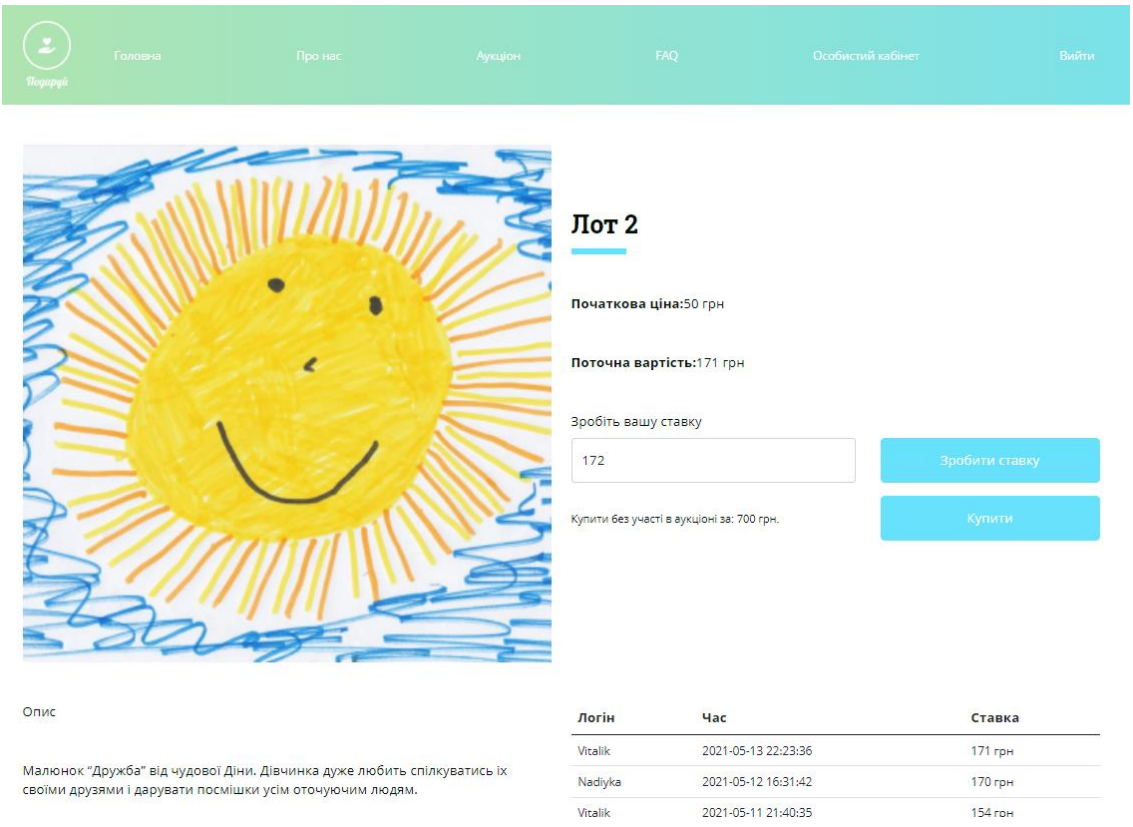

Рисунок 5.4 — Сторінка одного лота

На цій же сторінці відображається лічильник до кінця аукціону, коли запущено аукціон.

— «FAQ», де користувач по кліку на питання може побачити відповідь на нього.

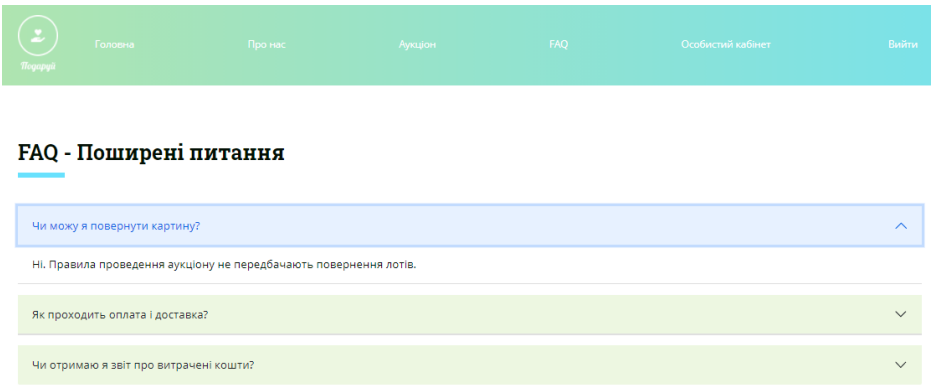

Рисунок 5.5 — Сторінка «Поширені питання»

— «Увійти» та «Зареєструватись» є стандартними та відповідають за можливості користувачів увійти у систему та зареєструватись.

Після того як користувач зареєструється та увійде в систему, йому стає доступною сторінка «Особистий кабінет», де він може переглянути свою інформацію, перейти до сторінки поповнення гаманця або сторінки пожертвування коштів, переглянути статистику своїх ставок та замовлення. Для користувачів із правами адміністратора передбачено кнопку входу в панель адміністратора.

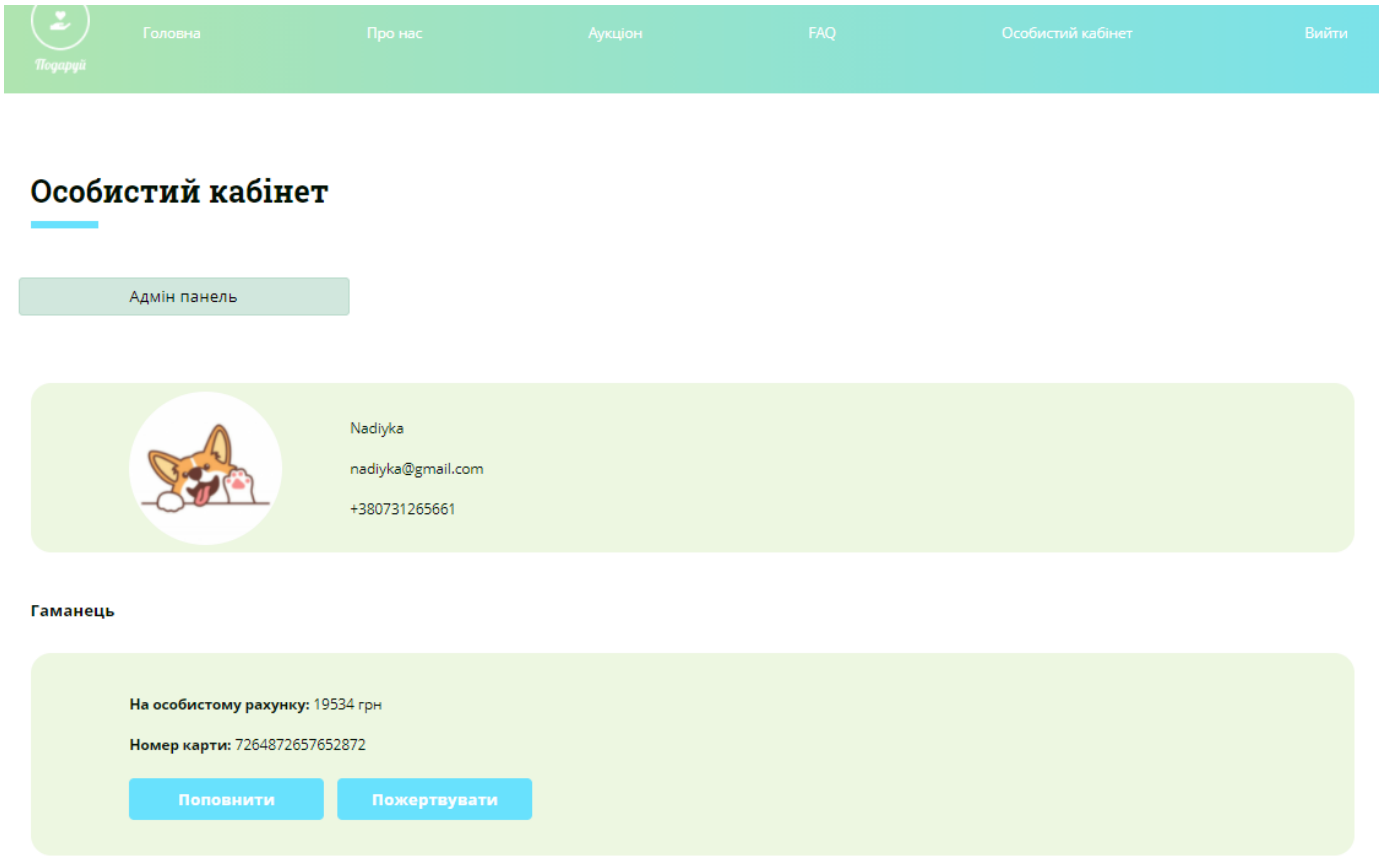

Рисунок 5.6 — Сторінка «Особистий кабінет»

При натисканні кнопки «Поповнити» користувач потрапляє на сторінку поповнення свого гаманця, де він має обрати платіжну систему та вписати суму поповнення.

Наразі дана функція не є повністю доступною до використання, бо для повної роботи треба підключати платіжні API, що у більшості випадків є небезкоштовним.

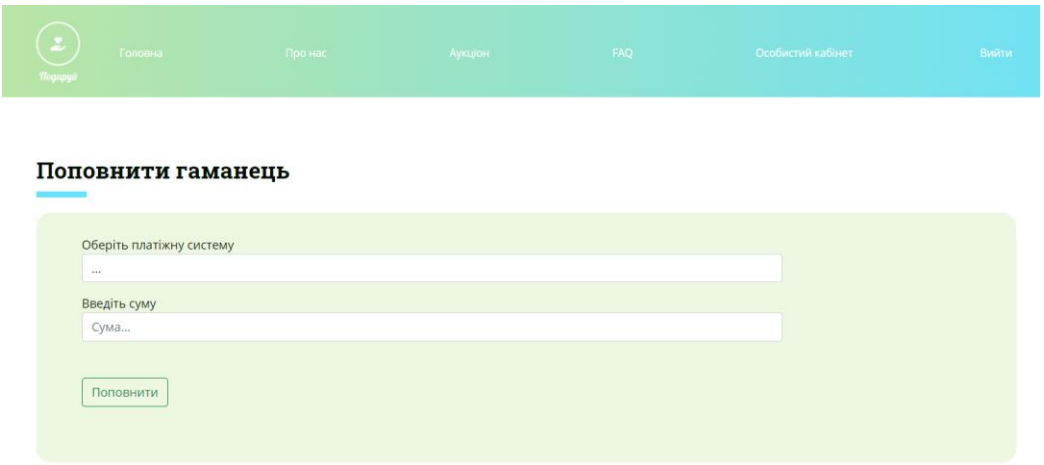

Рисунок 5.7 — Сторінка поповнення гаманця

На сторінці пожертвування навіть без реєстрації у веб-системі користувач може зробити благодійний внесок, ввівши у поля своє ім’я, номер карти та суму (рисунок 5.8).

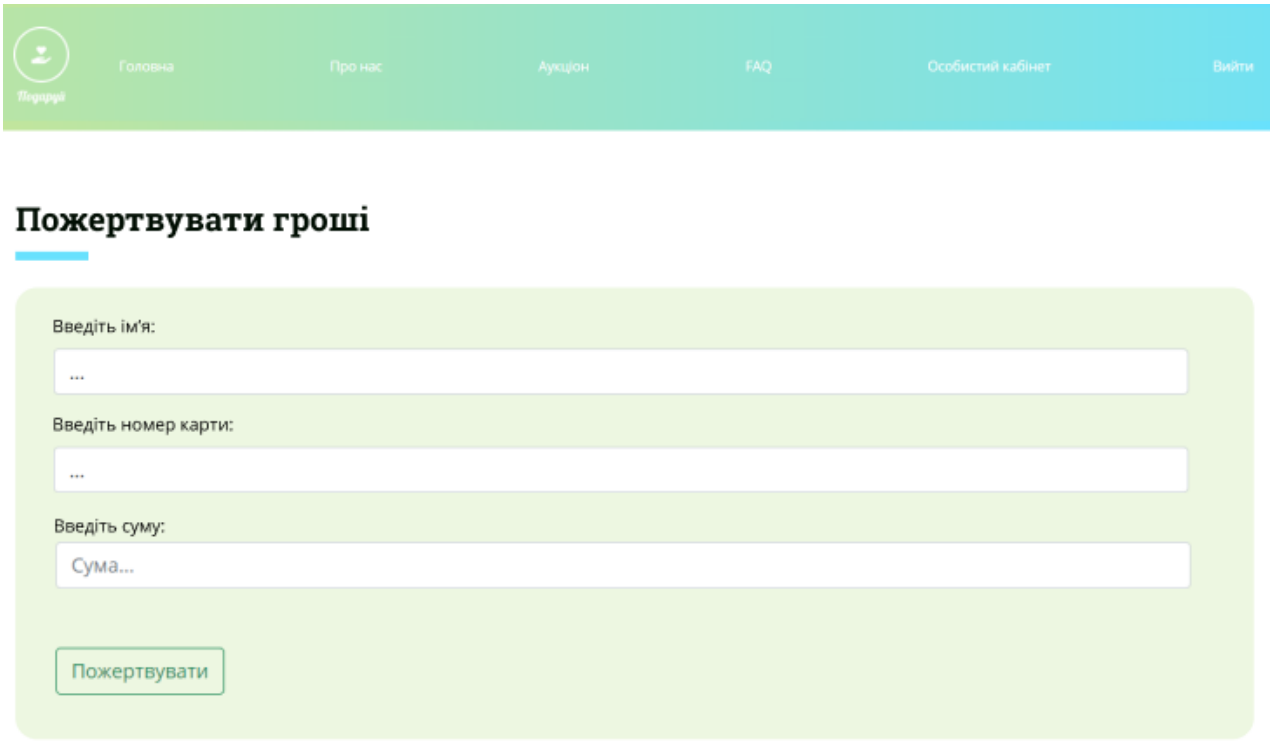

Рисунок 5.8 —Сторінка пожертвування коштів

5.3. Сценарій роботи адміністратора з системою

Адміністратор веб-системи може маніпулювати наповненням аукціону, ціллю

коштів на місяць та переглядати замовлення, не вдаючись до входу у базу даних.

Панель адміністратора наразі має такі сторінки та можливості:

— Списки лотів, серед яких можна окремо подивитись активні, неактивні та архів.

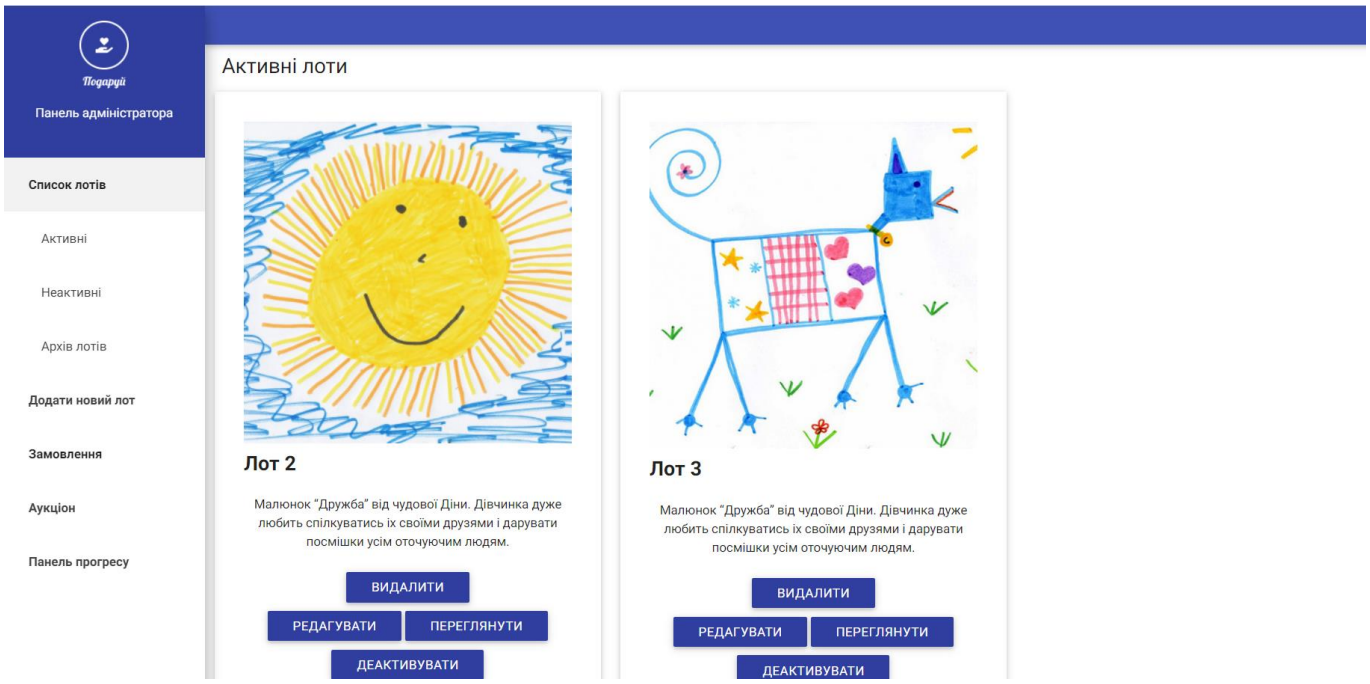

Рисунок 5.9 — Сторінка активних лотів

Тут адміністратор може переглянути активні лоти та провести деякі потрібні маніпуляції, наприклад, видалити лот зі списку.

Лот також можна відредагувати (рисунок 5.8).

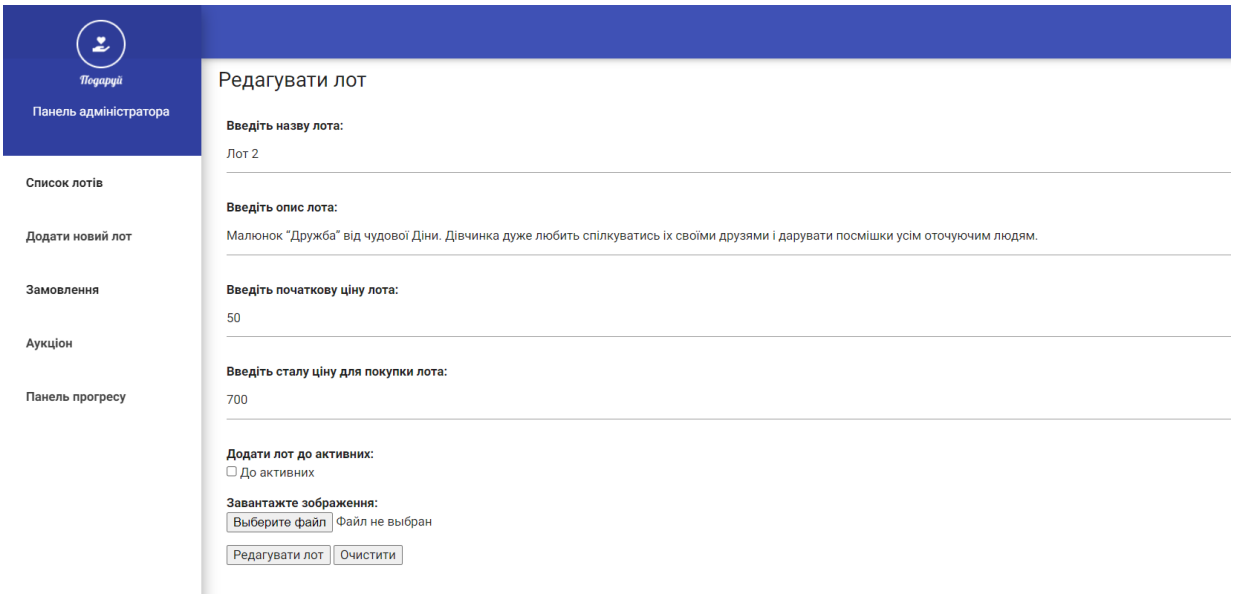

Рисунок 5.10 — Сторінка редагування лота

Сторінки активних та неактивних лотів є ідентичними, за винятком того, що на першій знаходиться кнопка «деактивувати», а на другій — «активувати».

Архів трохи відрізняється. Так як лоти, відображені на цій сторінці уже є купленими, то ми можемо їх тільки переглянути або видалити (рисунок 5.9).

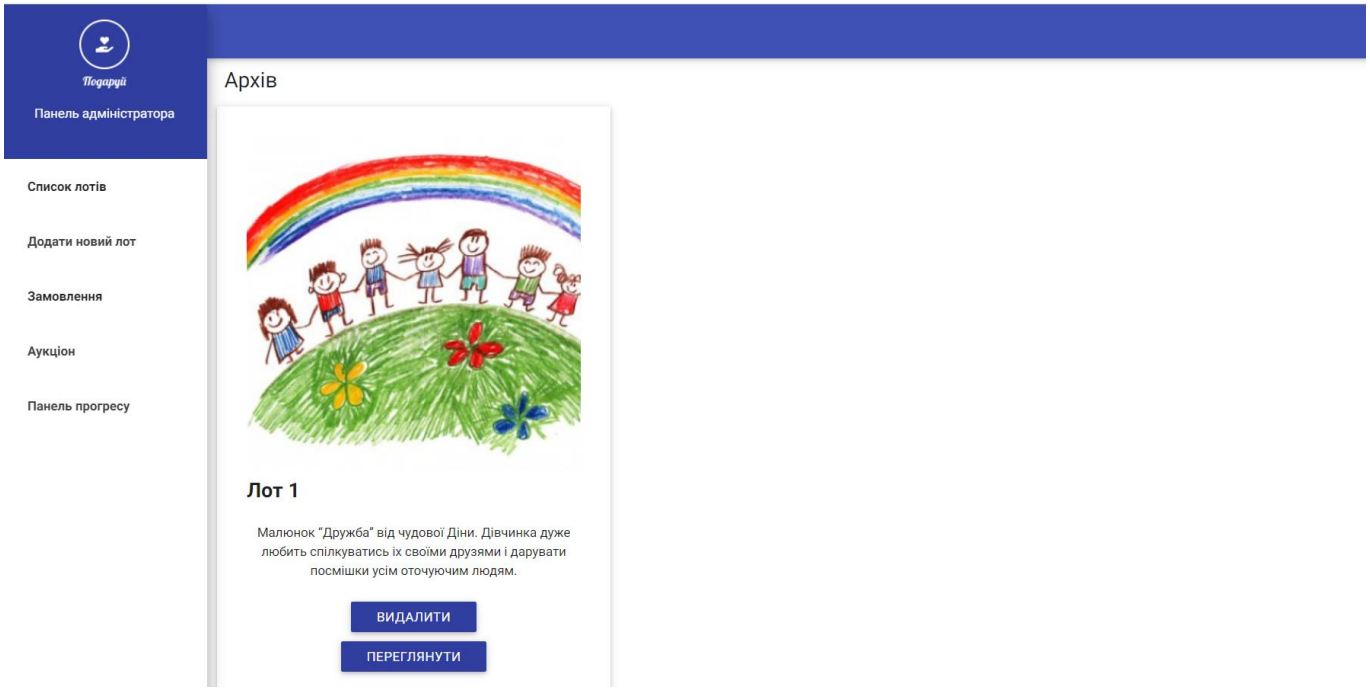

Рисунок 5.11 — Архів лотів

При натисканні кнопки «Переглянути», адміністратор потрапляє на сторінку перегляду усієї інформації про лот. Це потрібно для того, щоб адміністратор розумів який це саме лот, чи треба його редагувати тощо.

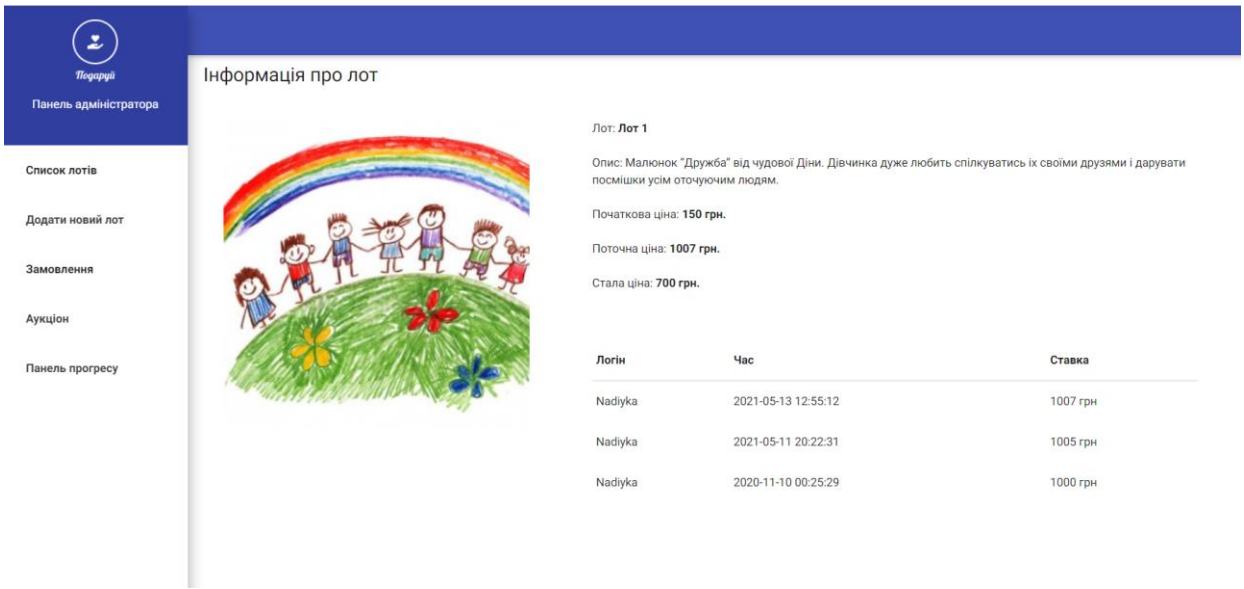

Рисунок 5.12 — Сторінка перегляду інформації про лот

— Сторінка «Додати новий лот» слугує для створення нових лотів для аукціону. Тут адміністратор може заповнити усю необхідну інформацію про лот, виставити ціну, завантажити зображення та виставити стан чек-бокса відповідного до того, чи треба цей лот одразу додати в існуючий аукціон.

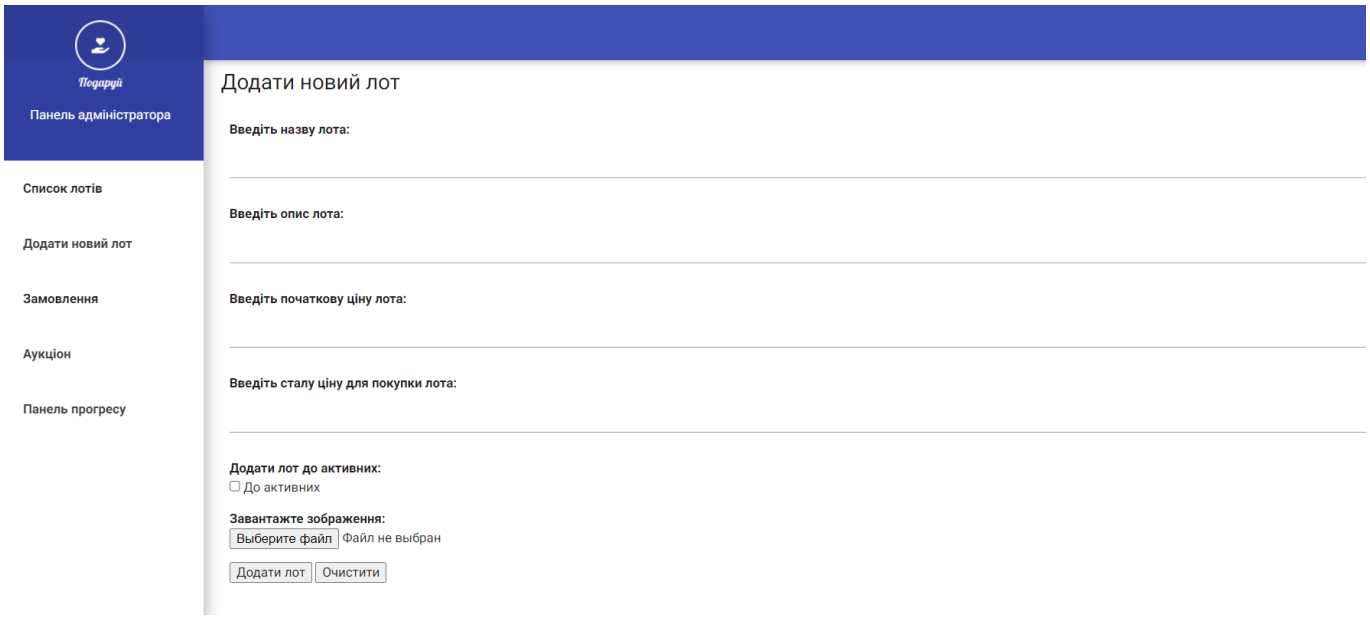

Рисунок 5.13 — Сторінка додавання нового лота

— Пункт меню «Замовлення» має три підпункти: «Виграні лоти» (відображаються переможці з найбільшою ставкою на лот), «За сталою ціною» (купівля за сталою ціною без участі в аукціоні) та «Пожертвування» (відображаються усі пожертвування).

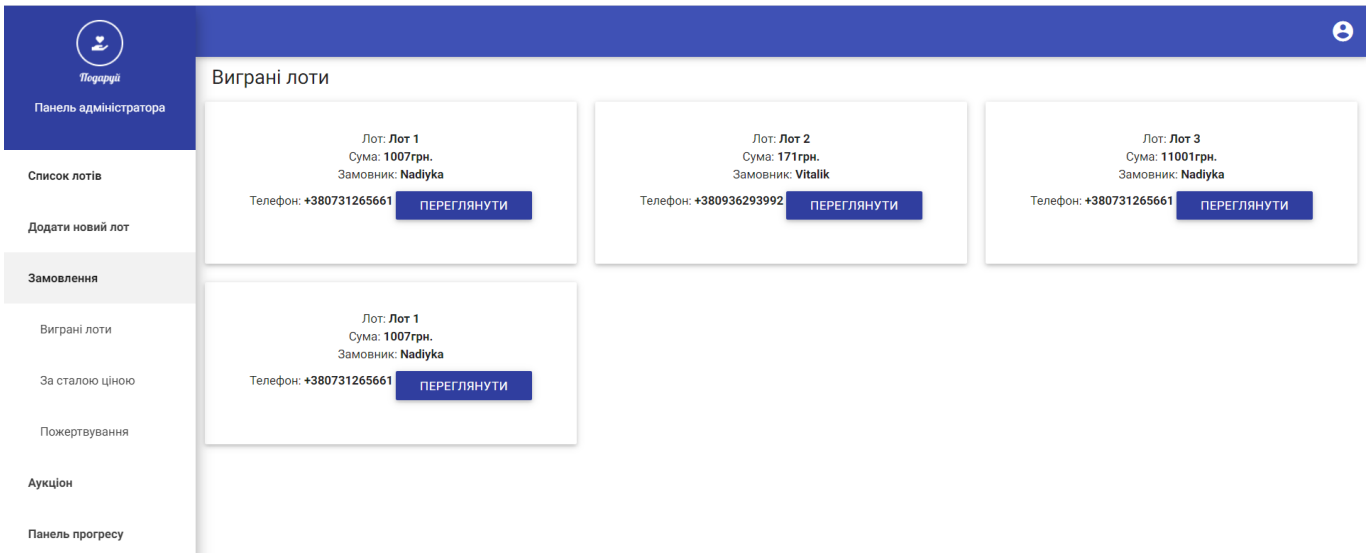

Рисунок 5.14 — Сторінка «Замовлення»-«Виграні лоти»

— На сторінці «Аукціон» адміністратор може налаштувати час завершення аукціону. Для цього йому потрібно обрати дату та час завершення за допомогою календаря. Починається аукціон за замовчуванням з того часу як час завершення є налаштованим та адміністратор натиснув «Почати аукціон». Також передбачено можливість передчасного завершення аукціону на випадок форс-мажорних ситуацій.

Рисунок 5.15 — Сторінка налаштування аукціону

— На рисунку 5.14 представлено сторінку «Панель прогресу», яка створена для зручного налаштування фінансової цілі місяця. Адміністратор має можливість написати суму коштів, яку у даному місяці заплановано забрати і натиснути на кнопку «Встановити ціль».

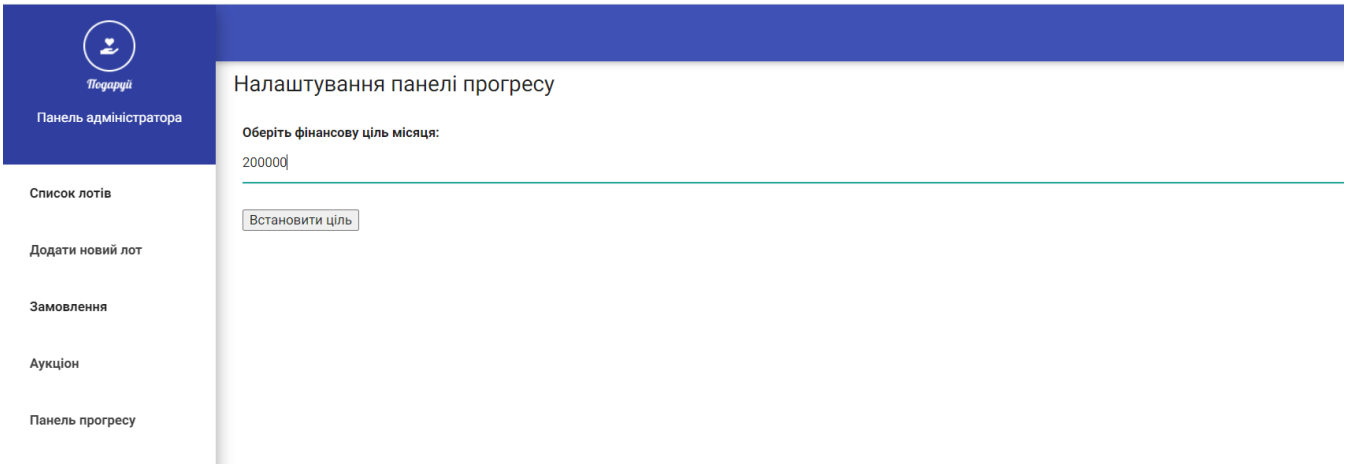

Рисунок 5.16 — Сторінка налаштування цілі місяця

Дана сума потім впливає на прогрес-бар, який присутній на головній сторінці та сторінці «Про нас» (рисунок 5.15).

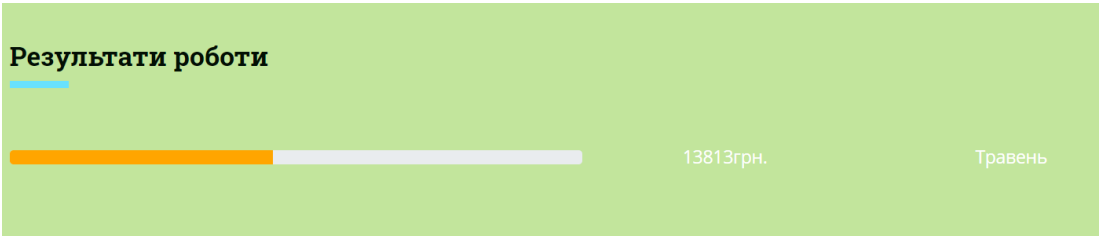

Рисунок 5.17 — Панель прогресу

Колір та положення будуть змінюватись у процентному співвідношенні, де сто відсотків — це сума встановлена в якості цілі на місяць. Ця панель зручна тим, що як користувачі веб-системи, так і організатори аукціону можуть оцінити наскільки вони вже наблизились до запланованої кількості грошей. У користувачів це також буде викликати певну довіру, адже вони будуть бачити, що інші люди також приймають участь в аукціонах та займаються благодійною роботою.

Отже, веб-система є досить широкою у своїй функціональності і з точки зору люди, яка хоче прийти на веб-сторінку та взяти участь у благодійному проекті, так і з точки зору адміністратора, що буде відповідальним за наповнення контентом та його редагування.

ВИСНОВКИ

Результатом даної дипломної роботи є розроблена система з організації благодійних аукціонів, на яких будуть продаватись дитячі малюнки. Дана система вирішує задачу автоматизації благодійної роботи.

Користувач системи може створити особистий кабінет та з гаманця робити ставки на обрані лоти. Після завершення аукціону обирається переможець, з яким далі зв’язується менеджер для уточнення усіх деталей.

У ході виконання роботи було:

1. Проведено аналіз питання благодійної роботи в Україні та існуючих систем відправлення коштів на допомогу. Виділено сильні та слабкі сторони систем.
2. Проаналізовано та обрано потужні та дієві технології реалізації системи.
3. Було створено діаграми прецедентів та класів, задля коректної побудови архітектури веб-системи, використовуючи засоби уніфікованої мови моделювання UML.
4. Продумано та створено візуальну частину.
5. Спроектовано та розроблено веб-систему для організації благодійних аукціонів із зрозумілим інтерфейсом та відповідним функціоналом.
6. Здійснено мануальне тестування, що допомогло виявити певні недоліки в системі і вчасно їх виправити.

Дана робота є актуальною і готовою до використання. За допомогою цього веб-застосунку уже можна влаштовувати аукціони, приймати в них участь та адмініструвати їх.

У подальшому є можливість розширення функціоналу, щоб сайт був більш клієнто-орієнтованим. Можлива співпраця з благодійними організаціями. У цьому випадку користувач зможе обрати до якого фонду він хоче відправити свою пожертву.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SQL [Електронний ресурс] — Режим доступу: https://www.zeluslugi.ru/info-czentr/it-glossary/term-sql.

2. Розвиток благодійності в Україні [Електронний ресурс] — Режим доступу: https://niss.gov.ua/doslidzhennya/gromadyanske-suspilstvo/rozvitok-blagodiynosti-v-ukraini.

3. PHPStorm [Електронний ресурс] — Режим доступу: https://www.jetbrains.com/ru-ru/phpstorm/.

4. Bootstrap documentation [Електронний ресурс] — Режим доступу: https://getbootstrap.com/docs/5.0/getting-started/introduction/.

5. Laravel documentation [Електронний ресурс] — Режим доступу: https://laravel.com/docs/8.x.

6. Основи CSS [Електронний ресурс] — Режим доступу: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/CSS_basics.

7. HTML [Електронний ресурс] — Режим доступу: https://developer.mozilla.org/ru/docs/Web/HTML.

8. MVC [Електронний ресурс] — Режим доступу: https://skillbox.ru/media/code/chto_takoe_mvc_bazovye_kontseptsii_i_primer_prilozheniya/

9. Figma [Електронний ресурс] — Режим доступу: https://www.figma.com/

10. Стаффер М. Laravel. Полное руководство. 2-е издание / Мэтт Стаффер. — Питер Пресс, 2020. — 512 с.

11. UML [Електронний ресурс] — Режим доступу: https://piter-soft.ru/knowledge/glossary/process/notatsiya-UML.html

ДОДАТОК А

Web-система з організації благодійних аукціонів

Специфікація

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТВ7165_21А

Аркушів 2

Київ – 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТВ7165_21Б	Записка Вахромова Н.О.ТВ-71.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТВ7165_21Б -1	Controllers	Оброблювач запитів до серверу
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТВ7165_21Б -2	Models	Представлення даних БД в коді
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТВ7165_21Б -3	Routes	Оголошення маршрутів

ДОДАТОК Б

Web-система з організації благодійних аукціонів

Лістинг програми

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТВ7165_21Б

Аркушів 18

Київ – 2021

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Auction;
use App\Models\Option;
use Carbon\Carbon;
use Illuminate\Http\Request;

class CommonController extends Controller
{
    const DATETIME_FORMAT = 'Y-m-d\TH:i';
    const LARAVEL_FORMAT = 'Y-m-d H:i:s';
    const OUTPUT_FORMAT = 'd.m.Y H:i';

    public function index()
    {
        return view('admin.index');
    }

    public function indexStart()
    {
        $auctionEntity = Auction::latest()->first();

        $auction = null;

        if ($auctionEntity) {
            $auction = [];
            $auction['created_at'] = Carbon::createFromFormat(self::LARAVEL_FORMAT,
$auctionEntity->created_at)->format(self::OUTPUT_FORMAT);
            $auction['end_time'] = Carbon::createFromFormat(self::LARAVEL_FORMAT,
$auctionEntity->end_time)->format(self::OUTPUT_FORMAT);
        }

        return view('admin.start_auction', ['auction' => $auction]);
    }

    public function indexProgressBar()
    {
        $goal = Option::where('name', 'goal')->first()->value;

        return view('admin.progress_bar_settings', ['goal' => $goal]);
```

```
    }

    public function startAuction(Request $request)
    {
        $request->validate([
            'end_time' => 'required|date|after:now|date_format:' . self::DATETIME_FORMAT,
        ]);

        $endTime = Carbon::createFromFormat(self::DATETIME_FORMAT, $request->end_time);

        $auction = new Auction();
        $auction->end_time = $endTime->format(self::LARAVEL_FORMAT);

        $auction->save();

        return back()->with('message', 'Аукціон успішно розпочато!');
    }

    public function setProgressOptions(Request $request)
    {
        $request->validate([
            'goal' => 'required|int',
        ]);

        $option = Option::where('name', 'goal')->first();
        $option->value = $request->goal;
        $option->save();

        return back()->with('message', 'Ціль успішно змінено!');
    }
}
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Bet;
use App\Models\Lot;
use Illuminate\Http\Request;

class LotController extends Controller
{
    public function indexActive()
    {
```

```
    $lots = Lot::where('state', 'active')->get();

    return view('admin.active_lots', ['lots' => $lots]);
}

public function indexInactive()
{
    $lots = Lot::where('state', 'inactive')->get();

    return view('admin.not_active_lots', ['lots' => $lots]);
}

public function indexArchive()
{
    $lots = Lot::where('state', 'archive')->get();

    return view('admin.archive', ['lots' => $lots]);
}

public function showLot($id)
{
    $lot = Lot::find($id);
    $bets = Bet::where('lot_id', $lot->id)->orderBy('created_at', 'DESC')->get();

    return view('admin.lot_info', ['lot' => $lot, 'bets' => $bets]);
}

public function edit($id, Request $request)
{
    $lot = Lot::find($id);

    return view('admin.edit_lot_info', ['lot' => $lot]);
}

public function update($id, Request $request)
{
    $request->validate([
        'title' => 'required|string|max:255',
        'description' => 'required|string|max:255',
        'initial_price' => 'required|integer',
        'fixed_price' => 'required|integer',
        'image' => 'image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        'activation' => 'bool'
    ]);
```

```
    $imageName = null;

    if (null !== $request->image) {
        $imageName = time() . '.' . $request->image->extension();
        $request->image->move(public_path('lots'), $imageName);
    }

    $lot = Lot::find($id);

    $lot->title = $request->title;
    $lot->description = $request->description;
    $lot->initial_price = $request->initial_price;
    $lot->fixed_price = $request->fixed_price;
    null === $imageName ?: $lot->image_name = $imageName;
    true != $request->activation ?: $lot->state = 'active';


    $lot->save();

    return back()->withInput()->with('message', 'Успішно оновлено!');
}

public function deactivate($id)
{
    $lot = Lot::find($id);

    $lot->state = 'inactive';
    $lot->save();

    return back()->with('message', 'Успішно деактивовано!');
}

public function activate($id)
{
    $lot = Lot::find($id);

    $lot->state = 'active';
    $lot->save();

    return back()->with('message', 'Успішно активовано!');
}

public function delete($id)
{
    $lot = Lot::find($id);
```

```
    $bets = $lot->bets;
    $lot->order->delete();

    foreach ($bets as $bet) {
        $bet->delete();
    }

    $lot->delete();

    return back()->with('message', 'Успішно видалено!');
}

public function create()
{
    return view('admin.add_new_lot');
}

public function store(Request $request)
{
    $request->validate([
        'title' => 'required|string|max:255',
        'description' => 'required|string|max:255',
        'initial_price' => 'required|integer',
        'fixed_price' => 'required|integer',
        'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        'activation' => 'bool'
    ]);

    $imageName = time() . '.' . $request->image->extension();
    $request->image->move(public_path('lots'), $imageName);

    $activation = true == $request->activation;

    $lot = new Lot();

    $lot->title = $request->title;
    $lot->description = $request->description;
    $lot->initial_price = $request->initial_price;
    $lot->current_bet = $request->initial_price;
    $lot->fixed_price = $request->fixed_price;
    $lot->image_name = $imageName;
    $lot->state = $activation ? 'active' : 'inactive';
```

```
        $lot->save();

        $redirectRoute = $activation ? 'active_lots' : 'inactive_lots';

        return redirect()->route($redirectRoute)->withInput()->with('message', 'Успішно
створено!');
    }
}
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Donation;
use App\Models\Order;

class OrderController extends Controller
{
    public function indexWon()
    {
        $orders = Order::where('status', 'won')->get();

        return view('admin.won_lots', ['orders' => $orders]);
    }

    public function indexSold()
    {
        $orders = Order::where('status', 'sold')->get();

        return view('admin.lots_fixed_price', ['orders' => $orders]);
    }

    public function showDonations()
    {
        $donations = Donation::all();

        return view('admin.donations', ['donations' => $donations]);
    }
}
<?php

namespace App\Http\Controllers;


use App\Http\Controllers\Admin\CommonController;
```

```
use App\Models\Auction;
use App\Models\Bet;
use App\Models\Donation;
use App\Models\Foundation;
use App\Models\Lot;
use App\Models\Order;
use App\Models\User;
use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Redirect;


class AuctionController extends Controller
{

    public function index()
    {
        $lots = Lot::where('state', 'active')->get();

        return view('auction.auction_page', ['lots' => $lots]);
    }

    public function lot($id)
    {
        $remainingTime = 0;
        $lot = Lot::find($id);
        $bets = Bet::where('lot_id', $lot->id)->orderBy('created_at', 'DESC')->get();
        $collection = Auction::all();
        if (1 === \count($collection)) {
            $auction = $collection->first();
            $remainingTime =
Carbon::createFromFormat(CommonController::LARAVEL_FORMAT, $auction-
>end_time)->diffInRealSeconds(Carbon::
            now());
        }

        return view('auction.lot_page', ['lot' => $lot, 'bets' => $bets, 'remainingTime' =>
$remainingTime]);
    }

    public function donate(Request $request)
    {
```

```
//        dd($request);

        $request->validate([
            'name' => 'required|string|max:255',
            'card_number' => 'required|numeric|digits:16',
            'sum' => 'required|integer|min:1',
        ]);

        $attributes = $request->all();
        unset($attributes['_token']);
        $donation = new Donation($attributes);
        $saved = $donation->save();

        if (!$saved) {
            return Redirect::back()->withErrors(['Неможливо виконати пожертвування!']);
        }

        return  Redirect::back()->with('message', "Дякую за ваше пожертвування!");
    }

    public function makeBet($id, Request $request): RedirectResponse
    {
        $lot = Lot::find($id);
        $wallet = User::find(Auth::id())->wallet;
        $amount = $request->amount;
        if ($lot->current_bet >= $amount) {

            return Redirect::back()->withErrors(['Занадто маленька ставка!']);
        }
        if ($amount > $wallet->amount) {
            return Redirect::back()->withErrors(['Недостатньо коштів на балансі!']);
        }
        $wallet->amount = $wallet->amount - $amount;
        $wallet->save();

        $lot->current_bet = $amount;
        $lot->save();

        $bet = new Bet();
        $bet->setAttribute('lot_id', $id);
        $bet->setAttribute('user_id', Auth::id());
        $bet->setAttribute('amount', $amount);
        $bet->save();

        return Redirect::back()->with('message', "Дякую за вашу ставку!");
```

```
}

public function showCharity()
{
   $foundations = array_column(Foundation::all('name')->toArray(), 'name');

   return view('auction.charity', ['foundations' => $foundations]);
}

public function endAuction()
{
   $this->finish();

   return redirect(route('start_auction'))->with('message', 'Аукціон завершено!');
}

public function finish(): void
{
   $lots = Lot::getAllActiveWithBets();

   foreach ($lots as $lot) {
      $highestBet = $lot->getHighestBet();

      if (null === $highestBet) {
         $lot->state = 'conflict';
         $lot->save();
         continue;
      }

      $user = $highestBet->user;

      $order = new Order();
      $order->user_id = $user->id;
      $order->lot_id = $lot->id;
      $order->status = 'won';
      $order->sum = $highestBet->amount;
      $order->save();

      $lot->winner_id = $user->id;
      $lot->state = 'archive';
      $lot->save();
   }

   $lotsWithoutBets = Lot::getAllActiveWithoutBets();
```

```
        foreach ($lotsWithoutBets as $lot) {
            $lot->state = 'inactive';

            $lot->save();
        }

        $auctions = Auction::all();
        foreach ($auctions as $auction) {
            $auction->delete();
        }
    }

    public function buyLot($id)
    {
        $lot = Lot::find($id);
        $wallet = User::find(Auth::id())->wallet;

        if ($lot->fixed_price > $wallet->amount) {
            return Redirect::back()->withErrors(['Недостатньо коштів на балансі!']);
        }

        $wallet->amount = $wallet->amount - $lot->fixed_price;
        $wallet->save();

        $order = new Order();
        $order->user_id = Auth::id();
        $order->lot_id = $lot->id;
        $order->status = 'sold';
        $order->sum = $lot->fixed_price;
        $order->save();

        $lot->state = 'archive';
        $lot->save();

        return redirect(route('auction_page'))->with('message', 'Дякую за покупку!');
    }
}
<?php
namespace App\Http\Controllers;


use App\Models\Bet;
use App\Models\User;
use Carbon\Carbon;
use DateTime;
```

```
use Illuminate\Database\Eloquent\Collection;

class MainController extends Controller
{
    public static $months = array(1 => 'Січень', 'Лютий', 'Березень', 'Квітень', 'Травень',
'Червень', 'Липень', 'Серпень', 'Вересень', 'Жовтень', 'Листопад', 'Грудень');


    public function index()
    {
        $bets = Bet::getBetsByLastMonths(5);
        $amountsByMonths = self::getSumOfBetsByLastMonths($bets);
        $userAmounts = Bet::groupBy('user_id')->selectRaw('user_id, sum(amount) as
sum')->orderBy('sum', 'DESC')->limit(8)->get()->toArray();

        $bestBenefactors = array_map(function ($userAmount) {
            return User::find($userAmount['user_id']);
        }, $userAmounts);

        return view('auction.main', ['amountsByMonths' => $amountsByMonths,
'bestBenefactors' => $bestBenefactors]);
    }

    public static function getSumOfBetsByLastMonths(Collection $bets): array
    {
        $amountByMonth = [];

        foreach ($bets as $bet) {
            if (!isset($amountByMonth[self::$months[Carbon::parse($bet->created_at)-
>month]]))
                $amountByMonth[self::$months[Carbon::parse($bet->created_at)->month]] = 0;
            $amountByMonth[self::$months[Carbon::parse($bet->created_at)->month]] +=
$bet->amount;
        }

        return $amountByMonth;
    }
}
```

```
<?php

namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

/**
 * App\Models\Bet
 *
 * @property int $id
 * @property int $user_id
 * @property int $lot_id
 * @property int $amount
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @property-read \App\Models\Lot $lot
 * @property-read \App\Models\User $user
 * @method static \Illuminate\Database\Eloquent\Builder|Bet newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Bet newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Bet query()
 * @method static \Illuminate\Database\Eloquent\Builder|Bet whereAmount($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Bet whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Bet whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Bet whereLotId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Bet whereUpdatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Bet whereUserId($value)
 * @method static where(string $string, $id)
 * @method static groupBy(string $string)
 * @mixin \Eloquent
 */
class Bet extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'lot_id',
        'amount',
        'created_at'
    ];

    public function user()
```

```
    {
        return $this->belongsTo(User::class);
    }

    public function lot()
    {
        return $this->belongsTo(Lot::class);
    }

    public static function getBetsByLastMonths(int $months)
    {
        return Bet::where('created_at', '>=', Carbon::now()->subMonths($months - 1))-
>get(['amount', 'created_at']);
    }
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Donation extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'card_number',
        'sum',
    ];
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Auction extends Model
{
    use HasFactory;
}
<?php
```

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Foundation extends Model
{
    use HasFactory;
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

/**
 * App\Models\Lot
 *
 * @property int $id
 * @property string $title
 * @property int $initial_price
 * @property int $current_bet
 * @property string $description
 * @property string $image_name
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @method static \Illuminate\Database\Eloquent\Builder|Lot newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Lot newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Lot query()
 * @method static \Illuminate\Database\Eloquent\Builder|Lot whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lot whereCurrentBet($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lot whereDescription($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lot whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lot whereImageName($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lot whereInitialPrice($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lot whereTitle($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Lot whereUpdatedAt($value)
 * @method static find($id)
 * @mixin \Eloquent
 */
class Lot extends Model
{
    use HasFactory;
```

```
    public static function getAllActiveWithoutBets()
    {
        return Lot::where('state', '=', 'active')
            ->whereRaw('initial_price = current_bet')
            ->get();
    }

    public function bets()
    {
        return $this->hasMany(Bet::class);
    }

    public function order()
    {
        return $this->hasOne(Order::class);
    }

    public static function getAllActiveWithBets()
    {
        return Lot::where('state', '=', 'active')
            ->whereRaw('initial_price != current_bet')
            ->get();
    }

    public function getHighestBet()
    {
        return $this->bets()->orderBy('amount', 'DESC')->first();
    }
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

/**
 * App\Models\Order
 *
 * @property int $id
 * @property int $user_id
 * @property int $lot_id
 * @property string $status
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
```

```
 * @method static \Illuminate\Database\Eloquent\Builder|Order newModelQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Order newQuery()
 * @method static \Illuminate\Database\Eloquent\Builder|Order query()
 * @method static \Illuminate\Database\Eloquent\Builder|Order whereCreatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Order whereId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Order whereLotId($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Order whereStatus($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Order whereUpdatedAt($value)
 * @method static \Illuminate\Database\Eloquent\Builder|Order whereUserId($value)
 * @mixin \Eloquent
 */
class Order extends Model
{
    use HasFactory;

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function lot()
    {
        return $this->belongsTo(Lot::class);
    }
}
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

/**
 * @method static find(int|string|null $id)
 */
class User extends Authenticatable
{
    use HasFactory, Notifiable;

    protected $fillable = [
        'name',
        'email',
        'password',
```

```
        'image_name',
        'phone_number',
        'wallet_id'
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    public function wallet()
    {
        return $this->belongsTo(Wallet::class);
    }

    public function bets()
    {
        return $this->hasMany(Bet::class);
    }

    public function hasRole()
    {
        return $this->role === 'admin';
    }

    public function orders()
    {
        return $this->hasMany(Order::class);
    }
}
```

```
<?php

use App\Http\Controllers\AboutController;
use App\Http\Controllers\Admin\CommonController;
use App\Http\Controllers\Admin\LotController;
use App\Http\Controllers\Admin\OrderController;
use App\Http\Controllers\AuctionController;
use App\Http\Controllers\FaqController;
use App\Http\Controllers\MainController;
use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', [MainController::class, 'index'])->name('main_page');
Route::get('/about_us', [AboutController::class, 'index'])->name('about_us');
Route::get('/faq', [FaqController::class, 'index'])->name('faq');
Route::get('/profile', [ProfileController::class, 'index'])->name('profile');
Route::get('/auction', [AuctionController::class, 'index'])->name('auction_page');
Route::get('/lot/{id}', [AuctionController::class, 'lot'])->name('lot');
Route::get('/charity', [AuctionController::class, 'showCharity'])->name('charity_page');
Route::get('/top_up', [ProfileController::class, 'showTopUp'])->name('top_up_page');
Route::get('/confirm_auction_end', function () {
    return view('auction.confirmation', ['title' => 'Confirmation', 'link' => 'end_auction', 'text'
=> 'Підтвердіть закінчення аукціону!']);
})->name('confirm_auction_name')->middleware(['auth','role:admin']);
Route::post('/lot/{id}/makeBet', [AuctionController::class, 'makeBet'])->name('makeBet')-
>middleware('auth');
Route::post('/end_auction', [AuctionController::class, 'endAuction'])-
>name('end_auction')->middleware(['auth', 'role:admin']);
Route::post('/donate', [AuctionController::class, 'donate'])->name('donate');

//admin
Route::prefix('admin')->middleware(['auth', 'role:admin'])->group(function () {
    Route::get('', [CommonController::class, 'index'])->name('admin_index');
```

```
    Route::get('/lot/active', [LotController::class, 'indexActive'])->name('active_lots');
    Route::get('/lot/inactive', [LotController::class, 'indexInactive'])->name('inactive_lots');
    Route::get('/lot/create', [LotController::class, 'create'])->name('create_lot');
    Route::get('/lot/archive', [LotController::class, 'indexArchive'])->name('archive_lots');
    Route::post('/lot/store', [LotController::class, 'store'])->name('store_lot');
    Route::post('/lot/update/{id}', [LotController::class, 'update'])->name('update_lot');
    Route::get('/lot/{id}', [LotController::class, 'showLot'])->name('lot_info');
    Route::get('/lot/edit/{id}', [LotController::class, 'edit'])->name('edit_lot');
    Route::post('/lot/delete/{id}', [LotController::class, 'delete'])->name('delete_lot');
    Route::post('/lot/deactivate/{id}', [LotController::class, 'deactivate'])-
>name('deactivate_lot');
    Route::post('/lot/activate/{id}', [LotController::class, 'activate'])->name('activate_lot');

    Route::get('/order/won', [OrderController::class, 'indexWon'])->name('won_lots');
    Route::get('/order/sold', [OrderController::class, 'indexSold'])->name('sold_lots');
    Route::get('/donation', [OrderController::class, 'showDonations'])->name('donations');
    Route::get('/start-auction', [CommonController::class, 'indexStart'])-
>name('index_auction_start');
    Route::post('/start-auction', [CommonController::class, 'startAuction'])-
>name('start_auction');
    Route::get('/progress-options', [CommonController::class, 'indexProgressBar'])-
>name('index_progress_options');
    Route::post('/progress-options', [CommonController::class, 'setProgressOptions'])-
>name('set_progress_options');
    Route::post('lot/buy/{id}', [AuctionController::class, 'buyLot'])->name('buy_lot')-
>middleware('auth');
});


require __DIR__.'/auth.php';
```

ДОДАТОК В

Web-система з організації благодійних аукціонів

Опис програмного коду

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ7165_21В

Аркушів 9

Київ – 2021 року

АННОТАЦІЯ

Додаток містить опис основних структурних елементів програмного коду веб-системи з проведення благодійних аукціонів. Розглянуті компоненти надають функціональні можливості для авторизації та реєстрації користувачів у системі, прийняття участі в аукціоні, замовленні лота за сталою ціною, перегляду панелі прогресу, кращих благодійників та особистого профілю, редагування аукціону та лотів, додавання нових лотів, перегляд замовлень та пожертвувань.

Система розроблена мовою програмування PHP. Вона написана за архітектурним шаблоном MVC та має розділення у структурі коду на компоненти. У розробці системи використовувалася документація Laravel для забезпечення правильного підходу до організації компонентів та їх написання.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ 75

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ 76

ОПИС ЛОГІЧНОЇ СТРУКТУРИ..... 77

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ 78

ВИКЛИК І ЗАВАНТАЖЕННЯ 79

ВХІДНІ ТА ВИХІДНІ ДАНІ..... 80

ЗАГАЛЬНІ ВІДОМОСТІ

У даному додатку містить міститься опис програмної реалізації класів та файлів, що надають функціональність для проведення благодійного аукціону.

Код, який описується, міститься у додатку Б даної роботи. Система працює через клієнтський застосунок, що був написаний за допомогою мови PHP з використанням фреймворку Laravel. При виконанні дій у інтерфейсі користувача відбувається запит на серверну частину системи. Тобто для роботи з панеллю потрібен доступ до мережі інтернет. Сервер опрацьовує запит, збираючи потрібну інформацію та повертаючи її клієнту. Зберігання даних відбувається у базі даних MySQL.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений застосунок надає можливість користувачам бачити їх метрики по проходженні навчання на інформаційно-навчальному порталі.

Повна функціональність веб-системи включає в себе:

— Можливість переглянути лоти, доступні у даний момент, та прийняти участь в аукціоні, тобто зробити ставку на обраний лот.

— Реєстрація у веб-застосунку із створенням особистого кабінету користувача.

— Купити лот за сталою ціною без необхідності робити ставку.

— Пожертвувати гроші незареєстрованому користувачеві без участі в аукціоні.

— Надання визначеним користувачам ролі адміністратора з можливістю входу в панель адміністратора.

— Для адміністратора: можливість додавання, редагування та видалення лотів за допомогою зручного інтерфейсу.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

За архітектурним шаблоном MVC застосунок складається з трьох компонентів:

— контролери;

— моделі;

— представлення.

Контролери це оброблювачі маршрутів. Для оголошення маршрутів виділений окремий файл web.php. У ньому прописані всі маршрути, по яких може переміщатись користувач веб-застосунку.

Також контролери використовують для роботи моделі, що є представленням таблиць бази даних.

Головним компонентом логіки у серверній частині є модуль AuctionController, який відповідає за правильне проведення аукціону і виконує перевірку ставок, перевірку особистого залишку у гаманці користувача, знімає гроші з гаманця у випадку, якщо користувач купив лот або поставив ставку.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

При розробці використовувалося інтегроване середовище розробки (IDE) PHPStorm. Також для дотримання парадигми ООП в розробці було використано успадкування, інкапсуляція тощо.

Для зручної маніпуляції з даними використовувалися стандартні оператори та методи мови програмування PHP:

— foreach;

— for;

— arsort;

— count;

— date тощо.

Для зберігання постійних даних користувача використовується з’єднання з базою даних MySQL. Для отримання даних з БД використовувався SQLBuilder та потужності ORM.

До списку методів для збору даних з БД входять:

— table;

— where;

— first;

— select;

— update;

— save тощо.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для використання аналітичної панелі інформаційно навчального порталу потрібно мати всі необхідні файли на локальному комп’ютері та браузер. Для коректної роботи застосунку у браузері має бути ввімкнена підтримка JavaScript.

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідна інформація при роботі з веб-системою з організації благодійних аукціонів включає в себе логін та пароль користувача системи, але є можливість реєстрації нових користувачів через інтерфейс користувача. Для адміністратора потрібно мати спеціальну роль, інакше вхід в адмін-панель не буде здійснена.

Результатом і вихідними даними програми є участь користувача в аукціоні, перегляд даних у особистому кабінеті, покупка лотів. Для адміністратора — налаштування аукціону, списків лотів та панелі прогресу.