МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ТИМЧИШИН Василь Степанович

Репозиторій навчального контенту/ Repository of the learning content

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-41
В. С. Тимчишин
_____

Науковий керівник:
викладач ПОРПЛИЦЯ Н.П.
_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

РЕЗЮМЕ

Дипломна робота містить 91 сторінку, 19 таблиць, 46 рисунків, списку використаних джерел із 20 найменувань, 3 додатки.

Метою дипломної роботи є розробка інформаційного веб-сайту Петриківського дитячого будинку-інтернату.

Об'єктом дослідження є процес залучення благодійних внесків на допомогу вихованцям дитячих будинків.

Предметом дослідження є методи та засоби для розробки веб-орієнтованої програмної системи для дитячого будинку.

Одержані результати полягають у розробці спеціалізованої веб-орієнтованої програмної системи дитячого будинку, яка забезпечує можливість перегляду детальної інформації про дитячий будинок в режимі онлайн.

Ключові слова: дитячий будинок; веб-сайт; UML моделі; меценат; специфікація вимог.

SUMMARY

Thesis contains 91 pages, 19 tables, 46 figures, list of sources with 20 titles, 3 application.

The aim of the thesis is to develop an information website of the Petrykivskyi orphanage.

Object of research is the process of the attracting donations to help the orphans.

The subject of research are methods and tools for developing web-based software system of the orphans.

The resulting is creation web-based software system of the orphans. This system provides the ability to view the detailed information about the orphans online.

Keywords: orphans; website; UML models; nurser; requirement’s specification.

ЗМІСТ

ВСТУП

Актуальність задачі

Тема дитячих будинків є дуже популярною, тому що в останні роки в Україні катастрофічно зростає кількість дітей, які за певних обставин лишаються поза сімейним вихованням, позбавляються батьківської опіки. Заклади, де живуть та розвиваються *діти* з *обмеженими можливостями*, мають багато труднощів, з якими стикаються кожного дня.

Мета і задачі розробки

Ціллю даного проекту є популяризація дитячого будинку в мережі Інтернет.

Об’єкт дослідження

Об'єктом дослідження є процес залучення благодійних внесків на допомогу вихованцям дитячих будинків.

Предмет дослідження

Предметом дослідження є методи та засоби для розробки веб-орієнтованої програмної системи для дитячого будинку.

Практична цінність

Проект передбачає розробку веб-орієнтованої системи, яка надає можливість отримати інформацію про дитячий будинок.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Коротка характеристика об'єкта управління

Об’єктом управління в моїй дипломній роботі став Петриківський обласний комунальний дитячий будинок-інтернат. Це заклад, в якому виховуються, навчаються та проходять соціальну реабілітацію діти з вадами розвитку.

При інтернаті працюють гуртки художньої самодіяльності, "Умілі руки” (вишивка, бісероплетіння, шиття), діє танцювальний ансамбль та ансамбль пісні.

У вказаному закладі перебувають дівчатка з розумовими та фізичними вадами, переважно інваліди II групи. На кожного підопічного ведеться історія хвороби та амбулаторна карта, складені МСЕК та дитячим ЛКК індивідуальні програми реабілітації інвалідів (ІПРІ).

Через те, що дана установа не є великою, тому інформація про неї доступна небагатьом. Тому розробка та впровадження такої програмної системи дасть можливість користувачам мережі Інтернет дізнатись більше про цей заклад. Водночас, необхідно провести огляд та аналіз систем-аналогів, щоб скористатись їхніми перевагами та уникнути недоліків. Головна мета веб-сайту – надати необхідну інформацію про дитячий будинок.

1.2 Опис предметної області

Для належної організації роботи системи необхідне виконання таких основних бізнес-процесів:

- перегляд основної інформації;
- авторизація адміністратора в системі;
- управління даними веб-сайту.

На рисунку 1.1 проілюстровано організацію роботи описаних вище процесів.

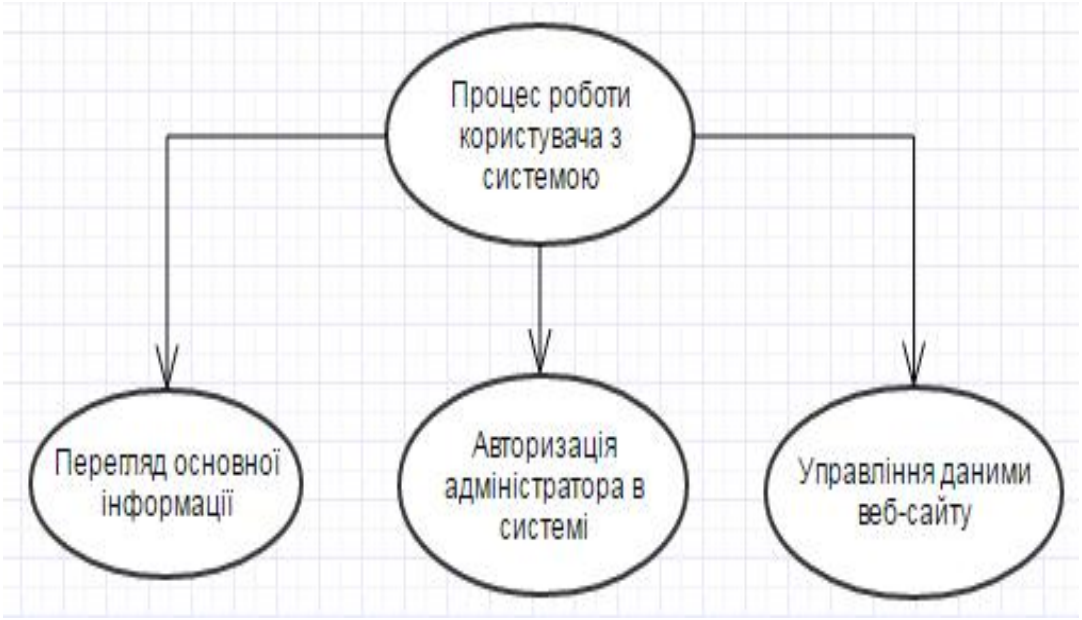

Рисунок 1.1 – Діаграма бізнес-процесів розроблюваного програмного продукту

Тепер проаналізуємо детальніше кожен з наведених бізнес-процесів.

На рисунку 1.2 зображено діаграму функцій процесу перегляду основної інформації.

Рисунок 1.2 – Діаграма функцій процесу перегляду основної інформації

Для того, щоб користувач мав можливість переглянути інформацію, яка його цікавить, він повинен виконати два простих кроки:

1) зайти на веб-сайт дитячого будинку;
2) вибрати мову перегляду даного контенту.

Характеристику даного бізнес-процесу наведено в таблиці 1.1.

Таблиця 1.1

Характеристика бізнес- процесу перегляду основної інформації

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Перегляд основної інформації
Основні учасники	Клієнт
Вхідна подія	Перехід на головну сторінку веб-сайту
Вхідні документи	-
Вихідна подія	Користувач переглянув інформацію, яка його цікавить
Вихідні документи	-
Клієнт бізнес-процесу	Користувач

На рисунку 1.3 подано діаграму функцій процесу авторизації адміністратора в системі.

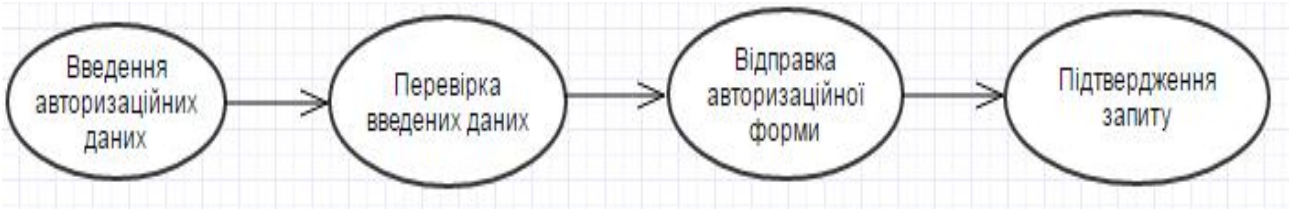

Рисунок 1.3 – Діаграма функцій процесу авторизації користувача

Для того, щоб зайти в свій аккаунт у системі, адміністратор повинен заповнити форму авторизації, а саме два її поля: «Логін», «Пароль». Після цього авторизований адміністратор буде мати доступ до свого кабінету. При неправильному введені логіна або пароля адміністратору буде надано право повторно ввести авторизаційні дані.

Характеристику бізнес-процесу авторизації користувача в системі наведено в таблиці 1.2.

Таблиця 1.2

Характеристика бізнес-процесу авторизації адміністратора в системі

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Авторизація адміністратора в системі
Основні учасники	Адміністратор
Вхідна подія	Перехід на сторінку авторизації
Вхідні документи	Дані з авторизаційної форми
Вихідна подія	Вхід в особистий кабінет
Вихідні документи	-
Клієнт бізнес-процесу	Адміністратор

На рисунку 1.4 зображено діаграму функцій процесу управління даними в системі.

Рисунок 1.4 – Діаграма функцій процесу управління даними в системі

Як видно з рисунку 1.4 процес управління даними складається з трьох функції:

1) додавання нових записів;
2) видалення застарілої інформації;
3) редагування потрібної інформації.

Дані функції будуть доступні тільки адміністратору через адмінпанель веб-сайту.

Характеристику бізнес-процесу реєстрації адміністратора в системі наведено в таблиці 1.3

Таблиця 1.3

Характеристика бізнес-процесу реєстрації адміністратора в системі

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Управління даними веб-сайту
Основні учасники	Адміністратор
Вхідна подія	Перехід на сторінку управління даними.
Вхідні документи	-
Вихідна подія	Виконані дії над потрібними даними
Вихідні документи	-
Клієнт бізнес-процесу	Адміністратор

1.3 Огляд і аналіз існуючих аналогів

Основною метою проведення аналізу існуючих аналогів є виявлення недоліків, порівняння можливостей та інтерфейсу.

Для проведення огляду та аналізу існуючих аналогів було обрано наступні програмні системи: «Pchelka», «Тернопільський дитячий будинок», «Харківський дитячий будинок «Родина»». Не зважаючи на те, що усі зазначені аналоги розроблені різними софтверними компаніями, але призначення у них одне: розповісти якомога більшому числу користувачів мережі Інтернет про дитячі будинки.

Розглянемо детальніше веб-сайт «Pchelka». Благодійний фонд «Бджілка» здійснює свою діяльність з лютого 2008 року. Фонд надає адресну допомогу дітям з важкими патологіями, закуповує дорогі препарати, оплачує операції і лікування.

На рисунку 1.5 проілюстровано загальний вигляд головної сторінки веб-сайту.

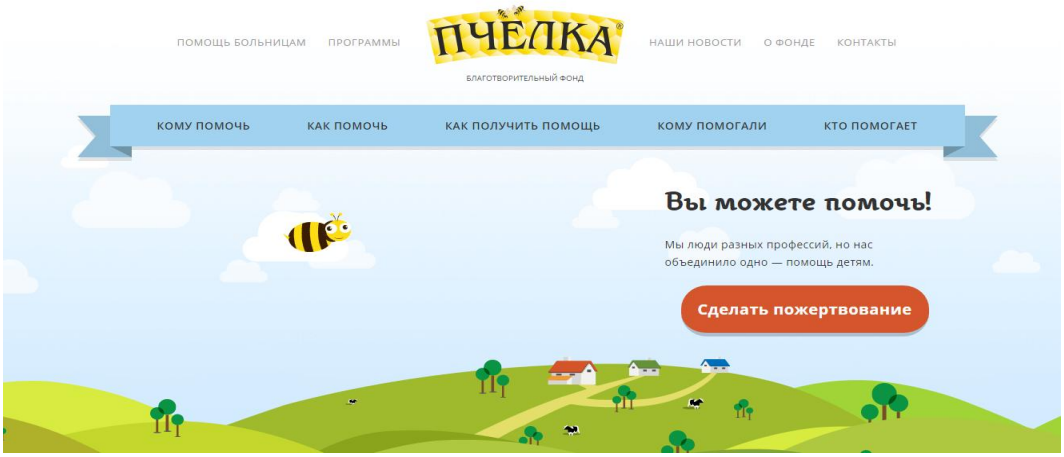

Рисунок 1.5 – Головна сторінка веб-сайту «Pchelka»

Як видно з рисунку 1.5, інтерфейс системи є інтуїтивно зрозумілим. Адже, навіть якщо ви – новий користувач, то труднощів при пошуку потрібної інформації не буде.

У верхній частині сторінки знаходиться навігаційне меню, яке містить посилання на такі сторінки: «Кому допомогти», «Як допомогти», «Як отримати допомогу», «Кому допомагали», «Хто допомагає». Праворуч знаходиться кнопка «Зробити пожертву». При натисканні на неї перед користувачем з’являється сторінка, яка зображена на рисунку 1.6.

Рисунок 1.6 – Сторінка «Як допомогти»

Як видно з вищенаведеного рисунку надати допомогу благодійному фонду можна різними способами, а саме:

1) грошима;
2) речами;
3) інформаційною підтримкою;
4) стати донором;
5) стати волонтером;
6) іншим шляхом.

При виборі першого способу перед користувачем відкривається сторінка, яка зображена на рисунку 1.7.

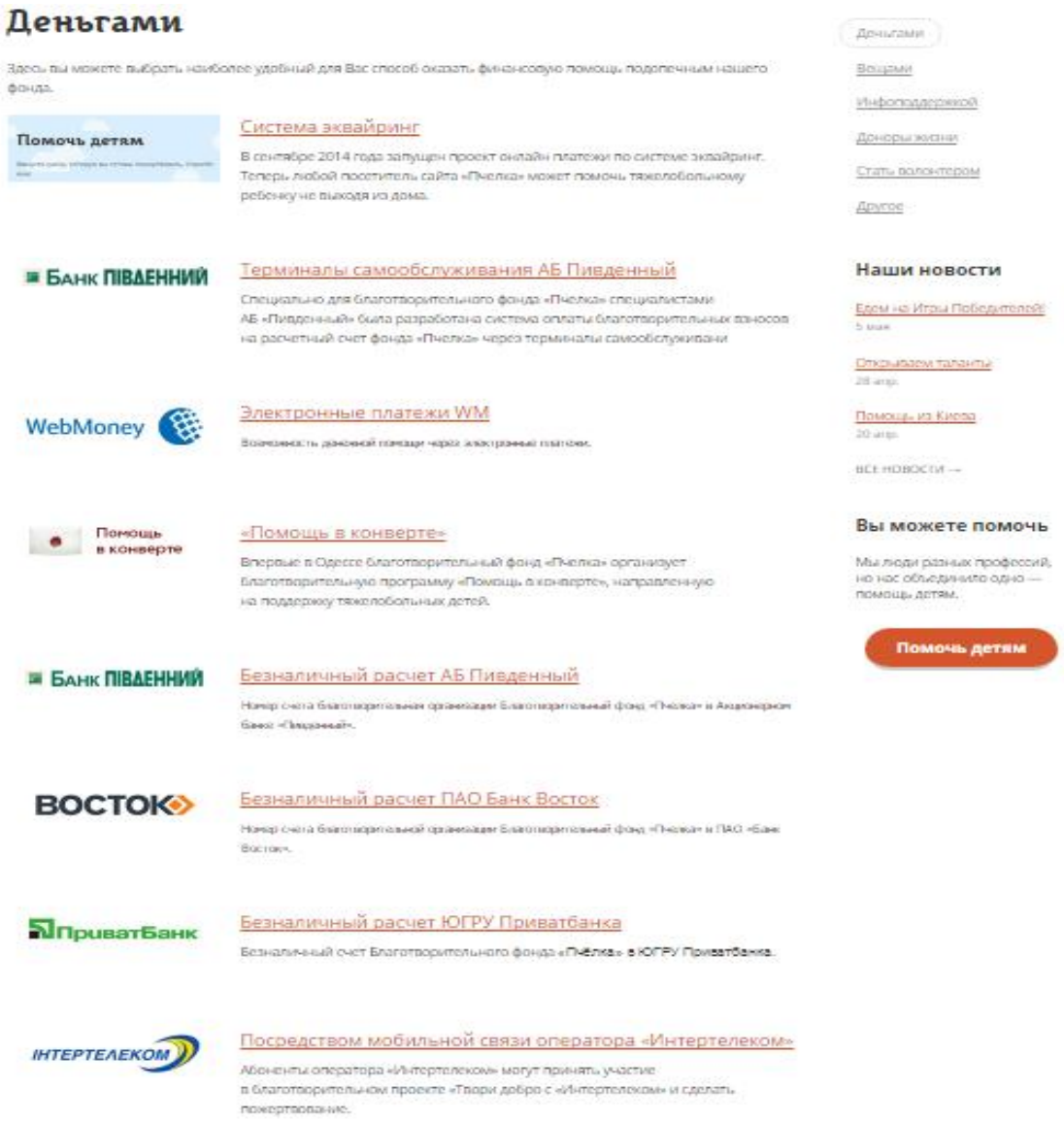

Рисунок 1.7 – Сторінка, на якій користувач може вибрати зручний спосіб оплати

Як видно з рисунку 1.7 на сторінці перераховано всі можливі способи дистанційної оплати. Користувач має можливість вибрати найбільш зручний для нього спосіб переказу коштів.

При виборі всіх інших способів допомоги користувачу буде надано детальний алгоритм дій, відповідно до того варіанту, який він вибрав.

Коли користувач повернеться на головну сторінку та прокрутить її вміст перед ним з'являться останні новини благодійного фонду (рисунок 1.8).

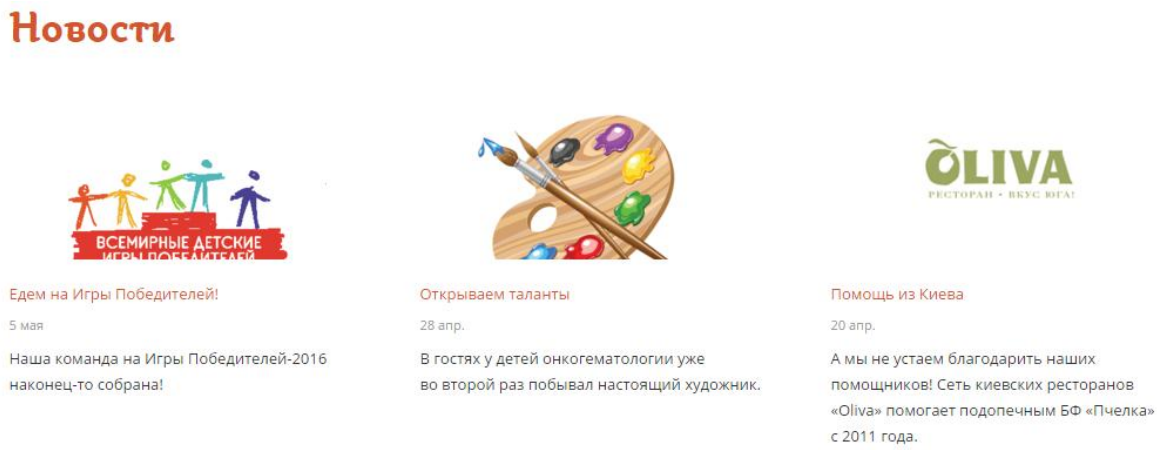

Рисунок 1.8 – Новини фонду

Гортаючи далі, користувач побачить фотографії тих дітей, які потребують допомоги. Результат цієї дії зображений на рисунку1.9. Під фото розміщено загальну інформацію: прізвище, ім'я, та яка сума необхідна для дитини.

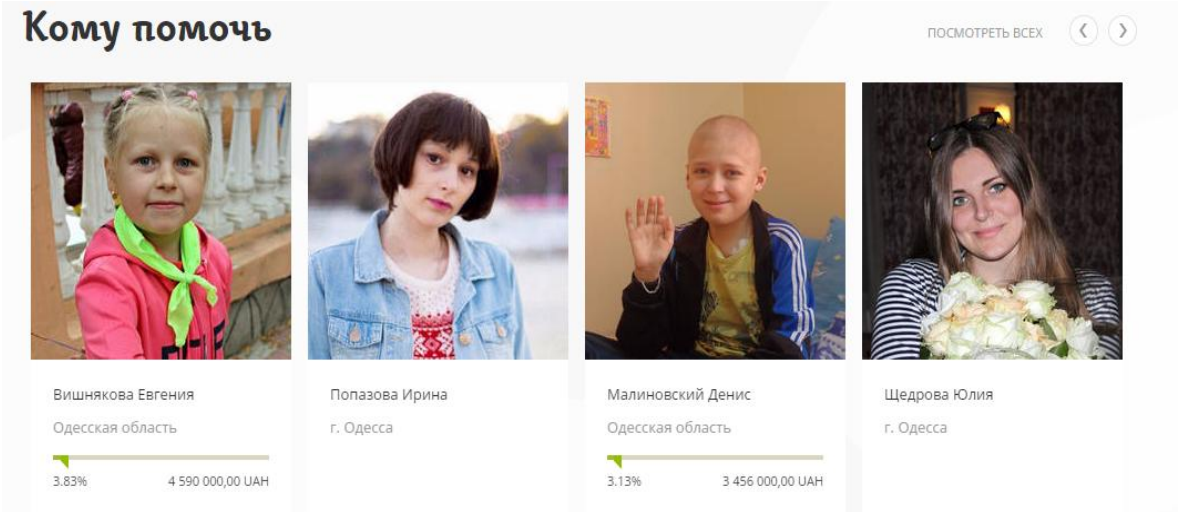

Рисунок 1.9 – Інформація про дітей, які потребують допомоги

При кліку на фотографію конкретної дитини із слайдера перед користувачем відкриється нова сторінка із повною інформацією про дитину. Результат цієї дії зображений на рисунку 1.10.

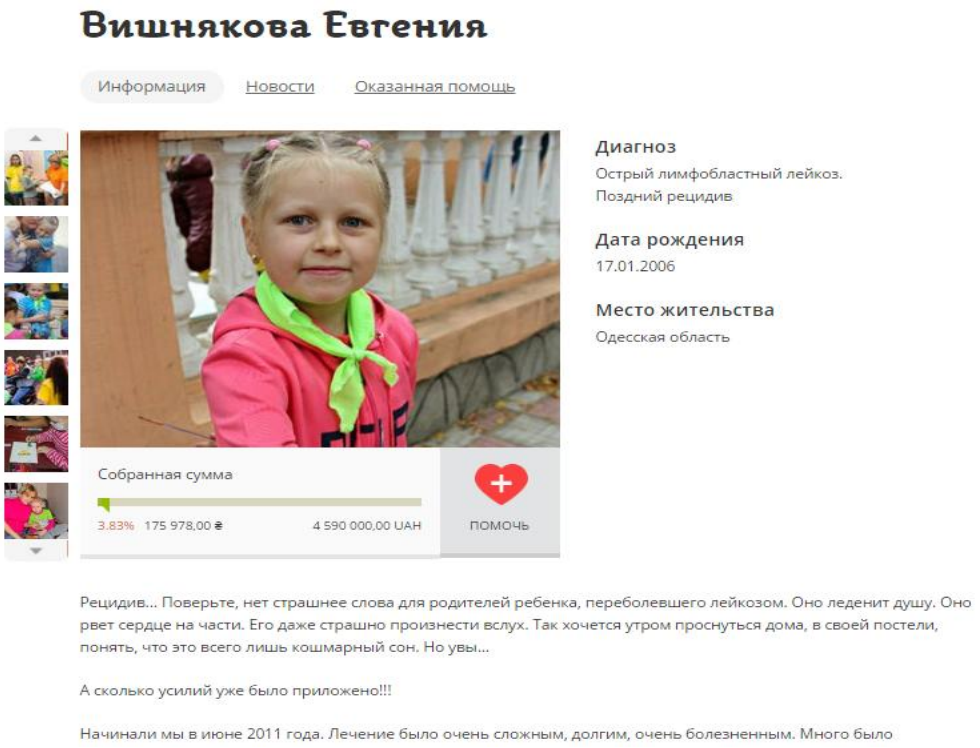

Рисунок 1.10 – Сторінка із повною інформацією про дитину

У нижній частині головної сторінки користувач побачить інформацію про тих, хто допоміг фонду (рисунок1.11) та тих дітей, яким допомогли (рисунок1.12).

Рисунок 1.11 – Інформація про тих, хто допоміг фонду

Рисунок 1.12 – Інформація про тих, кому допомогли

Результати огляду та аналізу можливостей системи «Pchelka» показали: веб-сайт має інтуїтивно зрозумілий інтерфейс та є простим у користуванні; контент є добре погрупованим, що значно спрощує процес пошуку необхідної інформації. Однак, було виявлено і певні недоліки: веб-сайт характеризується не дуже високою швидкодією та є російськомовним.

Розглянем веб-сайт Тернопільського обласного комунального дитячого будинку для дітей шкільного віку. На рисунку 1.13 проілюстровано головну сторінку системи.

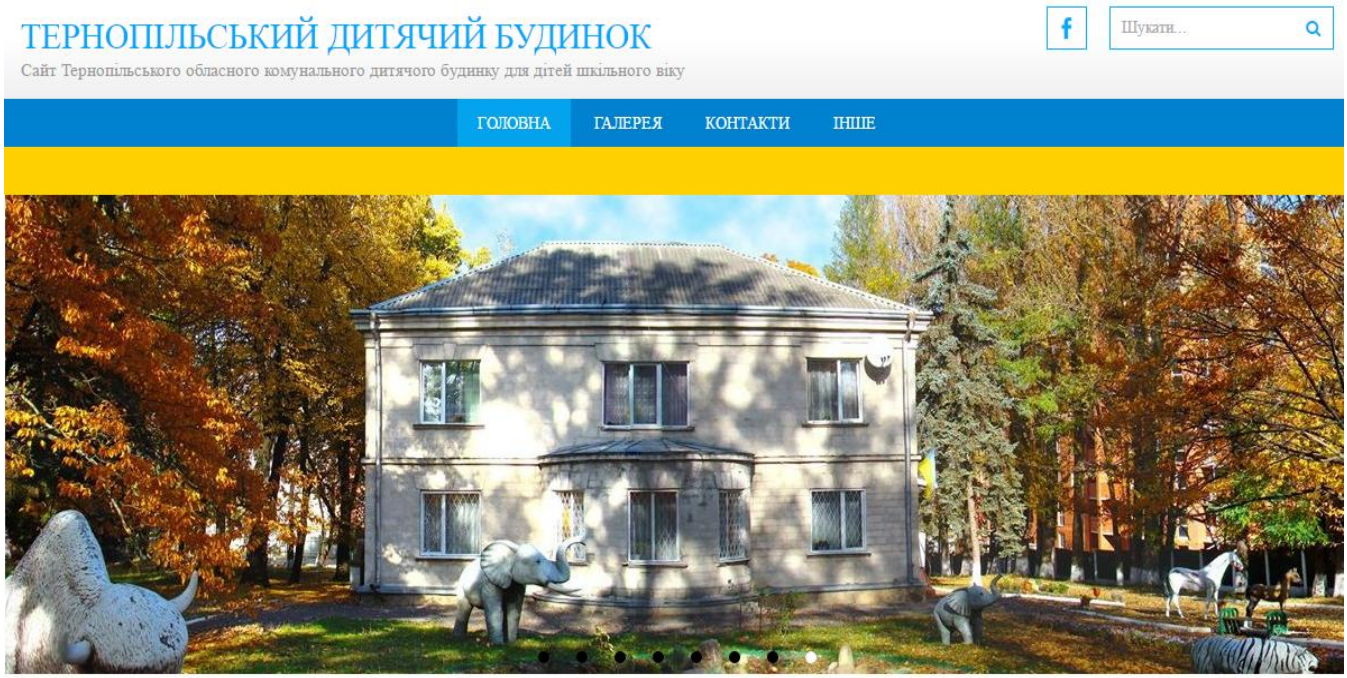

Рисунок 1.13 – Головна сторінка веб-сайту

Як видно з рисунку 1.13 веб-сайт має інтуїтивно зрозумілий інтерфейс. Посередині сторінки зображено фотографію дитячого будинку. В правому верхньому куті розміщено пошукове вікно, в якому користувач може знайти інформацію, яка його цікавить, а також іконку Facebook, яка надає посилання на сторінку дитячого будинку, у вищезгаданій соціальній мережі. На синьо-жовтому фоні розташоване меню, яке містить посилання на такі сторінки: «Головна», «Галерея», «Контакти», «Інше».

Внизу сторінки користувач побачить новини веб-сайту (рисунок 1.14).

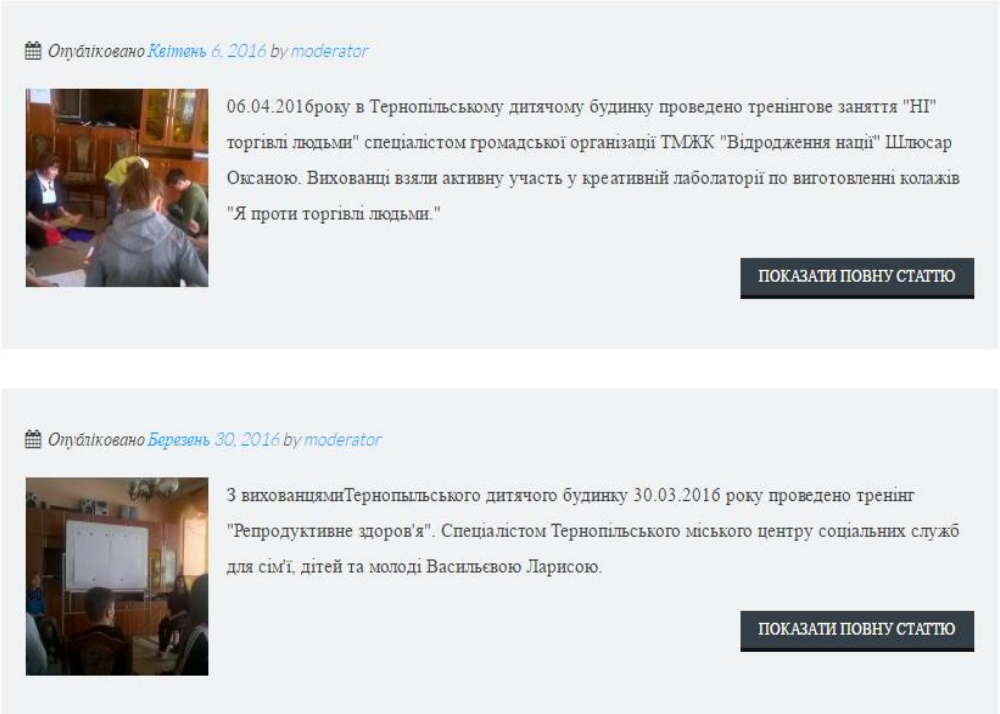

Рисунок 1.14 – Новини веб-сайту Тернопільського дитячого будинку

Біля кожної опублікованої новини є кнопка «Показати повну статтю». При кліку кнопки мишки на дану кнопку відкриється сторінка, зображена на рисунку 1.15.

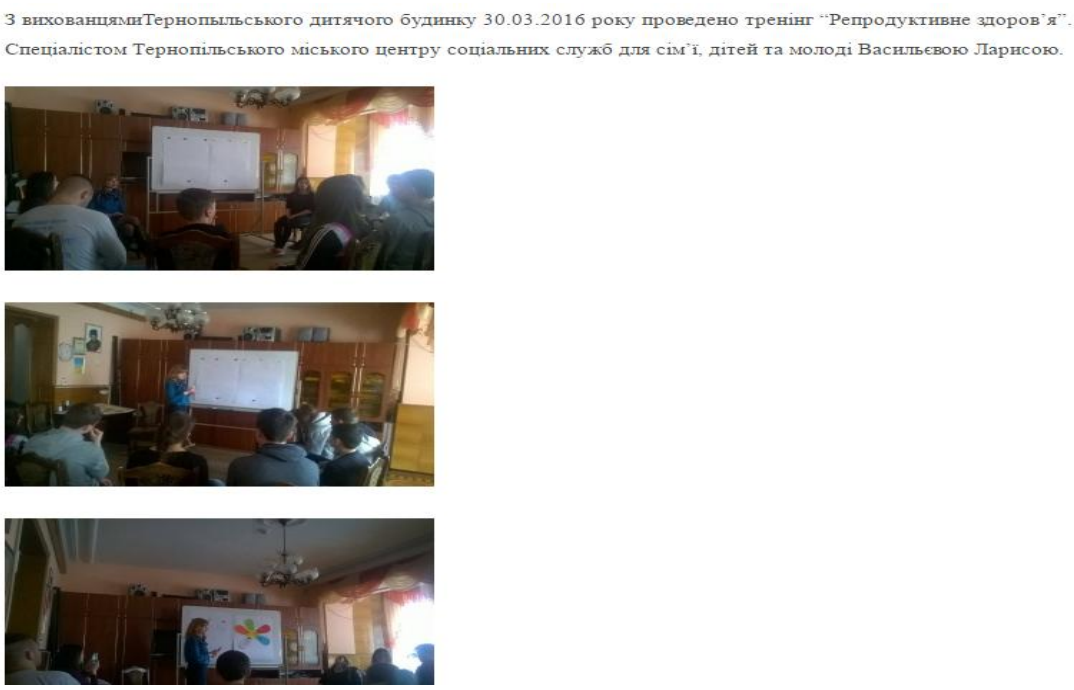

Рисунок 1.15 – Сторінка з деякою новиною

При виборі пункту меню «Галерея» перед користувачем відкриється сторінка, проілюстрована на рисунку 1.16. Як видно з даного зображення, фотографії погруповані за певною тематикою. Наприклад: «Андріївські вечорниці», «Весняний футбол», «Вечір реквієм» та інші.

Рисунок 1.16 – Галерея веб-сайту

Якщо користувач вибере пункт меню «Контакти», то перед ним відкриється сторінка зображена на рисунку 1.17.

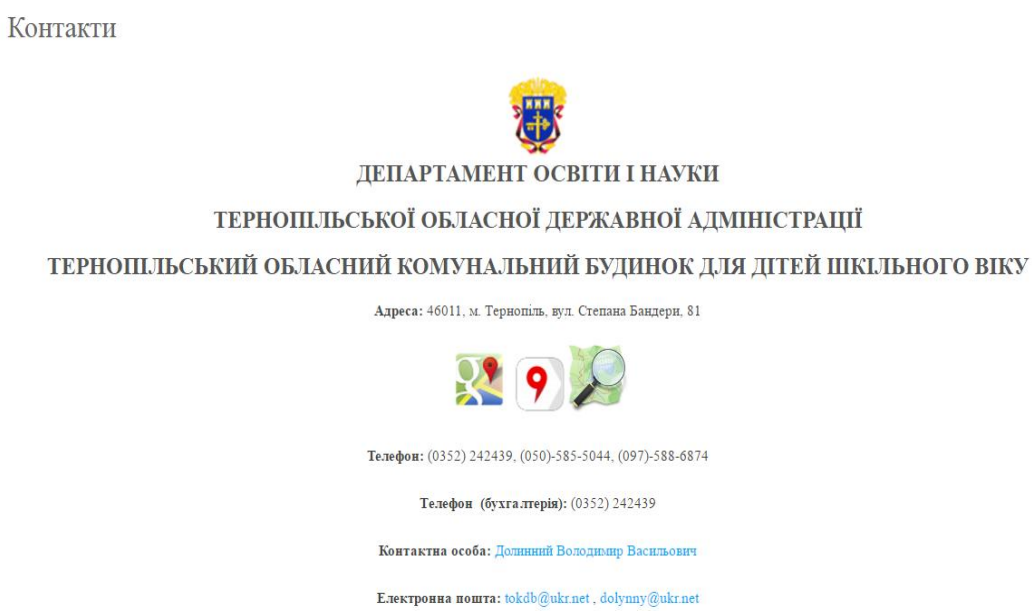

Рисунок 1.17 – Сторінка з контактними даними дитячого будинку

Підсумовуючи аналіз вищезгаданої системи, можна сказати, що вона є простою у користуванні. Основною перевагою є відсутність надлишкової інформації. Водночас це є і недоліком, оскільки інформація, яка висвітлена на веб-сайті не є достатньою.

Розглянемо сайт Харківського дитячого будинку «Родина». На рисунку 1.18 зображено його головну сторінку.

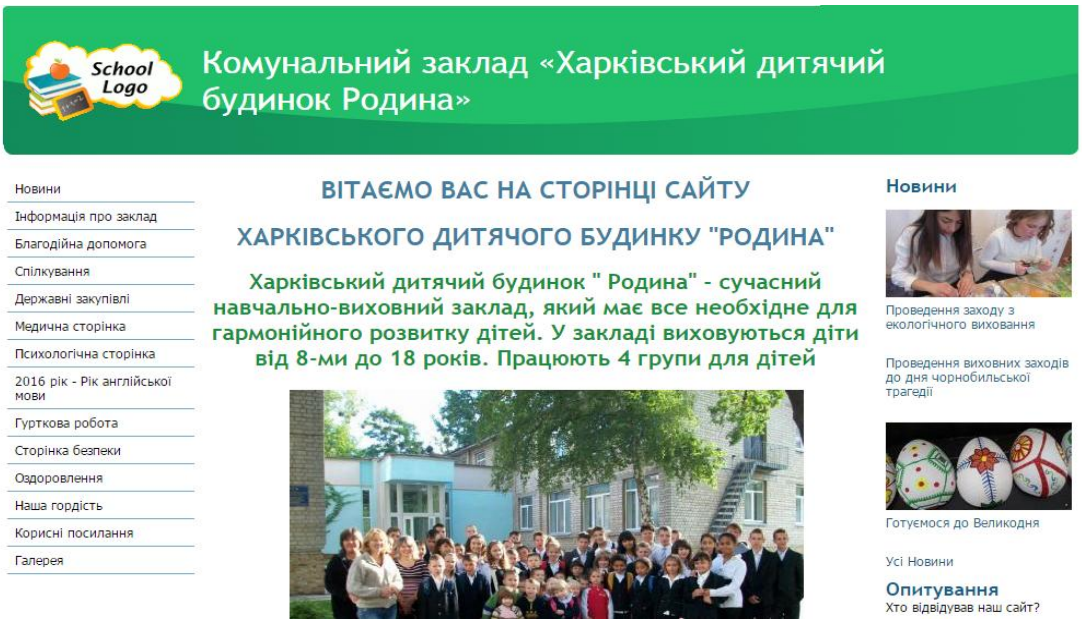

Рисунок 1.18 – Головна сторінка веб-сайту Харківського дитбудинку

Як видно із зображення 1.18, зліва знаходиться клікабельне меню веб-сайту з посиланням на наступні сторінки: «Новини», «Інформація про заклад», «Благодійна допомога», «Медична сторінка», «Наша гордість», «Галерея». Посередині зображено коротку інформацію про дитячий будинок з прикріпленими фотографіями. Зліва розташовані пости з останніми новинами.

Вибравши пункт меню «Новини», перед користувачем відкривається наступна сторінка, яка зображена на рисунку 1.19.

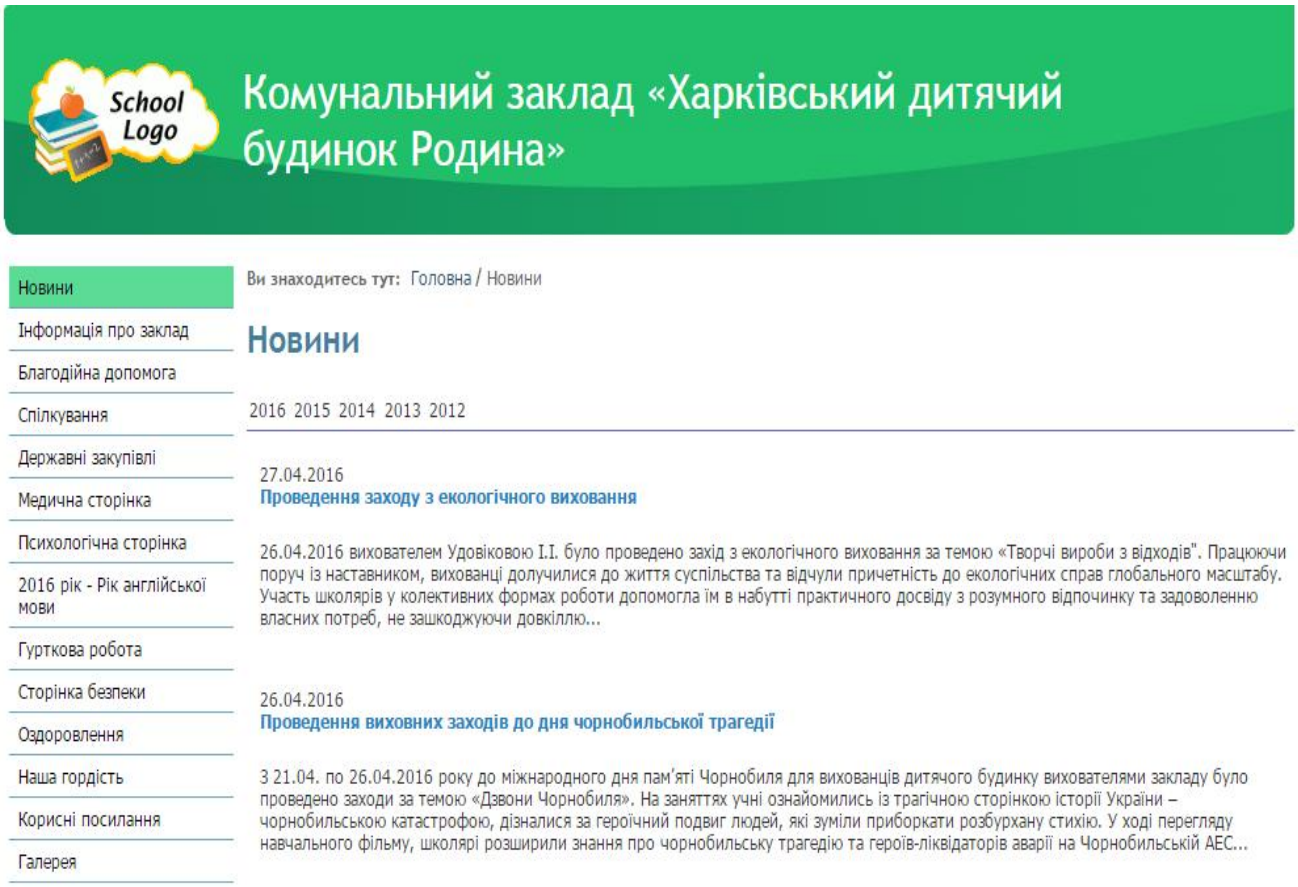

Рисунок 1.19 – Новини веб-сайту Харківського дитбудинку

Як видно з вищенаведеного зображення, новини веб-сайту подані у вигляді постів. Якщо користувачу цікава та чи інша новина, він просто клікає на її заголовок, який виділено синім кольором, і перед ним відкривається сторінка з детальним її описом та фотографіями.

Вибравши пункт меню «Благодійна допомога», відвідувач веб-сайту переходить на сторінку (рисунок1.20), де розміщена інформація про тих меценатів, які допомогли дитячому будинку.

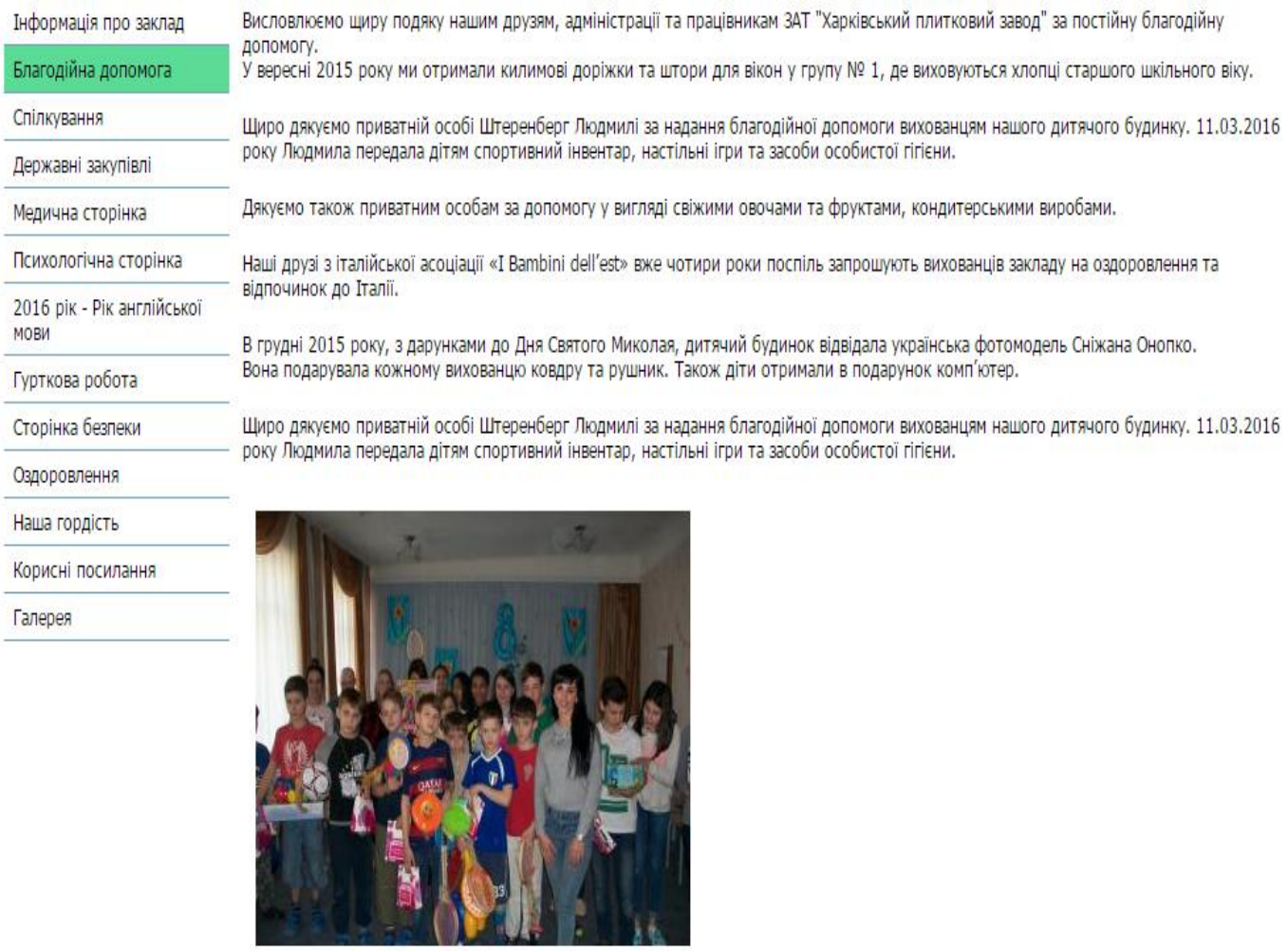

Рисунок 1.20 – Сторінка, на якій розміщена інформація про меценатів дитбудинку

Отже, веб-сайт Харківського дитячого будинку «Родина» є доволі простим у користуванні. В певній мірі присутня надлишковість інформації, що негативно позначається при роботі з системою.

В таблиці 1.4 наведено порівняльну характеристику вищеописаних програмних продуктів.

Таблиця 1.4

Порівняльна характеристика описаних програмних продуктів

Фірма розробник	Sponge	inNovatoinStudio	School Champion
Назва програмного продукту	Pchelka	Тернопільський обласний дитячий будинок	Харківський дитячий будинок «Родина»
Функціональність	Надання необхідної інформації	Надання необхідної інформації	Надання необхідної інформації
Інтерфейс користувача	Простий, зручний у користуванні	Простий, зручний у користуванні	Простий, зручний у користуванні
Допомога користувачу	Правила користування веб-сайтом	Немає	Немає

Провівши аналіз аналогових програмних продуктів, можна сказати, що подібні веб-сайти повністю задовольняють поставлену перед ними мету, а саме надають необхідну інформацію про той чи інший дитячий будинок користувачам мережі Інтернет, спрощують пошук потрібної інформації, є зручними у користуванні та мають інтуїтивно зрозумілий інтерфейс. Проте даним системам властивий і ряд недоліків. Зокрема надлишковість або недостатня кількість інформації, відсутність допомоги користувачу та недостатньо висока швидкодія.

1.4 Специфікація вимог до системи

Глосарій – це словник основних використовуваних термінів. Глосарій проекту знаходиться у додатку А.

На рисунку 1.21 проілюстровано діаграму варіантів використання системи.

Рисунок 1.21 – Діаграма варіантів використання системи

Нижче наведено опис варіантів використання системи.

Варіант використання «Перегляд інформації про вихованців дитячого будинку». Інформація про тих дітей, які потребують допомоги буде подана у вигляді фотографій з коротким описом, які знаходяться в слайдері. При кліку на зображення користувач перейде на сторінку з детальним описом. У таблиці 1.5 наведено короткий опис даного варіанту використання.

Таблиця 1.5

Варіант використання «Перегляд інформації про вихованців дитячого будинку»

Контекст використання	Перегляд інформації про вихованців дитячого будинку
Дійові особи	Користувач
Передумова	Користувач повинен зайти на веб-сайт
Тригер	-
Сценарій	Перехід на головну сторінку.
Пост-умова	-

На рисунку 1.22 проілюстровано розкадровку даного варіанту використання.

Рисунок 1.22 – Розкадровка варіанту використання «Перегляд інформації про вихованців дитячого будинку»

Варіант використання «Перегляд інформації про меценатів». Інформація про тих меценатів, які допомогли буде подана у вигляді фотографій з коротким описом, які знаходяться в слайдері. При кліку на зображення, користувач

перейде на сторінку з детальним описом. У таблиці 1.6 наведено короткий опис даного варіанту використання.

Таблиця 1.6

Варіант використання «Перегляд інформації про меценатів»

Контекст використання	Перегляд інформації про меценатів
Дійові особи	Користувач
Передумова	Користувач повинен зайти на веб-сайт
Тригер	-
Сценарій	Перехід на головну сторінку.
Пост-умова	-

На рисунку 1.23 зображено розкадровку даного варіанту використання.

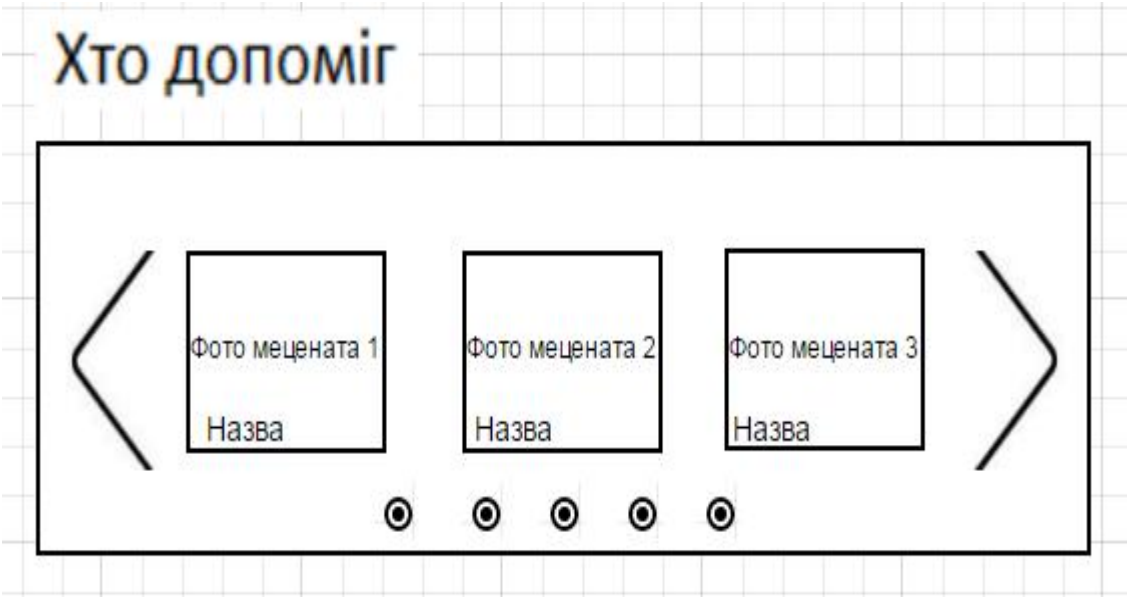

Рисунок 1.23 – Розкадровка варіанту використання «Перегляд інформації про тих, хто допоміг»

Варіанти використання «Перегляд контактної інформації» та «Перегляд інформації як допомогти» будуть реалізовано у вигляді постів. В таблиці 1.7 наведено короткий опис цих варіантів

Таблиця 1.7

Варіанти використання «Перегляд контактної інформації» та «Перегляд інформації як допомогти»

Контекст використання	Перегляд контактної інформації, Перегляд інформації «Як допомогти»
Дійові особи	Користувач
Передумова	Користувач повинен зайти на веб-сайт
Тригер	-
Сценарій	Перехід на головну сторінку.
Пост-умова	-

На рисунку 1.24 проілюстровано розкадровку даних варіантів використання.

Рисунок 1.24 – Розкадровка варіанту використання «Як допомогти» та «Контактна інформація»

Варіант використання «Перегляд фотографій». При виборі цього варіанту перед користувачем відкривається нова сторінка, де фотографії знаходяться в папках. При виборі певної папки перед відвідувачем відкриється нова сторінка, де він може побачити всі фотографії, які містяться в папці. В таблиці 1.8 наведено короткий опис даного варіанту використання.

Таблиця 1.8

Варіант використання «Перегляд фотографій»

Контекст використання	Перегляд фотографій
Дійові особи	Користувач
Передумова	Користувач повинен зайти на веб-сайт
Тригер	Відкрита сторінка з папками в яких містяться фотографії
Сценарій	1. Перехід на головну сторінку; 2. Вибір пункту меню «Галерея».
Пост-умова	-

На рисунку 1.25 зображено розкадровку даного варіанту використання.

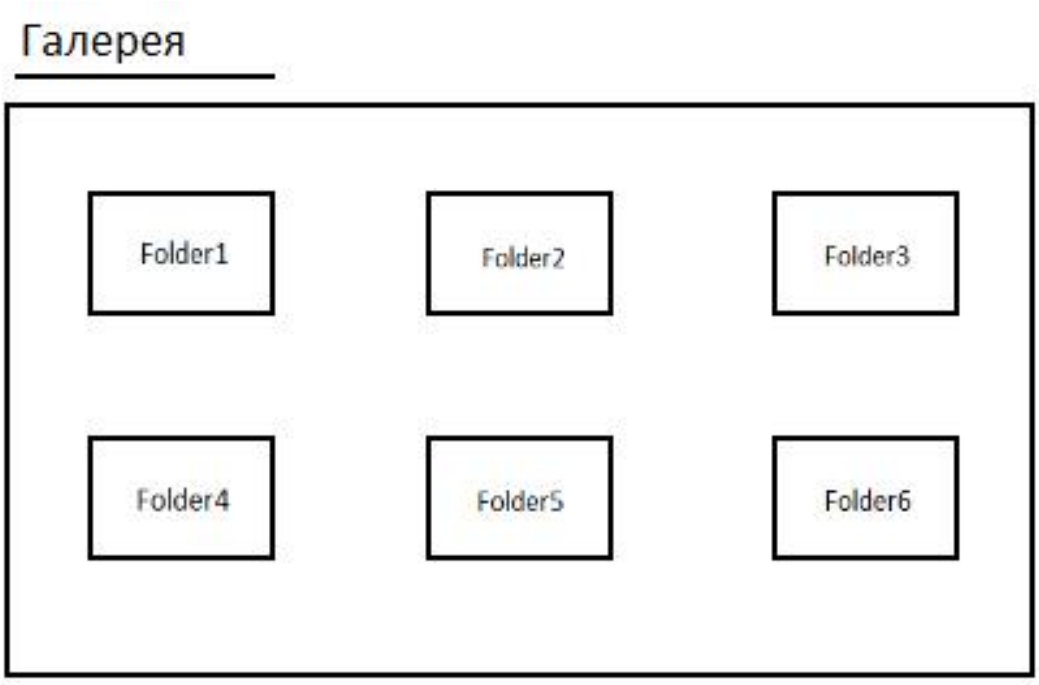

Рисунок 1.25 – Розкадровка варіанту використання «Перегляд фотографій»

Варіант використання «Авторизація в системі». При роботі з системою адміністратор повинен авторизуватись. Для цього він повинен зайти в свій аккаунт, ввівши у відповідні поля свій логін і пароль. В таблиці 1.9 наведено короткий опис варіанту використання «Авторизація в системі».

Таблиця 1.9

Варіант використання «Авторизація в системі»

Контекст використання	Авторизація в системі
Дійові особи	Адміністратор
Передумова	Реєстрація в системі
Тригер	Відкрита авторизаційна сторінка
Сценарій	1. Перехід на сторінку авторизації; 2. Заповнення полів форми; 3. Підтвердження.
Пост-умова	Зайдено в особистий кабінет

Варіант використання «Редагування інформації на сайті». При потребі адміністратор має можливість редагувати інформацію, яка знаходиться на веб-сайті. В таблиці 1.10 наведено короткий опис даного варіанту використання.

Таблиця 1.10

Варіант використання «Редагування інформації на сайті»

Контекст використання	Редагування інформації на сайті
Дійові особи	Адміністратор
Передумова	Авторизація в системі
Тригер	Відкрита сторінка редагування в адмінпанелі веб-сайту

Сценарій	1. Перехід в адмінпанель веб-сайту. 2. Внесення потрібних змін. 3. Збереження.
Пост-умова	-

У таблиці 1.11 наведено специфікацію функціональних вимог

Таблиця 1.11

Специфікація функціональних вимог

Ідентифікатор вимог	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
1	Перегляд потрібної інформації	Обов’язкове	Висока	-
2	Редагування Інформації	Обов’язкове	Висока	-
3	Авторизація в системі	Обов’язкове	Середня	-

У таблиці 1.12 наведено специфікацію нефункціональних вимог

Таблиця 1.12

Специфікація нефункціональних вимог

№.	Назва вимоги	Характеристики
1.	Застосовність	
1.1	Час для навчання користувачів	20 хв
1.2	Вимірюваний час відгуку для типових завдань	10-15 с
2.	Надійність	

2.1	Середній час безвідмовної роботи	50год
4	Проектні обмеження	
4.1	Мова програмування	PHP
5	Вимоги до документації, призначеної для користувача, і до системи допомоги	Інструкція для користувача
2.2	Середнє напрацювання до ремонту	10 міс
2.3	Максимальна норма помилок	35-40
3	Робочі характеристики	
3.1	Місткість (максимальне значення)	1000 користувачів

Продовження таблиці 1.12

6	Інтерфейси	
6.1	Інтерфейс користувача	Веб-додаток
6.2	Програмні інтерфейси	СУБД MySQL PhpMyAdmin
6.3	Комунікаційні інтерфейси	Доступ до мережі інтернет
7	Інші вимоги	
7.1	Вимоги до шрифтів	Гротеск(без зачісок)
7.2	Вимоги до дизайну	Молодіжний, веселий
7.3	Вимоги до мови	Українська

Висновки до першого розділу:

В цьому розділі було дано коротку характеристику об’єкта управління. Щоб краще зрозуміти суть проблеми було досліджено предметну область та проаналізовано аналогові продукти з метою виявлення переваг та недоліків. Створено словник основних використовуваних термінів та специфікацію вимог до системи.

РОЗДІЛ 2
ПРОЕКТУВАННЯ

2.1 Розробка архітектури програмної системи

Для створення архітектури даного веб-застосунку було обрано клієн-серверну архітектуру.

Архітектура клієнт-сервер працює наступним чином: коли клієнтський комп'ютер відправляє запит на ресурс або процес до сервера через мережеве

з'єднання, який потім обробляється і доставляється клієнту. Комп'ютер-сервер може управляти декількома клієнтами одночасно, в той час як один клієнт може бути пов'язаний з декількома серверами одночасно, кожен з яких забезпечує різний набір послуг[1].

Архітектура клієнт-сервер являє собою розподілену структуру програми, яка розділяє завдання або робочі навантаження між постачальниками ресурсу або сервісу, тобто серверами, і ініціаторами запитів, тобто клієнтами. Часто клієнти і сервери взаємодіють через комп'ютерну мережу на окремому обладнанні, але і клієнт і сервер може знаходитися в тій же системі.

Нижче наведені найбільш важливі переваги архітектури клієнт-сервер:

- знижує навантаження на робочу станцію, на якій працює додаток, тому що запити обробляються, таблиці скануються і т.д., на іншому комп'ютері, спеціально обладнаному для цих операцій;
- знижує навантаження на мережу, оскільки тільки результат пошуку транспортується до робочої станції, а не всі дані, які були відсканованої;
- у міру того як обсяг бази даних зростає, робочі станції не повинні бути обладнані додатковими дисками або пам'яттю. Ці зміни стосуються лише комп'ютера, на якому працює сервер бази даних.

Перехід до архітектури клієнт-сервер може значно підвищити продуктивність додатку, не вимагаючи, щоб ПК, на якому виконується клієнт, були змінені.

На рисунку 2.1 зображено принцип роботи архітектури веб-додатку.

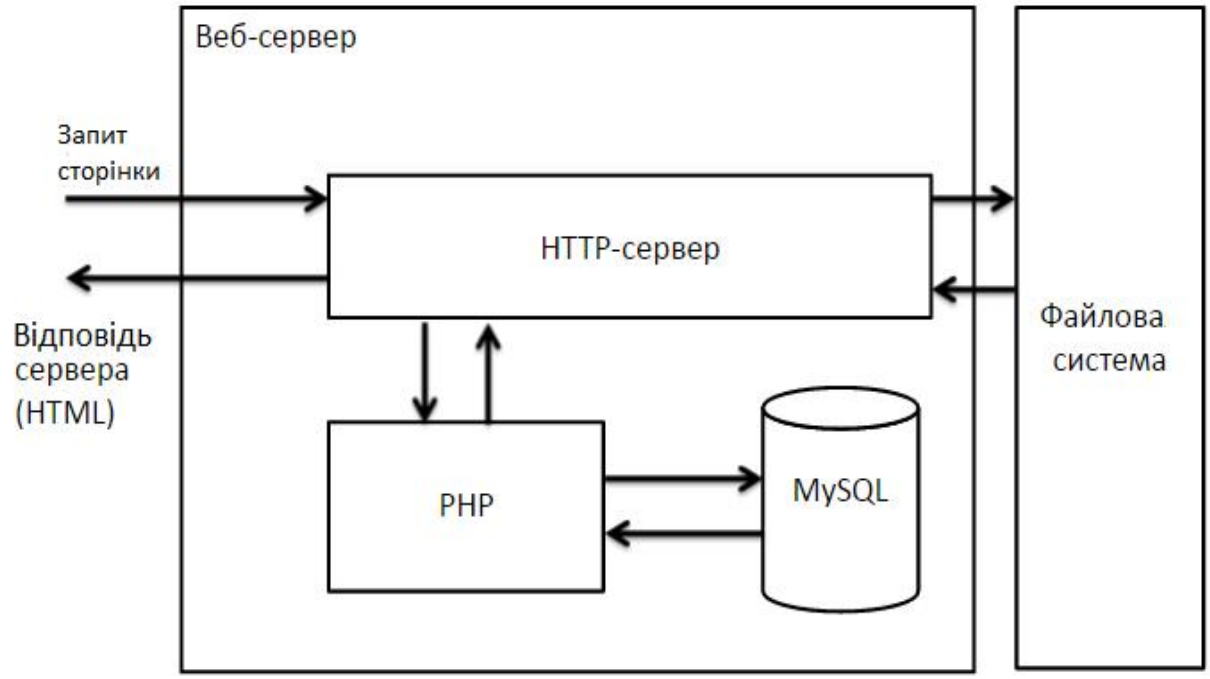

Рисунок 2.1 – Принцип роботи клієнт-серверної архітектури

Діаграма станів показує поведінку об’єктів у відповідь на зовнішні впливи. Діаграми станів подають абстрактний опис поведінки системи. Така поведінка аналізується і подається у вигляді ряду подій, які можуть відбутися в одному або декількох можливих станах. Таким чином кожна діаграма зазвичай представляє об'єкти одного класу і відстежує різні стани своїх об'єктів через систему. Існують різні типи діаграм станів, які мають різну семантику.

На відміну від дій користувача, дії адміністратора призводять до зміни системи, тому раціональніше буде створити діаграму станів[2,3] при використанні системи адміністратором.

На рисунку 2.2 проілюстровано діаграму станів системи, яка описує поведінку системи відповідно до дій адміністратора. Після авторизації в системі адміністратор заходить в особистий кабінет, що дає йому можливість додавати, видаляти або редагувати певну інформацію.

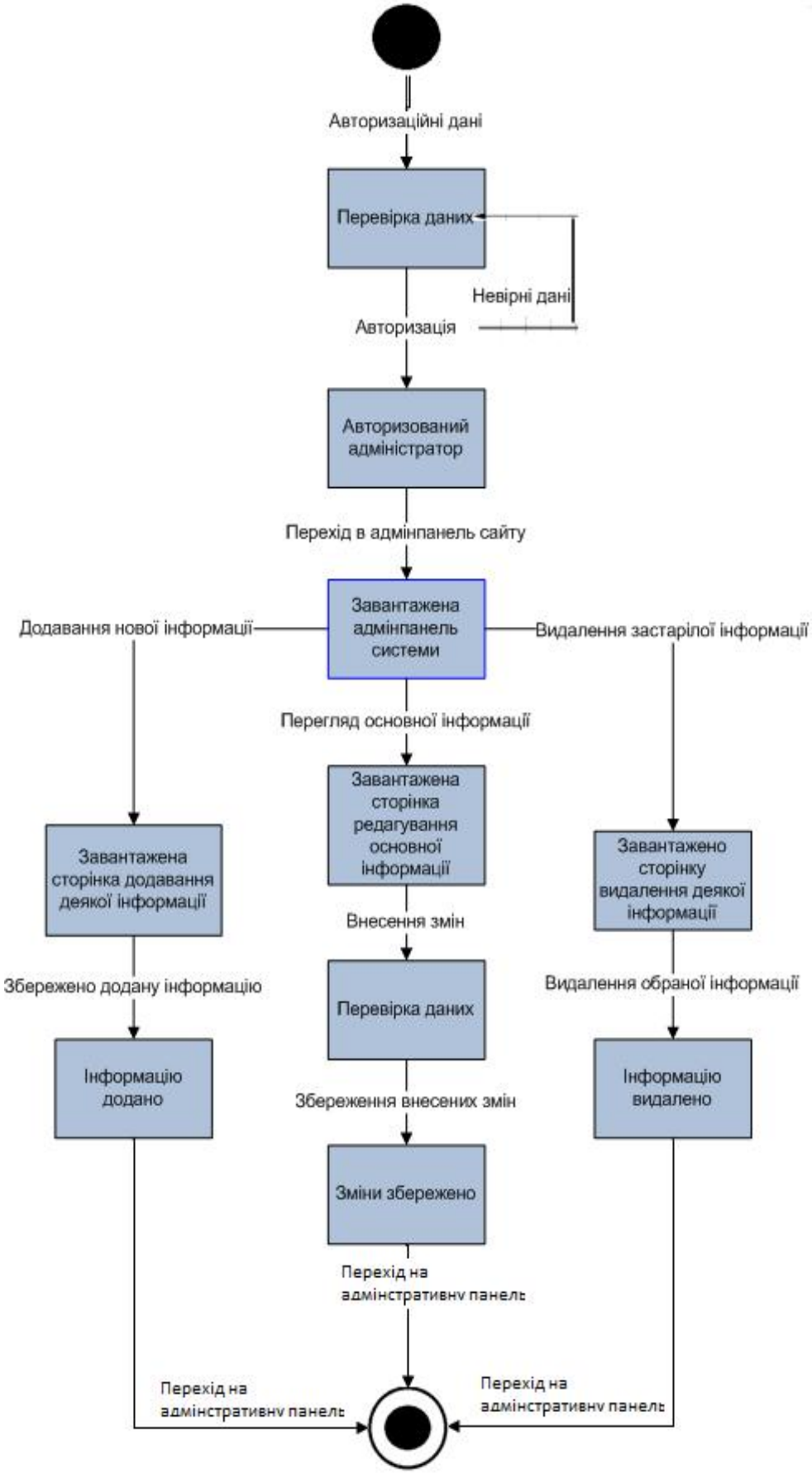

Рис. 2.2. Діаграма станів системи

На рисунку 2.3 зображено діаграму ієрархій функцій системи.

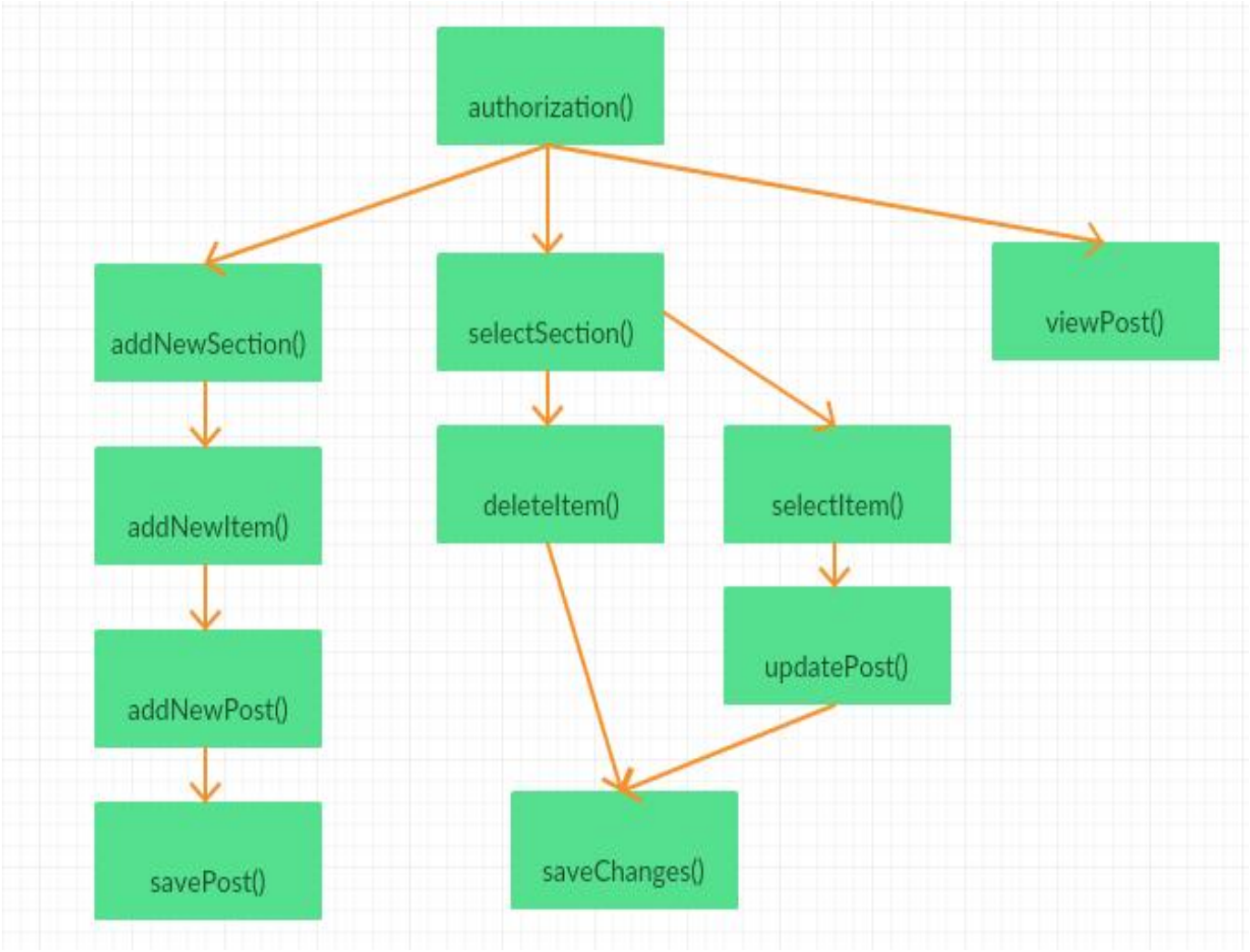

Рисунок 2.3 –Діаграма ієрархії функцій системи

На діаграмі ієрархії функцій показано назви функцій системи та послідовність їхнього виклику відповідно до дій адміністратора в системі. Наприклад функція addNewSection() не спрацює, якщо адміністратор не пройшов процес авторизації, за який відповідає функція authorization().

2.2 Проектування структури бази даних

Для реалізації структури бази даних обрано реляційну модель даних. Реляційна модель даних дозволяє представляти інформацію про предметну область за допомогою зв'язаних між собою таблиць (відношень). Основною

перевагою такого підходу є, перш за все, дуже висока швидкість пошуку за певним параметром, висока стабільність, невелика кількість програмного забезпечення для їх підтримки і розробки.

Процес логічного/фізичного проектування реляційної бази даних зводиться до визначення набору відношень, зв'язків між ними, що представлені у відношеннях парами „первинний ключ – зовнішній ключ" та формулюванням правил забезпечення цілісності (посилань, сутностей та корпоративної).

Одним з основних етапів проектування є створення діаграми корпоративної моделі даних. На рисунку 2.4 проілюстровано цю діаграму.

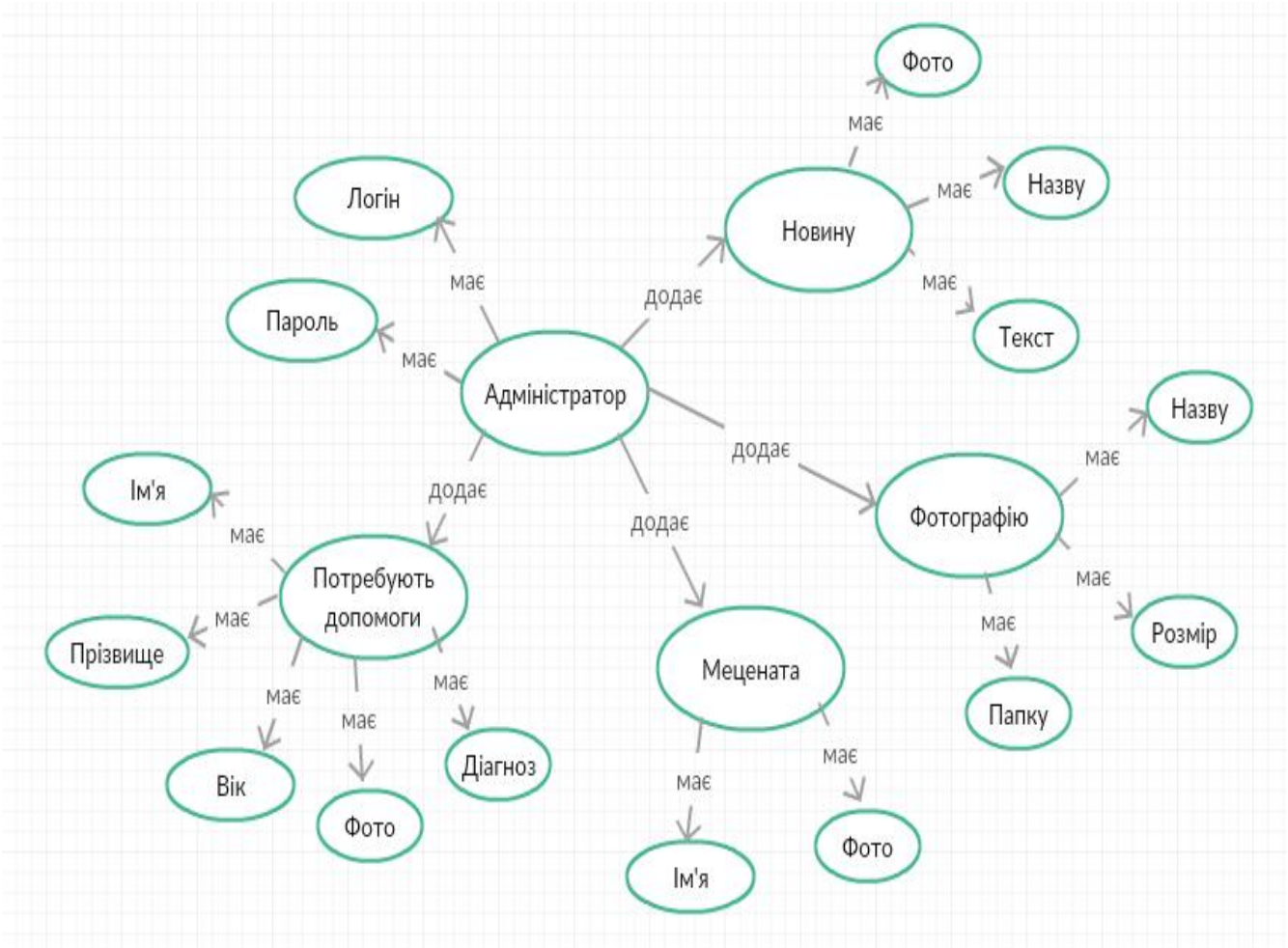

Рисунок 2.4 – Діаграма корпоративної моделі даних

Наступним етапом проектування є створення таблиці ідентифікаторів.

Таблиця 2.1

Таблиця ідентифікаторів

Об’єкт	Властивість	Тип	Розмірність	Ідентифікатор
Адміністратор	Логін Пароль	текст число	15 15	Login Password
Потребують допомоги	Ім’я Прізвище Вік Фото Діагноз	текст текст число фото текст	15 20 5 30	Name Surame Age Photo Diagnosis
Хто допоміг	Ім’я Фото	текст фото	15	Name Photo
Фотографія	Назву Розмір Назва папки	текст число текст	15 10 20	Name Size NameOfFolder
Новина	Назву Фото	текст фото	30	Name Photo

Діаграма сутність-зв'язок (ERD) являє собою графічне представлення інформаційної системи, яка показує відношення між предметами, в рамках цієї системи. Мета створення ERD-діаграми – забезпечити перегляд вимог, достатніх для задоволення потреб розроблюваної ІС (інформаційної системи). Фактично з ERD-діаграми і починається розробка моделі, коли визначається загальний перелік таблиць і зв'язків між ними. На рисунку 2.5 проілюстровано

ERD-діаграму системи «Інтерактивний веб-сервіс «Петриківський дитячий комунальний будинок-інтернат»».

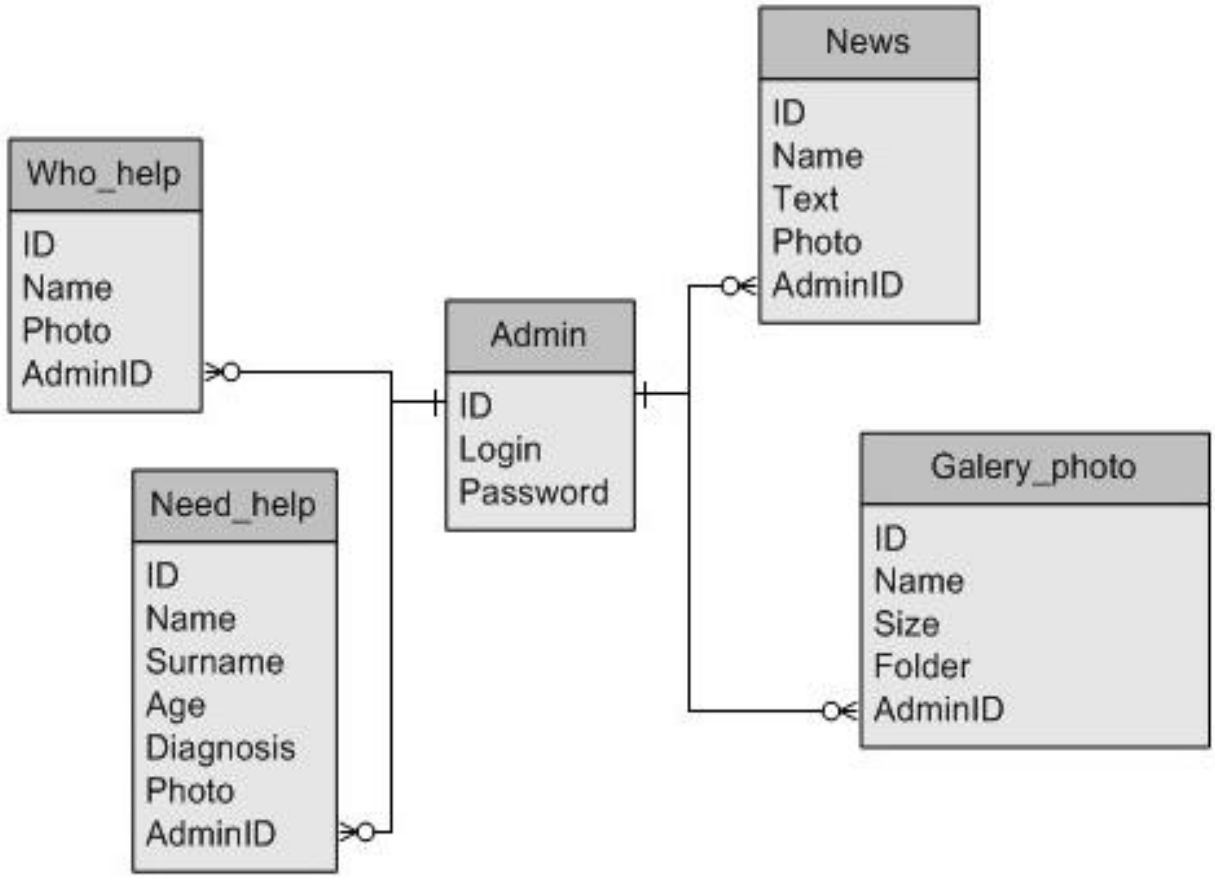

Рисунок 2.5 – ER – діаграма

На етапі фізичного проектування бази даних було вибрано систему управління базами даних MySql[5,6]. Створено набір реляційних таблиць і обмежень для них на основі інформації, представленої в логічній моделі даних, які зображені на рисунках 2.6– 2.10.

Фрагмент DDL коду для створення таблиці «Admin»:

```
CREATE TABLE `admin` (
  `id` INTEGER PRIMARY KEY AUTO_INCREMENT,
  `login` VARCHAR(255) NOT NULL,
  `password` VARCHAR(255) NOT NULL)
```

На рисунку 2.6 проілюстрована таблиця «Admin». У цій таблиці будуть зберігатись такі дані: логін та пароль адміністратора системи.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**ID** 🔑	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**Login**	varchar(20)	utf8_general_ci		No	*None*		
☐	3	**Password**	varchar(20)	utf8_general_ci		No	*None*		

Рисунок 2.6 – Структура таблиці «Admin»

Як видно з рисунку 2.6 дана таблиця складається з таких полів:

1) id – унікальний ідентифікатор. Тип даних integer;
2) login – поле для збереження логіна адміністратора. Тип даних varchar(20);
3) password – поле для збереження пароля адміністратора. Тип даних varchar(20).

На рисунку 2.7 зображена структура таблиці «Who_help». В даній таблиці буде зберігатись наступна інформація: ім’я того, хто допоміг дитячому будинку, зображення цієї людини або компанії та ідентифікатор адміністратора, який додавав запис в дану таблицю.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**ID** 🔑	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**Name**	varchar(15)	utf8_general_ci		No	*None*		
☐	3	**Photo**	mediumblob			No	*None*		
☐	4	**AdminID**	int(10)			No	*None*		

Рисунок 2.7 - Структура таблиці «Who_help»

Таблиця «Who_help» складається з таких полів:

1) id – унікальний ідентифікатор. Тип даних integer;

2) Name – поле для збереження імені того, хто допоміг. Тип даних varchar(15);
3) Photo – поле для збереження зображень. Тип даних mediumblob. Даний тип дозволяє зберігати зображення до 16 мегабайт.
4) AdminID – поле для збереження ідентифікатора адміністратора, який додавав запис в дану таблицю. Тип даних integer.

На рисунку 2.8 проілюстрована таблиця «News». Дана таблиця призначена для зберігання новин веб-сайту.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	ID 🔑	int(11)			No	*None*		AUTO_INCREMENT
☐	2	Name	varchar(20)	utf8_general_ci		No	*None*		
☐	3	Text	text	utf8_general_ci		No	*None*		
☐	4	Photo	mediumblob			No	*None*		
☐	5	AdminID	int(10)			No	*None*		

Рисунок 2.8 – Структура таблиці «News»

Таблиця «News» складається з таких полів:

1) id – унікальний ідентифікатор. Тип даних integer;
2) Name – поле для збереження назви новини. Тип даних varchar(20);
3) Text – поле для зберігання тексту новини. Тип даних text.
4) Photo – поле для збереження зображень. Тип даних mediumblob. Даний тип дозволяє зберігати зображення до 16 мегабайт.
5) AdminID – поле для збереження ідентифікатора адміністратора, який додавав запис в дану таблицю. Тип даних integer.

На рисунку 2.9 зображено структуру таблиці «Need_help». У цій таблиці будуть зберігатись дані про дітей, які потребують допомоги, а саме ім’я та прізвище, вік, фотографія, діагноз та ідентифікатор адміністратора, який додавав запис в дану таблицю.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**ID** 🔑	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**Name**	varchar(20)	utf8_general_ci		No	*None*		
☐	3	**Surname**	varchar(20)	utf8_general_ci		No	*None*		
☐	4	**Age**	int(5)			No	*None*		
☐	5	**Diagnosis**	text	utf8_general_ci		No	*None*		
☐	6	**Photo**	mediumblob			No	*None*		
☐	7	**AdminID**	int(10)			No	*None*		

Рисунок 2.9 - Структура таблиці «Need_help»

Таблиця «Need_help» складається з таких полів:

1) id – унікальний ідентифікатор. Тип даних integer;
2) Name – поле для збереження імені. Тип даних varchar(20);
3) Surname – поле для збереження прізвища. Тип даних varchar(20);
4) Age – поле для збереження віку. Тип даних integer(5);
5) Diagnosis – поле для збереження діагнозу. Тип даних text.
6) Photo – поле для збереження зображень. Тип даних mediumblob. Даний тип дозволяє зберігати зображення до 16 мегеабайт.
7) AdminID – поле для збереження ідентифікатора адміністратора, який додавав запис в дану таблицю. Тип даних integer.

На рисунку 2.10 проілюстровано таблицю «Galery_photo». В даній таблиці будуть зберігатись фотографій з галереї та дані про них, а саме: назва, розмір, папка.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**ID** 🔑	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**Name**	varchar(15)	utf8_general_ci		No	*None*		
☐	3	**Size**	int(10)			No	*None*		
☐	4	**Folder**	varchar(20)	utf8_general_ci		No	*None*		
☐	5	**AdminID**	int(11)			No	*None*		

Рисунок 2.10 - Структура таблиці «Galery_photo»

Таблиця «Galeri_photo» складається з таких полів:

1) id – унікальний ідентифікатор. Тип даних integer;
2) Name – поле для збереження назви зображення. Тип даних varchar(15);
3) Size – поле для збереження розміру картинки. Тип даних integer(10);
4) Folder – поле для збереження назви папки, в якій знаходиться зображення. Тип даних varchar(20);
5) AdminID – поле для збереження ідентифікатора адміністратора, який додавав запис в дану таблицю. Тип даних integer.

Висновки до другого розділу:

У даному розділі спроектовано архітектуру системи. За допомогою CASE засобів описано її поведінку.

За допомогою ефективних інструментів аналізу і проектування детально описано обробку інформації в системі. Спроектовано архітектуру бази даних. Створено та описано відношення, їх атрибути та взаємозв’язки між ними.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Програмна реалізація проекту

При створенні сучасного веб-сайту важливо, яким чином він буде реалізований, на якій платформі, на якій мові програмування. Саме від цього залежить надійність, продуктивність і якість роботи веб-сайту. Від цього залежить, наскільки просто буде підтримувати і розвивати цей веб-сайт в подальшому. Адже веб-сайти, які регулярно не оновлюються, поступово старіють і у відвідувачів втрачається інтерес.

Оскільки система є веб-орієнтованою, в процесі розробки буде використовувати мова PHP. Це скриптова мова загального призначення для розробки серверних частин веб-додатків[9,10].

Ця мова надає програмісту засоби для швидкого і ефективного вирішення поставлених завдань. Практичний характер PHP обумовлений п'ятьма важливими характеристиками:

- традиційністю;
- простотою;
- ефективністю;
- безпекою;
- гнучкістю.

Для реалізації цього проекту було обрано систему управління контентом Wordpress. WordPress - це CMS (движок), який використовують для створення, ведення та підтримки веб-сайту. Це система, яка зберігає записи, фотографії, замітки, коментарі і т.д. в єдине ціле, і дозволяє йому активно функціонувати. Основними перевагами WordPress над іншими CMS є:

- проста і швидка установка;
- простота у використанні;

- величезна кількість плагінів, які можуть розширити функціонал даної системи;
- WordPress абсолютно безкоштовний.

Для зберігання різноманітної інформації WordPress використовує базу даних MySql[11]. Тому перш ніж виконати установки CMS для веб-сайту, потрібно створити базу даних, з якою ця платформа буде працювати. Mysql - це реляційна система управління базами даних. Тобто дані в її базах зберігаються у вигляді логічно пов'язаних між собою таблиць, доступ до яких здійснюється за допомогою мови запитів SQL. Це досить швидка, надійна і, головне, проста у використанні СУБД, цілком підходяща для не надто глобальних проектів.

Працювати з Mysql можна не тільки в текстовому режимі, але і в графічному. Існує дуже популярний візуальний інтерфейс для роботи з цією СУБД. Називається він PhpMyAdmin. Цей інтерфейс дозволяє значно спростити роботу з базами даних в Mysql[13].

PhpMyAdmin дозволяє користуватися всіма перевагами браузера, включаючи прокрутку зображення, якщо воно не вміщається на екран. Багато з базових SQLфункцій роботи з даними в PhpMyAdmin зведені до інтуїтивно зрозумілих інтерфейсів, що нагадує перехід по посиланнях в мережі Інтернет.

Обраним середовищем для написання коду став Sublime Text. Це кросплатформенний текстовий редактор, який підтримує велику кількість мов програмування.

Першим етапом створення веб-сайту є верстка шаблону. Це процес формування веб-сторінки по готовому макету. Процес верстки полягає в створенні коду сторінки, за допомогою зрозумілої браузерам мови розмітки гіпертексту html, і оформлення її за допомогою каскадних таблиць стилів (CSS).

Цей процес складається з двох етапів:

1) Логічна розмітка. На цьому етапі пишеться html-код, який буде містити елементи для майбутнього макета.

2) Презентаційна розмітка. Тут, елементи, які створені засобами html-оформляються в відповідний вигляд, щоб можна було зрозуміти, за яку частину веб-сайту той чи інший елемент відповідає і де він повинен знаходитися.

Для початку створюємо макет веб-сайту[14]. В файл index.php в якому і почнемо будувати наш макет прописуємо стандартну розмітку, підключаємо кодування utf-8, і вказуємо українську мову мову для html (вказавши lang = "ua" в html тезі).

Додаємо метатеги в відділ head. Додамо метатег author, який в майбутньому допоможе нам з підтвердженням авторських прав на контент. Після чого прописуємо meta og (Open Graph), які допоможуть виводити інформацію на веб-сайт.

Нижче наведено фрагмент коду, який описує вище названі дії:

```
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="author" content="Тимчишин Василь" />
<meta content="Петриківський дитячий будинок-інтернат" />
<meta content="website" />
```

Далі підключаємо стилі, CSS файли, в яких прописані стилі оформлення елементів сторінок:

```
<link href="./css/bootstrap.css" rel="stylesheet" />
<link href="./style.css" rel="stylesheet" />
<link href="./css/bootstrap-responsive.css" rel="stylesheet" />
```

Потім створюємо файл style.css для подальшої стилізації та підключення шаблону на Wordpress. Нижче наведено фрагмент розмітки шаблону:

```
<body>
<div id="wrap">
<header>
<div class="navbar navbar-fixed-top">
<div class="navbar-inner">
<div class="container"><a class="brand" href="#"> Петриківсьсий дитячий будинок-
інтернат </a>
<div class="nav-collapse collapse">
<menu></menu></div>
</div>
</div>
</div>
</header>
```

```
<div class="container">
<div class="row">
<div class="span7">Тест контент</div>
<div id="push"></div>
</div>
<footer>
<div id="footer">
<div class="container">Футер</div>
</div>
</footer>
</body>
</html>
    }
```

Тепер наша розмітка готова. Переносимо шаблон в папку шаблонів WordPress. Для цього необхідно виконати наступні дії:

1) Переходимо в wp-content;
2) Вибираємо папку themes;
3) Копіюємо наш шаблон.

Після цього шаблон потрібно активувати. Для цього виконуємо наступні дії:

1) Відкриємо адмінпанель нашого WordPress (рисунок 3.1);
2) Переходимо в Зовнішній вигляд – Теми;
3) Знаходимо наш шаблон і активуємо.

Після активації шаблону, наш каркас перестав коректно відображатися. Це все тому, що в рядках, де підключаються css і js файли потрібно прописати коректний шлях до шаблону. Розробники WordPress для цього пропонують використовувати функцію get_template_directory_uri (). Нижче наведено фрагмент коду підключення шаблону:

```
    <Link href = "<? Php echo get_template_directory_uri ();?> / Style.css" rel = "stylesheet">
```

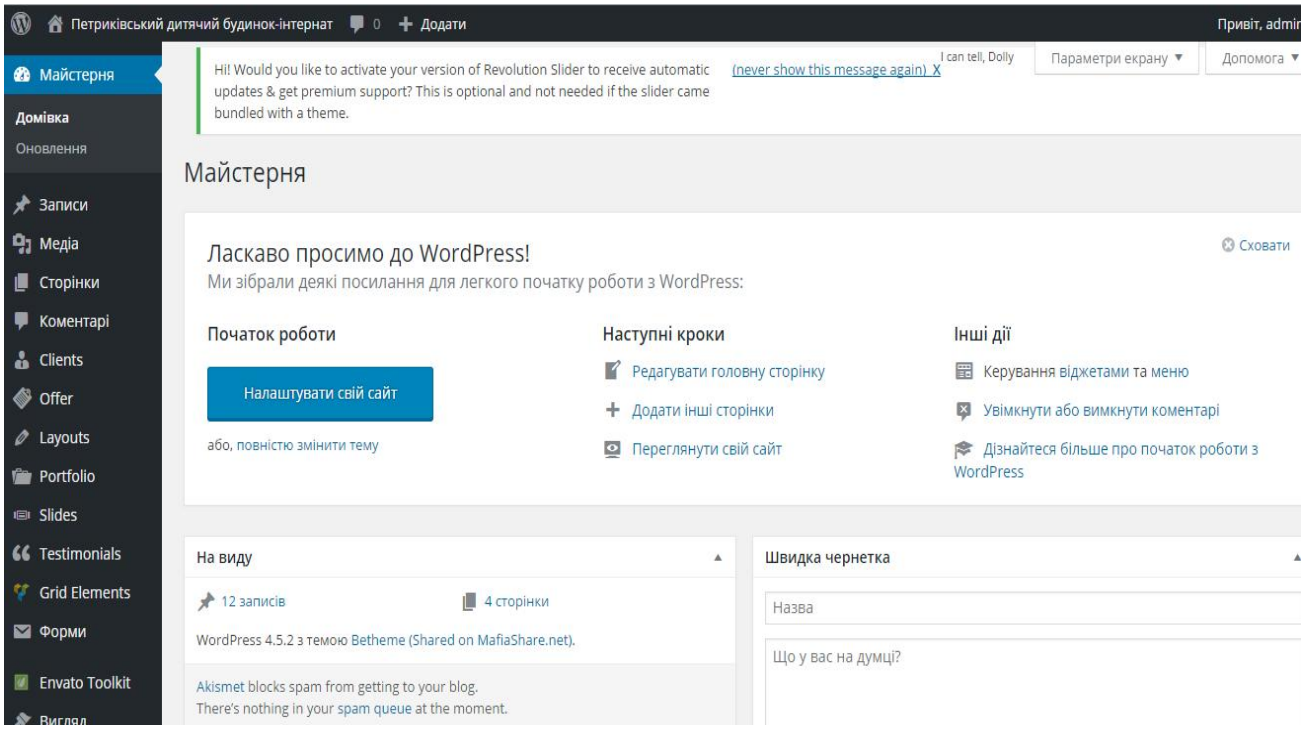

Рисунок 3.1 – Адмінпанель CMS WordPress

Потім розділяємо index.php на файли header, sidebar, footer.

Робота з файлом header.php:

1) Відкриваємо наш index.php файл;
2) Виділяємо фрагмент коду від початку файлу до тега </ header>;
3) Вирізаємо виділений фрагмент коду і копіюємо в файл header.php;
4) На місце вирізаного фрагмента з файлу index.php ставимо код виклику файлу header.php <? Php get_header (); ?>;
5) У файлі header.php перед тегом </ head> вставляємо код <? Php wp_head (); ?>. Він відповідає за виклик стилів і скриптів плагінів.

Робота з файлом sidebar.php:

1) В файлі index.php вирізаємо весь тег <aside>;
2) Вставляємо вирізаний фрагмент в файл sidebar.php;
3) На місці вирізаного коду прописуємо <? Php get_sidebar (); ?>.

Робота з файлом footer.php:

1) У файлі index.php вирізаємо фрагмент коду від тега <footer> до кінця файлу;
2) Вставляємо вирізаний фрагмент в файл footer.php;

3) Замість вирізаного коду з першого пункту ставимо код виклику <? Php get_footer (); ?>.

Тепер каркас майбутнього шаблону повністю готовий до роботи з ним. Для початку створимо навігацію веб-сайту. Перше, що нам знадобиться, це невеликий скрипт під назвою wp-bootstrap-navwalker, який забезпечує сумісність навігації Bootstrap і WordPress. Наступним етапом у створенні навігації буде підключення потрібних рядків у файлі functions.php. Нижче наведено фрагмент коду, який реалізує вищезгадані кроки:

```
<?php
 require_once('wp_bootstrap_navwalker.php');
 if (function_exists('add_theme_support')) {
 add_theme_support('menus');
 }
?>
```

Далі відкриваємо файл header.php і прописуємо код який буде виводити наше навігаційне меню. Код прописуємо всередину тегів <menu> </ menu>:

```
<?php
 'menu' => 'headMenu',
 'depth' => 2,
 'container' => false,
 'menu_class' => 'nav',
 );
 ?>
```

Після чого в файл style.css прописуємо, скидання відступів для тега <menu>:

```
 menu {
  padding:0;
  margin:0;
  }
```

Відкриємо файл footer.php і пропишемо jquery код який буде додавати клас .active в навігацію. Прописуємо даний скрипт після підключення jquery:

```
<script>
 jQuery(document).ready(function () {
 $(".current-menu-item").addClass("active");
 });
 </script>
```

Горизонтальне меню можна створити шляхом стилізації звичайного списку. Для елементів <li> потрібно присвоїти значення inline, щоб пункти списку розташовувалися один за одним.

Для розміщення пунктів меню по горизонталі, спочатку створимо маркований список з посиланнями:

```
<ul id="nav">
  <li><a href="#">Головна</a></li>
  <li><a href="#">Новини</a></li>
  <li><a href="#">Допомога</a></li>
  <li><a href="#">Фотогалерея</a></li>
  <li><a href="#">Про нас</a></li>
  <li><a href="#">Контакти</a></li>
</ul>
```

Тепер визначимо оформлення меню. Для цього задаємо такі стилі:

```
#nav {
 margin: 0;
 padding: 0;
 list-style-type: none;
 border: 2px solid #0066FF;
 width: 550px;
 text-align: center;
 background-color: #33ADFF;
```

В результаті отримуємо готове навігаційне меню, яке зображено на рисунку 3.2.

Рисунок 3.2– Меню веб-сайту

У файлі index.php прописуємо обов'язкову функцію циклу, всередині якої ми будемо формувати майбутній вигляд запису. Далі наведено фрагмент коду, який реалізує дану функцію.

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
<a  href="<?php  the_permalink();?>"><h2><?php  the_title();  ?></h2></a>
<?php endwhile; ?><?php endif; ?>
```

Для оформлення вид виведення інформації про записи, для цього відкриємо style.css і пишемо код:

```
.custom {
font-size: 13px;
color: grey;
padding-bottom: 5px;
margin-bottom: 20px;
border-bottom: 1px solid #EEE;
}

.custom ul {
list-style: none;
display: inline-block;
margin: 0;
}
```

Аналогічно додаємо всі інші записи.

Далі виводимо інформацію в футер веб-сайту. Для цього в файлі footer.php прописуємо код:

```
<P> (C) 2013- 2014 <? Php echo date ('Y');?> Петриківський дитячий будинок-інтернат </p>
```

Результат даної дії зображений на рисунку 3.3.

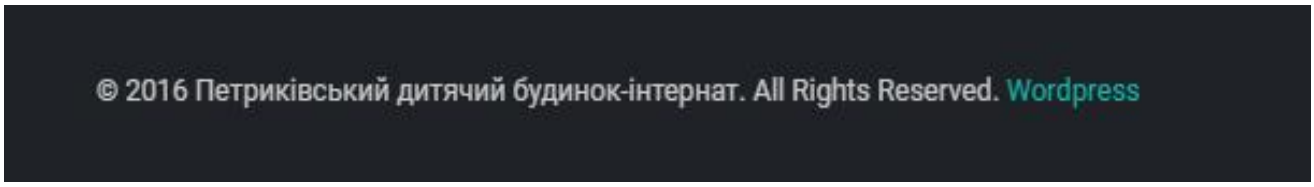

Рисунок 3.3 - Футер веб-сайту

А далі перейдемо безпосередньо до кодування основних функцій системи. Для прикладу розглянемо фрагмент коду для реалізації слайдера:

```
<div class="image_holder">
<img src="/images/photo1.jpg">
<img src="/images/photo2.jpg">
<img src="/images/photo3.jpg">
<img src="/images/photo4.jpg">
<img src="/images/photo5.jpg">
</div>
```

Для підключення слайдера на сторінку, потрібно підключити бібліотеку jQuery і файл, в якому описані три ефекти слайда:

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>
<script src="/js/slider_parts.js" type="text/javascript"></script>
```

Для управління слайдером потрібний такий код:

```
$(document).ready(function() {
 $(".slider").each(function () {
  var obj = $(this);
  $(obj).append("<div class='nav'></div>");
  $(obj).find("li").each(function () {
   $(obj).find(".nav").append("<span rel='"+$(this).index()+"'></span>");
   $(this).addClass("slider"+$(this).index());
  });
  $(obj).find("span").first().addClass("on");
 });
});
function slider (obj, sl) {
 var ul = $(sl).find("ul");
 var bl = $(sl).find("li.slider"+obj);
 var step = $(bl).width();
 $(ul).animate({marginLeft: "-"+step*obj}, 500);
}
$(document).on("click", ".slider .nav span", function() {
 var sl = $(this).closest(".slider");
 $(sl).find("span").removeClass("on");
 $(this).addClass("on");

  $(this).addClass("on");
    var obj = $(this).attr("rel");
    slider(obj, sl);
    return false;
   });
   </script>
```

На рисунку 3.4 зображено результат наведених вище дій. Зауважимо, що весь лістинг реалізації слайдера та інших модулів знаходиться в додатку В.

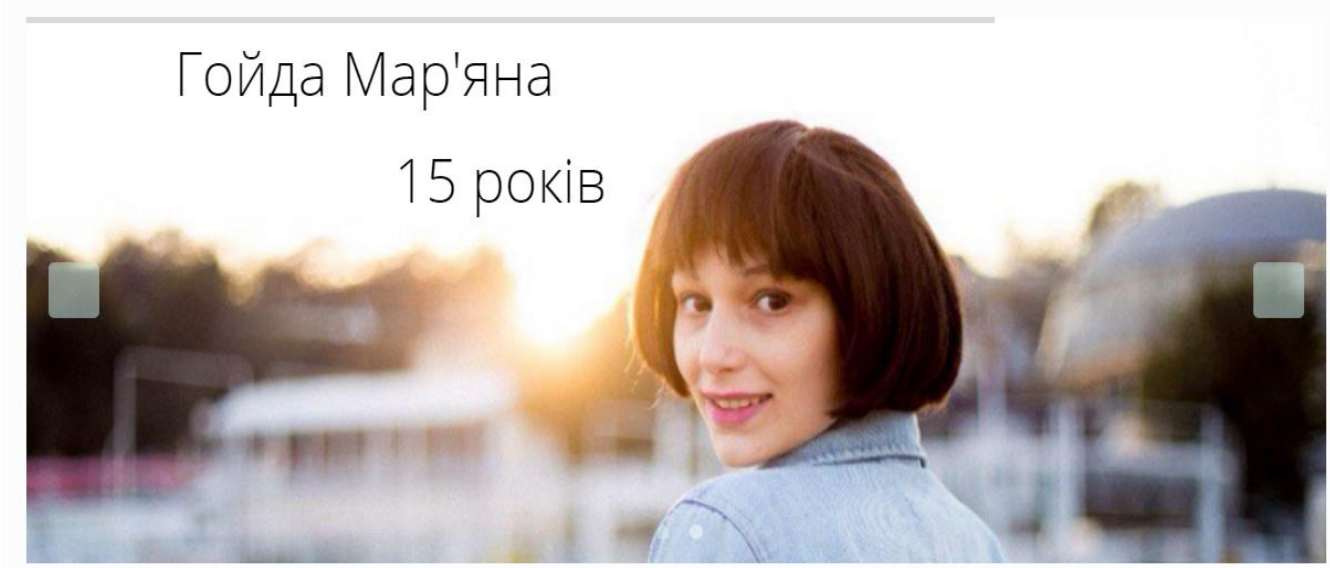

Рисунок 3.4 – Слайдер з інформацією про вихованців дитячого будинку

3.2 Програмна реалізація бази даних

Після процесу проектування бази даних відбувається реалізація спроектованої схеми бази даних у вигляді реальної бази даних, яка отримана як результат виконання скрипта. Для програмної реалізації бази даних було обрано СУБД MySQL. Вона є найбільш пристосованою для застосування у веб-середовищі. Відомо, що для виконання додатків клієнта на більшості хостинг-майданчиків провайдери надають невелику кількість ресурсів. Тому для даного застосування необхідна високоефективна СУБД, що має високу надійність (більшість веб-додатків і веб-сайтів повинні працювати в режимі 24/7).

В якості інтерфейсу для MySQL використовується утиліта phpMyAdmin.

Щоб створити таблицю, скористаємося виразом CREATE TABLE, в якому задамо ім'я нашої нової таблиці. Вираз починається з CREATE TABLE, далі вказуємо ім'я таблиці. Потім в дужках вказується список стовпців і інформація про ключі. Кожному стовпцю дається ім'я, вказується тип даних, вказується атрибут NULL або NOT NULL (тут NOT NULL означає, що колонка не може мати значення NULL), і значення за замовчуванням, якщо воно доречно. Нижче наведено код створення таблиці «Admin»:

```
CREATE TABLE `admin` (
  `id` INTEGER PRIMARY KEY AUTO_INCREMENT,
  `login` VARCHAR(20) NOT NULL,
  `password` VARCHAR(20) NOT NULL)
```

Код створення таблиці «Who_help»:

```
CREATE TABLE ` Who_help ` (
  `id` INTEGER PRIMARY KEY AUTO_INCREMENT,
  `name` VARCHAR(15) NOT NULL,
  `photo` mediumblob NOT NULL)
```

Код створення таблиці «Need_help»:

```
CREATE TABLE ` Need_help ` (
  `id` INTEGER PRIMARY KEY AUTO_INCREMENT,
  `name` VARCHAR(20) NOT NULL,
  `surname` VARCHAR(20) NOT NULL,
  `age` INT (15) NOT NULL,
  ` Diagnosis ` text NOT NULL
  `photo` mediumblob NOT NULL,)
```

Код створення таблиці «News»:

```
CREATE TABLE ` News ` (
  `id` INTEGER PRIMARY KEY AUTO_INCREMENT,
  `name` VARCHAR(20) NOT NULL,
  `text` text NOT NULL,
  `photo` mediumblob NOT NULL)
```

Код створення таблиці «Galery_photo»:

```
CREATE TABLE ` Galery_photo ` (
  `id` INTEGER PRIMARY KEY AUTO_INCREMENT,
  `name` VARCHAR(15) NOT NULL,
  `size` INT(10) NOT NULL,
  `folder` VARCHAR(20) NOT NULL)
```

Стовпець id - це первинний ключ (primary key), який унікально ідентифікує кожного адміністратора. Тип даних цього стовпця - INT (ціле число нормального розміру), MySQL призначає унікальні значення для цього стовпця, завдяки атрибуту auto_increment. Інші стовпці використовують в якості типів даних рядки змінної довжини (VARCHAR).

Далі додаємо в таблицю запис – адміністратора з логіном і паролем admin.

Нижче наведено SQL код даного запиту:

```
INSERT INTO `admin`(`ID`, `Login`, `Password`) VALUES (1,admin,admin)
```

В таблиці 3.1 наведено структуру звіту, який використовується при виведенні на веб-сайт інформації про вихованців дитячого будинку.

Таблиці 3.1

Звіт по вихованцю дитячого будинку

Name	Surname	Age	Diagnosis
Мар’яна	Гойда	15	Остеосаркома

Реалізовувати такий звіт буде наступний код:

```
SELECT * FROM `need_help` WHERE ID=1
```

Кожний запис такого звіту буде відповідати окремому слайду в слайдері на головній сторінці веб-сайту.

В таблиці 3.2 наведено структуру звіту, який використовується при виведенні на веб-сайт інформації про меценатів дитячого будинку.

Таблиці 3.2

Звіт по меценатах

Name	Photo
Благодійний фонд "Приятелі дітей"	

Реалізовувати такий звіт буде наступний код:

```
SELECT * FROM `Who_help` WHERE ID=1
```

Аналогічним чином будуть реалізовані та застосовані інші звіти, які використовуються при виводі інформації на сторінки веб-сайту.

Висновки до третього розділу:

У даному розділі було детально описано програмну реалізацію проекту. Наведено фрагменти коду та зображення тих інструментів, які використовувались при розробці даної системи.

Програмно реалізовано базу даних та описано дії, які виконувались відносно конкретних таблиць.

РОЗДІЛ4
ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1 Тестування

Тестування програмного забезпечення – це процес дослідження, випробування програмного продукту, який має дві цілі:

- продемонструвати розробникам і замовникам, що програма відповідає вимогам;
- виявити ситуації, в яких поведінка програми є неправильною, небажаною або не відповідає специфікації.

Для тестування системи було розроблено два види тестів: функціональне та GUI(graphical user interface).

Функціональне тестування - це тестування програмного забезпечення з метою перевірки реалізації функціональних вимог, тобто здатності ПЗ в певних умовах вирішувати завдання, потрібні користувачу[17,18].

У таблиці 4.1 наведено результати функціонального тестування для всіх варіантів використання системи.

Таблиця 4.1

Результати проходження функціонально тестування

	Варіант використання	Результат тесту
1.	Авторизація адміністратора в системі	Пройдено
2.	Додавання інформації про вихованців дитячого будинку	Пройдено
3.	Додавання інформації про меценатів	Пройдено
4.	Редагування основної інформації	Пройдено

5.	Додавання нових фотографій	Пройдено

Продовження таблиці 4.1

6.	Додавння нових постів	Пройдено
7.	Перегляд основної інформації	Пройдено
8.	Видалення застарілої інформації	Пройдено

Опис тестових випадків для основних функцій системи наведено в таблиці 4.2.

Таблиця 4.2

Test cases

N/n	Тестовий випадок	Результат проходження тесту
1.	Перевірка на проктрутку слайдера №1	Пройдено
2.	Перевірка на прокрутку слайдера №2	Пройдено
3.	Перевірка заповнення обов’язкових полів (для адміністратора)	Пройдено
4.	Перевірка відповідності введених даних у полях (для адміністратора)	Пройдено
5.	Перевірка пунктів меню	Пройдено
6.	Перевірка посилань	Пройдено

Під час фази розробки тестів було спроектовано 34 функціональні тестові випадки[15,16]. Таблиця 4.3 показує розподіл функціональних тестових

випадків і наборів тестових даних для цих випадків за варіантами використання.

	Варіант використання	Тестові випадки	Тестові дані
1.	Авторизація адміністратора в системі	2	9
2.	Додавання інформації про вихованців дитячого будинку	4	12
3.	Додавання інформації про меценатів	4	12
4.	Редагування основної інформації	5	15
5.	Додавання нових фотографій	4	12
6.	Додавння нових постів	4	12
7.	Перегляд основної інформації	6	18
8.	Видалення застарілої інформації	4	10
	Загалом	33	100

У таблиці 4.3 приведено тестові випадки для тестування користувацького інтерфейсу.

Таблиця 4.4

Тестові випадки тестування інтерфейсу користувача

	Тестовий випадок	Результат тесту
1.	Перевірка розміщення елементів на екранних формах	Пройдено
2.	Перевірка змісту і оформлень, що виводяться	Пройдено
4.	Перевірка реакції системи на дії	Пройдено

	користувача	
5.	Перевірка часу відгуку на команди	Пройдено

Тестові випадки, описані у таблиці 4.3, пройдено за допомогою ручного тестування.

4.2 Розгортання програмного продукту

Для того щоб всі мали можливість побачити веб-сайт дитячого будинку його необхідно розмістити на хостинг, який потрібно оплатити[20]. Після оплати до користувача на пошту прийде лист, в якому, крім інших, будуть дані для підключення по FTP і до панелі управління.

В першу чергу, користувачу потрібно зайти в панель управління хостингом та створити базу даних. Потім звідти ж перейти в phpmyadmin і імпортувати всю базу даних веб-сайту на хостинг в створену базу даних. Для підключення по FTP можна використовувати програму: Filezilla.

Тепер користувачу потрібно завантажити всі файли веб-сайту в папку www (або public_html).

Нижче наведено алгоритм дій для завантаження веб-сайту на хостинг:

1) зайти в панель управління хостингом;
2) створити в ній нову базу даних;
3) через phpMyAdmin імпортувати базу даних на хостинг;
4) встановити з'єднання із хостингу по FTP;
5) завантажити всі файли сайту в папку www на хостингу.

4.3 Інструкція користувача

Для того щоб користувач мав можливість повноцінно використовувати систему потрібно мати встановлений браузер на ПК. На рисунку 4.1 зображено вигляд головної сторінки веб-сайту. Зверху знаходиться навігаційне меню.

Нижче розміщено слайдер з вихованцем дитячого будинку, який потребує допомоги, де вказано його ім’я та вік.

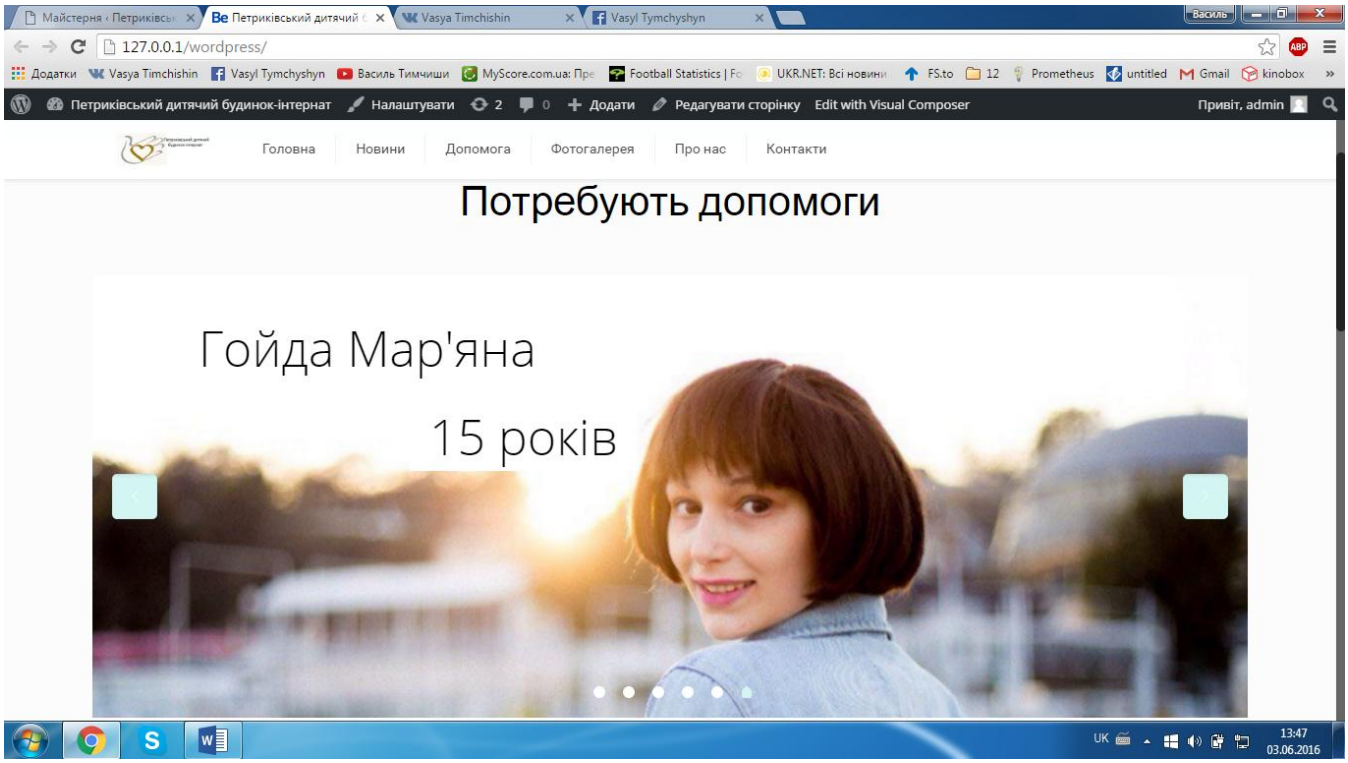

Рисунок 4.1 – Головна сторінка веб-сайту

Нижче розміщено секцію новин. Кожна новина має назву, прикріплене фото та дату публікації.

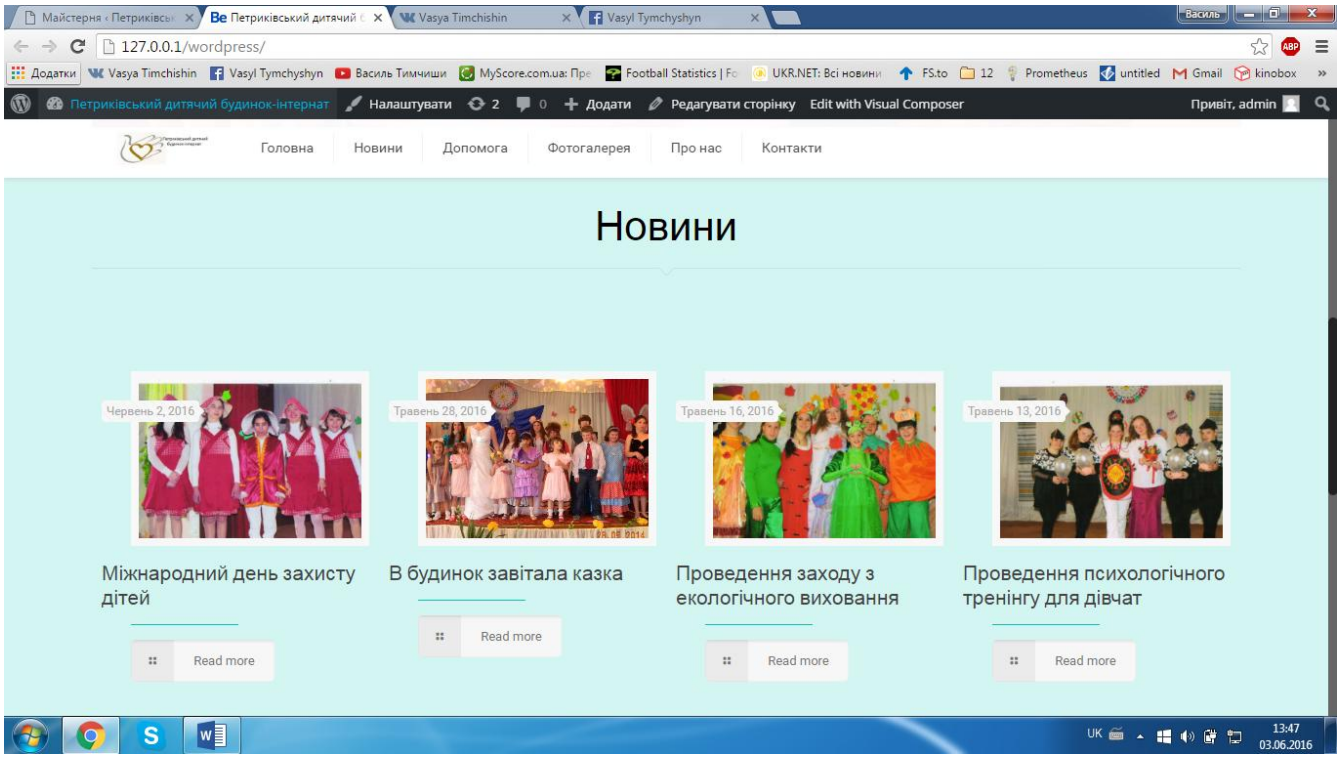

Рисунок 4.2 – Новини веб-сайту

Натиснувши кнопку «Read more», перед користувачем відкривається сторінка з вибраною новиною. Результат цієї дії зображений на рисунку 4.3.

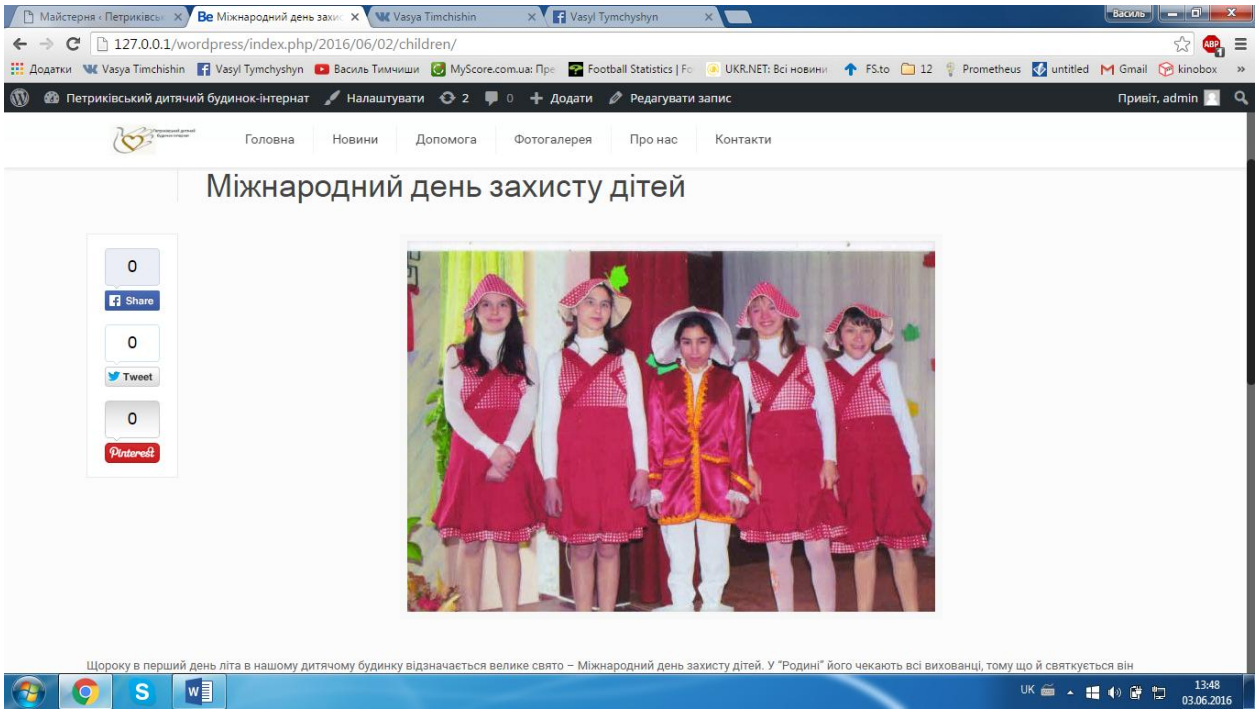

Рисунок 4.3 – Сторінка з вибраною новиною

Погортавши вниз користувач має можливість побачити коротку інформацію про дитячий будинок та яким чином йому можна допомогти.

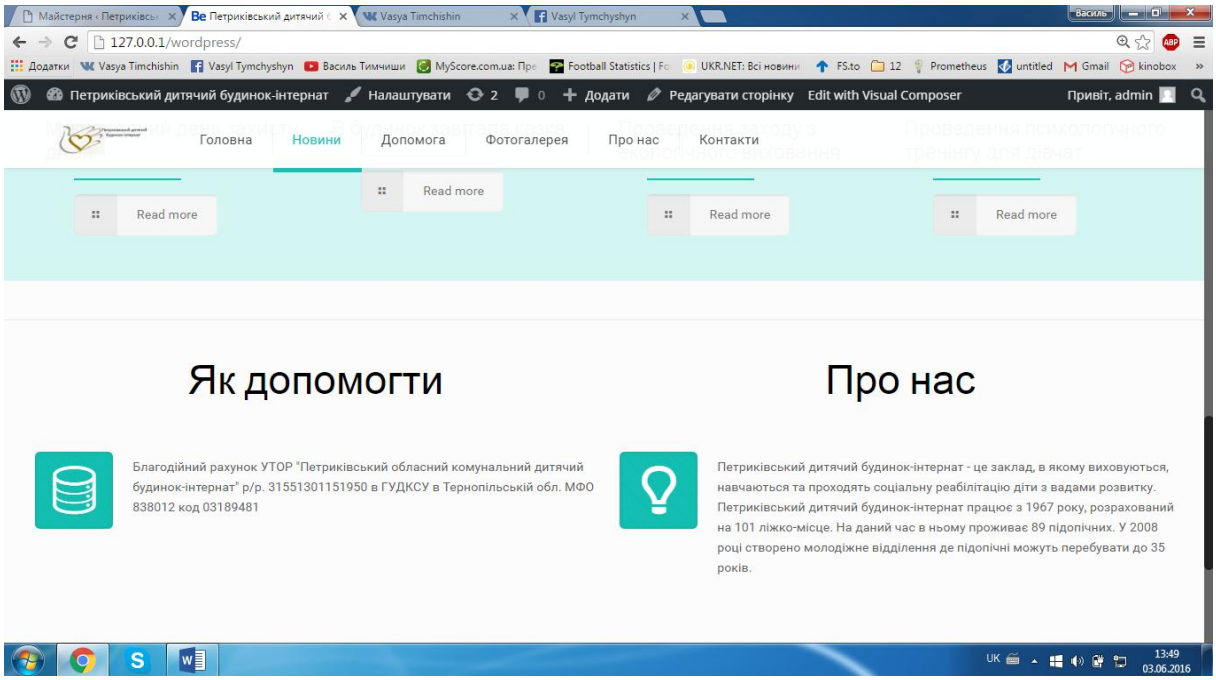

Рисунок 4.4 – Інформація про дитячий будинок та як йому допомогти

На рисунку 4.5 зображеного логотипи компаній, які допомагають дитячому буднику.

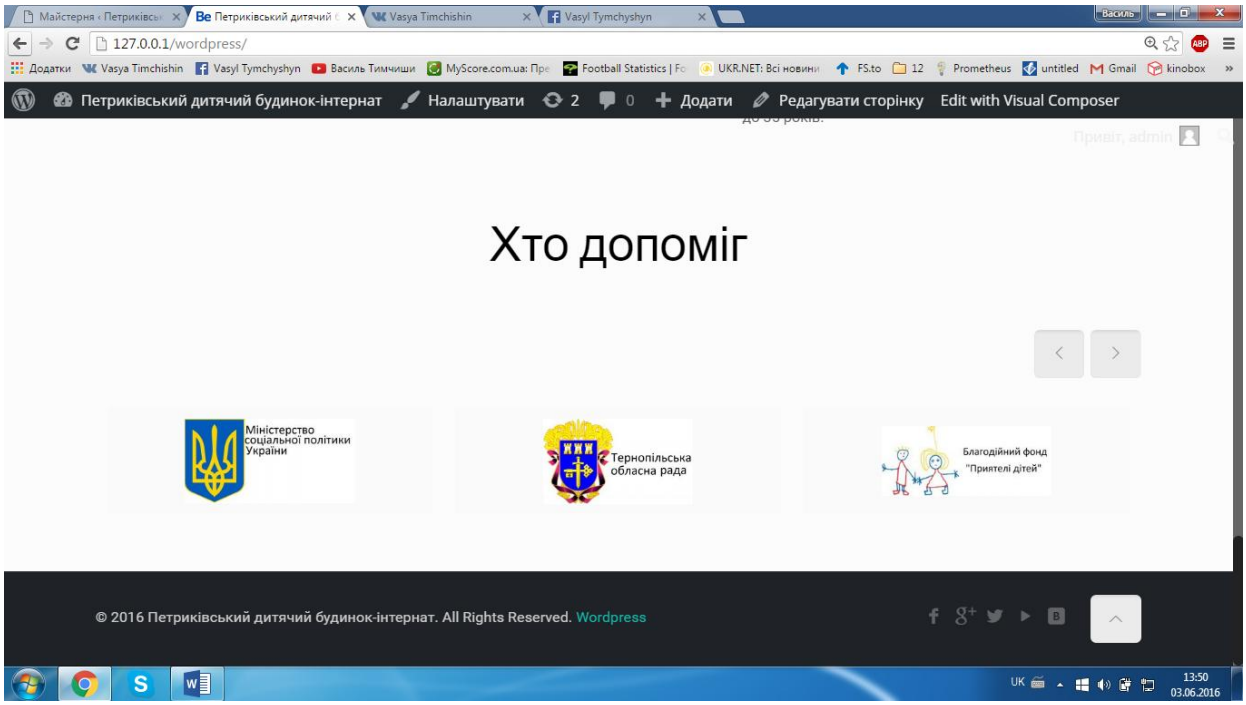

Рисунок 4.5 – Логотипи компаній, які допомогли дитбудинку

Вибравши на вкладці головного меню категорію «Фотогалерея» перед користувачем відкриється сторінка, яка проілюстрована на рисунку 4.6.

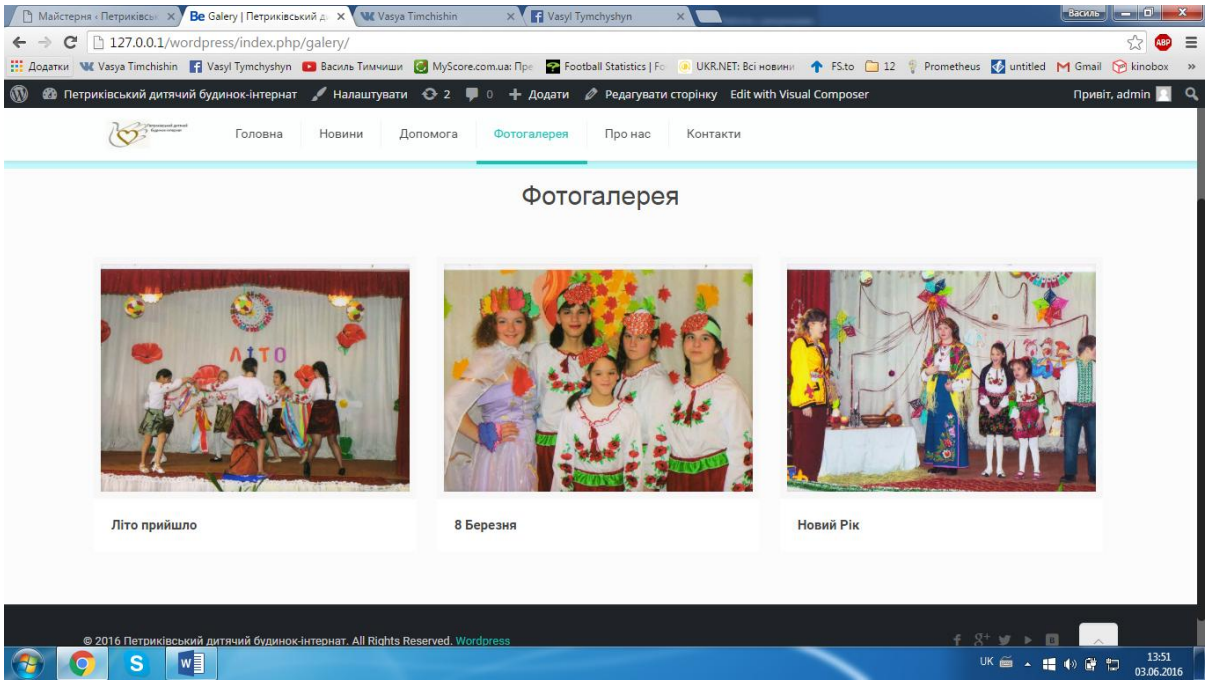

Рисунок 4.6 – Галерея веб-сайту

Вибравши на вкладці головного меню категорію «Контакти» перед користувачем відкриється сторінка, яка зображена на рисунку 4.7. На даній сторінці розміщена вся необхідна інформація для того щоб зв’язатись з представниками дитячого будинку.

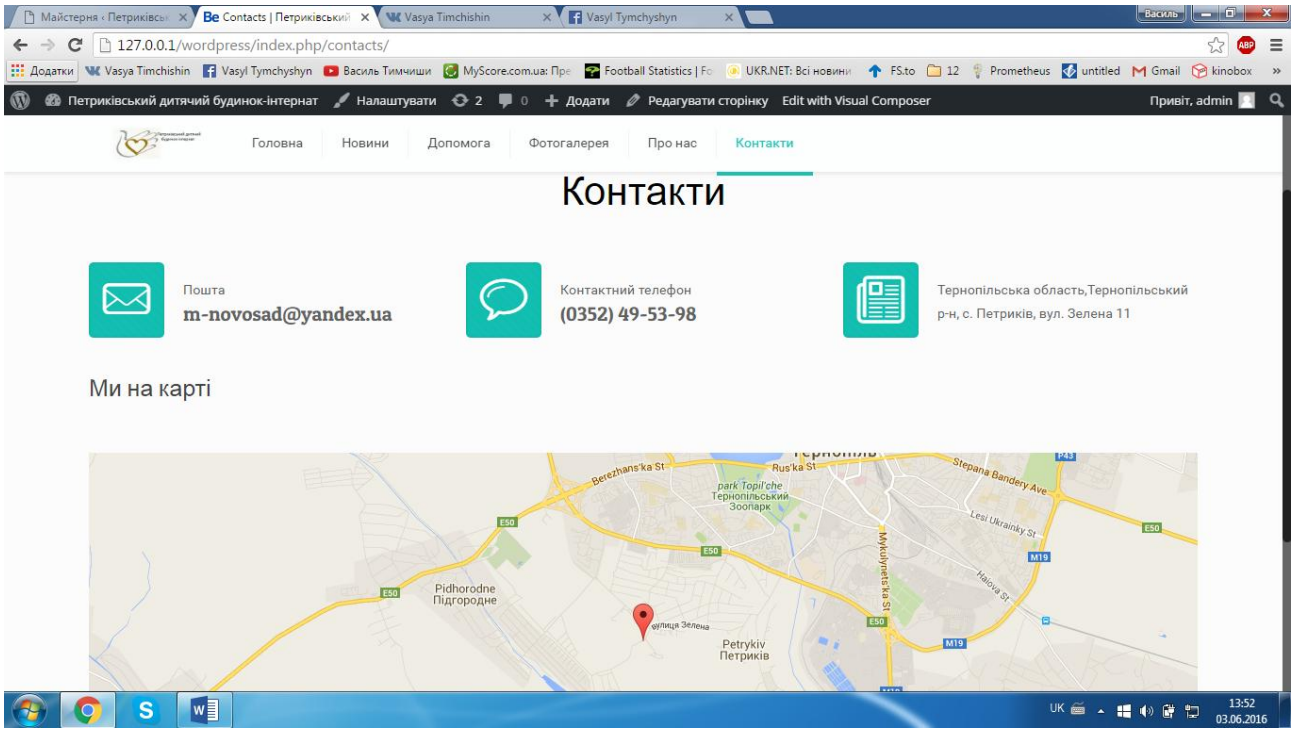

Рисунок 4.7 – Контакти дитячого будинку

Висновки до четвертого розділу:

Було проведено два види тестування: функціональне та графічного інтерфейсу і коротко описано інструкцію користувача.

ВИСНОВКИ

В результаті виконання дипломної роботи було розроблено інтерактивний веб-сайт Петриківського дитячого будинку-інтернату. На етапі аналізу області було досліджено предметну область, що дало можливість виявити вимоги до програмного продукту. З використанням структурного підходу було спроектовано програмну систему розроблено її архітектуру. На етапі програмної реалізації за допомогою технологій HTML5 та CSS3 було

розроблено шаблон веб-сайту. Для зручного управління даними шаблон було підключено системи управління контентом WordPress. На етапі тестування було проведено GUI та функціональне тестування. Створені тест-випадки для кожного виду тестування. Також було розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Валерій Коржов. Багаторівневі системи клієнт-сервер.:СПбГГУ, 2007.

2. Катренко А.В. Системний аналіз об'єктів та процесів комп'ютеризації. Навчальний посібник. Львів, "Новий світ-2000", 2003. 424 с.

3. Основи системного аналізу і проектування АСУ. / Під. ред.А.А.Павлова. - К: Вища школа, 2011.

4. Дмитрий Котеров, Алексей Костарев. PHP. В подлиннике. — Спб.: «БХВ-Петербург», 2005. — С. 1120. — ISBN 5-94157-245-X.

5. Кузнєцов С.Д. Основи сучасних баз даних ISBN.2008. — С. 1104

6. Вієйра Роберт. Програмування баз даних Microsoft SQL Server 2012.Базовий курс: 2008. - 832 с.

7. Денисов А.А., Колесніков Д.М. Теорія великих систем управління. -Л .: Енергоатомідат, 1992.

8. Калянов Г.Н. CASE - структурний системний аналіз. -М.: Лорі 1996. 242.

9. Костарев А. Ф. PHP — Спб.: «БХВ-Петербург», 2008. — С. 1104. — ISBN.

10. Мэтт Зандстра. PHP: объекты, шаблоны и методики программирования, 3-е издание PHP Objects, Patterns and Practice, Third Edition. — М.: «Вильямс», 2010. — С. 560.

11. Кристиан Дари, Эмилиан Баланеску. PHP и MySQL: «Вильямс», 2010.

12. Джейсон Ленгсторф. PHP и jQuery для профессионалов Pro PHP and jQuery. — М.: «Вильямс», 2010. — С. 352.

13. Стив Суэринг, Тим Конверс, Джойс Парк. PHP и MySQL. Библия программиста, 2-е издание PHP 6 and MySQL 6 Bible. — М.: «Диалектика», 2010. — С. 912. — ISBN 978-5-8459-1640-2.

14. Джон Дакетт «Основы веб-программирования с использованием HTML, XHTML и CSS»

15. Канер Кем, Фолк Джек, Нгуен Енг Кек. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. — Киев: ДиаСофт, 2001. — 544 с.

16. Калбертсон Роберт, Браун Крис, Кобб Гэри. Быстрое тестирование. — М.: «Вильямс», 2002. — 374 с.

17. Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения. — М.: БИНОМ, 2008. — 368 с.

18. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. — СПб.: Питер, 2004. — 320 с.

19. Маклаков С.В. BPwin ERwin CASE-средства разработки информационных систем. М.: Диалог-МИФИ, 2001, 304 с.

20. Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения. — М.: БИНОМ, 2008. — 368 с.

Додаток А

Таблиця А.1

Глосарій

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту.	
Веб-сайт	Сукупність веб-сторінок, доступних у мережі Інтернет, які об'єднані як за змістом, так і за

	навігацією. Фізично сайт може розміщуватися як на одному, так і на кількох серверах.
Web-система	Клієнт-серверний додаток, в якому клієнтом виступає браузер, а сервером – веб-сервер.
База даних	Впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.
Авторизація	Керування рівнями та засобами доступу до певного захищеного ресурсу (наприклад, автоматизована система контролю доступу) та ресурсів системи залежно від ідентифікатора і пароля користувача або надання певних повноважень (особі, програмі) на виконання деяких дій у системі обробки даних.
2. Користувачі системи	
Користувач	Простий користувач який може переглядати вміст сайту(системи), але не має права доступу до більшості функцій системи.
Адміністратор	Має повні права доступу до системи її функціональних можливостей, а також маніпуляції над даними, якими оперує система.

Додаток Б

Специфікаця вимог до програмного продукту для «Інтерактивний веб-сервіс «Петриківський дитячий комунальний будинок»»

1. Вступ

1.1 Призначення, мета

Метою веб-орієнтованої системи є популяризація Петриківського дитячого будинку-інтернату в мережі Інтернет.

1.2 Продукти-аналоги

Схожими продуктами є:

1. «Pchelka»;
2. «Тернопільський дитячий будинок»;
3. «Харківський дитячий будинок «Родина»».

2. Загальний опис

2.1 Характеристики продукту

Розроблювана система повинна мати такі функції:

- Перегляд інформації про вихованців дитячого будинку;
- Перегляд інформації про меценатів;
- Додавання нової інформації;
- Видалення застарілої інформації;
- Редагування потрібної інформації;

2.2 Класи користувачів та їх характеристики

У системі присутні два класи користувачів адміністратор та простий користувач. Функціонал даних користувачів відрізняється різними рівнями доступу та можливостей у системі.

Адміністратор керуватиме даними програми з адміністративної панелі. Він матиме можливість додавати, редагувати та видаляти інформацію веб-сайту.

Простим користувачам буде надана можливість тільки переглядти доступну для них інформацію.

2.3 Середовище функціонування

Середовища в яких буде функціонувати продукт – Google Chrome, Mozila FireFox, Opera, Internet Explorer, Safari.

3. Характеристики системи

3.1. Авторизація користувача в системі

3.1.1 Опис і пріоритет

Функціонал, що дає користувачу можливість авторизувавшись на сайті зайти в особистий кабінет.

3.1.2 Послідовності дія/відгук

- Відкрити сторінку авторизації.
- Заповнити авторизаційну форму.
- Натиснути кнопку «Увійти».

3.1.3 Функціональні вимоги

REQ-1.1: обов’язкове заповнення всіх полів;

3.2. Додавання нового запису

3.2.1 Опис і пріоритет

Функція, яка дає можливість додати новий матеріал в систему.

3.2.2 Послідовності дій/відгук

- Вибір потрібного матеріалу.
- Натиснути кнопку «Опублікувати»

3.3. Видалення застарілого матеріалу

3.3.1 Опис і пріоритет

Функція, яка дає можливість видалити вибраний матеріал.

3.3.2 Послідовності дія/відгук

- Вибрати відповідний пост.
- Натиснути кнопку «Перемістити до корзини».

4. Вимоги зовнішніх інтерфейсів

4.1 Користувацькі інтерфейси

Користувачі повинні мати чітке уявлення про користування інтерфейсом, саме тому програма повинна бути для них зрозуміла та візуально нагадувати інтерфейс програм із якими вони звикли працювати, тобто інтерфейс програм для операційної системи Windows версій 7, 8, 8.1, 10.

4.2 Апаратні інтерфейси

Не регламентуються.

5. Інші нефункціональні вимоги

5.1 Вимоги продуктивності

Система має працювати на не надто потужних комп’ютерах із ОЗУ не менше 1 ГБ та двох ядерним процесорам Intel чи AMD.

Відгук системи на усі дії користувача має бути не більшим за чотири секунди.

6. Інші вимоги

Не регламентуються.

Додаток В

Лістинг основних модулів веб-сайту

Header

```
<!DOCTYPE html>
<?php
	if( $_GET && key_exists('mfn-rtl', $_GET) ):
		echo '<html class="no-js" lang="ar" dir="rtl">';
	else:
```

```
?>

<html class="no-js" <?php language_attributes(); ?> <?php mfn_tag_schema(); ?>>

<?php endif; ?>


<!-- head -->

<head>

<!-- meta -->

<meta charset="<?php bloginfo( 'charset' ); ?>" />

<?php if( mfn_opts_get('responsive') ) echo '<meta name="viewport"
content="width=device-width, initial-scale=1, maximum-scale=1">'; ?>


<title itemprop="name"><?php

if( mfn_title() ){

	echo mfn_title();

} else {

	global $page, $paged;

	wp_title( '|', true, 'right' );

	bloginfo( 'name' );

	if ( $paged >= 2 || $page >= 2 ) echo ' | ' . sprintf( __( 'Page %s', 'betheme'
), max( $paged, $page ) );

}

?></title>


<?php do_action('wp_seo'); ?>


<link rel="shortcut icon" href="<?php mfn_opts_show('favicon-img',THEME_URI
.'/images/favicon.ico'); ?>" type="image/x-icon" />


<!-- wp_head() -->

<?php wp_head(); ?>

</head>


<!-- body -->

<body <?php body_class(); ?>>


	<?php do_action( 'mfn_hook_top' ); ?>
```

```
	<?php get_template_part( 'includes/header', 'sliding-area' ); ?>


	<?php if( mfn_header_style( true ) == 'header-creative' ) get_template_part( 'includes/header', 'creative' ); ?>


	<!-- #Wrapper -->

	<div id="Wrapper">


		<?php


			$header_style = '';


			if( mfn_ID() && ! is_search() ){

				if( ( ( mfn_ID() == get_option( 'page_for_posts' ) ) || ( get_post_type() == 'page' ) ) && has_post_thumbnail( mfn_ID() ) ){


					// Pages & Blog Page ---

					$subheader_image = wp_get_attachment_image_src( get_post_thumbnail_id( mfn_ID() ), 'full' );

					$header_style .= ' style="background-image:url('. $subheader_image[0] .');"';


				} elseif( get_post_meta( mfn_ID(), 'mfn-post-header-bg', true ) ){


					$header_style .= ' style="background-image:url('. get_post_meta( mfn_ID(), 'mfn-post-header-bg', true ) .');"';


				}

			}


			if( mfn_opts_get('img-subheader-attachment') == 'fixed' ){

				$header_style .= ' class="bg-fixed"';

			} elseif( mfn_opts_get('img-subheader-attachment') == 'parallax' ){

				$header_style .= ' class="bg-parallax" data-stellar-background-ratio="0.5"';

			}

		?>
```

```
		<?php if( mfn_header_style( true ) == 'header-below' ) echo mfn_slider();
?>

		<!-- #Header_bg -->
		<div id="Header_wrapper" <?php echo $header_style; ?>>

			<!-- #Header -->
			<header id="Header">
				<?php if( mfn_header_style( true ) != 'header-creative' )
get_template_part( 'includes/header', 'top-area' ); ?>
				<?php if( mfn_header_style( true ) != 'header-below' ) echo
mfn_slider(); ?>
			</header>

			<?php
				if( ( mfn_opts_get('subheader') != 'all' ) && !
get_post_meta( mfn_ID(), 'mfn-post-hide-title', true ) ){

					if( is_search() ){

						// Page title --------------------------
						echo '<div id="Subheader">';
							echo '<div class="container">';
								echo '<div class="column one">';

									global $wp_query;
									$total_results = $wp_query-
>found_posts;

									$translate['search-results']
= mfn_opts_get('translate') ? mfn_opts_get('translate-search-results','results found
for:') : __('results found for:','betheme');

									echo '<h1 class="title">'.
$total_results .' '. $translate['search-results'] .' '. get_search_query() .'</h1>';

								echo '</div>';
							echo '</div>';
```

```
                                    echo '</div>';

                                } elseif( ! mfn_slider() || mfn_opts_get( 'subheader-
slider-show' ) ){

      // Page title ------------------------
            echo '<div id="Subheader">';
            echo '<div class="container">';
            echo '<div class="column one">';

      // Title
echo '<h1 class="title">'. mfn_page_title() .'</h1>';

                                                        if( !
mfn_opts_get('subheader') ) mfn_breadcrumbs();

                                                    echo '</div>';
                                                echo '</div>';
                                            echo '</div>';

                                }

                            }
                    ?>

            </div>
            <?php do_action( 'mfn_hook_content_before' ); ?>\
```

Footer

```
<?php do_action( 'mfn_hook_content_after' ); ?>

<!-- #Footer -->
<footer id="Footer" class="clearfix">

      <?php if ( $footer_call_to_action = mfn_opts_get('footer-call-to-action') ): ?>
```

```
	<div class="footer_action">
		<div class="container">
			<div class="column one column_column">
				<?php echo $footer_call_to_action; ?>
			</div>
		</div>
	</div>
	<?php endif; ?>

	<?php
		$sidebars_count = 0;
		for( $i = 1; $i <= 4; $i++ ){
			if ( is_active_sidebar( 'footer-area-'. $i ) ) $sidebars_count++;
		}

		if( $sidebars_count > 0 ){
			echo '<div class="widgets_wrapper">';
				echo '<div class="container">';

					if( $footer_layout = mfn_opts_get( 'footer-layout' ) ){
						// Theme Options

						$footer_layout	= explode( ';', $footer_layout );
						$footer_cols	= $footer_layout[0];

						for( $i = 1; $i <= $footer_cols; $i++ ){
							if ( is_active_sidebar( 'footer-area-'. $i ) ){

								echo '<div class="column '. $footer_layout[$i] .'">';

									dynamic_sidebar( 'footer-area-'. $i );

								echo '</div>';
							}
						}
```

```
                        } else {
                                // Default - Equal Width

                                $sidebar_class = '';
                                switch( $sidebars_count ){
                                case 2: $sidebar_class = 'one-second'; break;
                                case 3: $sidebar_class = 'one-third'; break;
                                case 4: $sidebar_class = 'one-fourth'; break;
                                default: $sidebar_class = 'one';
                                }

                                for( $i = 1; $i <= 4; $i++ ){
                                if ( is_active_sidebar( 'footer-area-'. $i ) ){
                        echo '<div class="column '. $sidebar_class .'">';
                                        dynamic_sidebar( 'footer-area-'. $i );
                                                echo '</div>';
                                        }
                                }

                        }

                                echo '</div>';
                        echo '</div>';
                }
        ?>

        <?php if( mfn_opts_get('footer-hide') != 1 ): ?>
        <div class="footer_copy">
                <div class="container">
                        <div class="column one">
                                <a id="back_to_top" href="" class="button button_left
button_js"><span class="button_icon"><i class="icon-up-open-big"></i></span></a>

                                <!-- Copyrights -->
                                <div class="copyright">
                                        <?php
```

```
                                        echo '&copy; '. date( 'Y' ) .' '. get_bloginfo(
'name' ) .'. All Rights Reserved. <a target="_blank" rel="nofollow"
href="http://mafiashare.net">Wordpress</a>';

                                    ?>
                            </div>

                            <?php
                                    if( has_nav_menu( 'social-menu-bottom' ) ){

                                        // #social-menu
                                        mfn_wp_social_menu_bottom();

                                    } else {

                        $target = mfn_opts_get('social-target') ? 'target="_blank"' : false;

                                        echo '<ul class="social">';
                                        echo '</ul>';

                                    }
                            ?>

                        </div>
                    </div>
                </div>
                <?php endif; ?>

</footer>

</div><!-- #Wrapper -->

<?php if( mfn_opts_get('popup-contact-form') ): ?>
        <div id="popup_contact">
            <a class="button button_js" href="#"><i class="icon-mail-line"></i></a>
```

```
            <div class="popup_contact_wrapper">
                  <?php echo do_shortcode( mfn_opts_get('popup-contact-form') ); ?>
                  <span class="arrow"></span>
            </div>
      </div>
<?php endif; ?>

<?php do_action( 'mfn_hook_bottom' ); ?>

<!-- wp_footer() -->
<?php wp_footer(); ?>

</body>
</html>
```

Sidebar

```
$sidebars = mfn_opts_get( 'sidebars' );

if( get_post_type() == 'page' && mfn_opts_get('single-page-sidebar') ){

      // Theme Options | Single - Page
      $sidebar = trim( mfn_opts_get('single-page-sidebar') );

} elseif( get_post_type() == 'post' && is_single() && mfn_opts_get('single-sidebar') ){

      // Theme Options | Single - Post
      $sidebar = trim( mfn_opts_get('single-sidebar') );

} elseif( get_post_type() == 'portfolio' && is_single() && mfn_opts_get('single-
portfolio-sidebar') ){

      // Theme Options | Single - Portfolio
      $sidebar = trim( mfn_opts_get('single-portfolio-sidebar') );

} else {
```

```
	// Post Meta
	$sidebar = get_post_meta( mfn_ID(), 'mfn-post-sidebar', true);
	if( $sidebar || $sidebar === '0' ) $sidebar = $sidebars[$sidebar];

}

if( $_GET && key_exists('mfn-s', $_GET) ) $sidebar = $_GET['mfn-s']; // demo


// sidebar 2 ----------------------------------------------------
$sidebar2 = get_post_meta( mfn_ID(), 'mfn-post-sidebar2', true);
if( $sidebar2 || $sidebar2 === '0' ) $sidebar2 = $sidebars[$sidebar2];

if( $_GET && key_exists('mfn-s2', $_GET) ) $sidebar2 = $_GET['mfn-s2']; // demo

if( mfn_sidebar_classes() ){

	echo '<div class="sidebar sidebar-1 four columns">';
		echo '<div class="widget-area clearfix '. mfn_opts_get('sidebar-lines')
.'">';
			if ( ! dynamic_sidebar( $sidebar ) ) mfn_nosidebar();
		echo '</div>';
	echo '</div>';

	// both sidebars
	if( mfn_sidebar_classes( true ) ){
		echo '<div class="sidebar sidebar-2 four columns">';
			echo '<div class="widget-area clearfix '. mfn_opts_get('sidebar-
lines') .'">';
				if ( ! dynamic_sidebar( $sidebar2 ) ) mfn_nosidebar();
			echo '</div>';
		echo '</div>';
	}
}

?>
```

Index.php

```
get_header();
$blog_classes = array();

// layout
if( $_GET && key_exists('mfn-b', $_GET) ){
	$blog_layout = $_GET['mfn-b']; // demo
} else {
	$blog_layout = mfn_opts_get( 'blog-layout', 'classic' );
}
$blog_classes[] = $blog_layout;

// isotope
if( $blog_layout=='masonry' ) $blog_classes[] = 'isotope';

// ajax load more
$load_more = mfn_opts_get('blog-load-more');

$translate['filter']		= mfn_opts_get('translate') ? mfn_opts_get('translate-filter','Filter by') : __('Filter by','betheme');
$translate['tags']		= mfn_opts_get('translate') ? mfn_opts_get('translate-tags','Tags') : __('Tags','betheme');
$translate['authors']		= mfn_opts_get('translate') ? mfn_opts_get('translate-authors','Authors') : __('Authors','betheme');
$translate['all']		= mfn_opts_get('translate') ? mfn_opts_get('translate-all','Show all') : __('Show all','betheme');
$translate['categories']	= mfn_opts_get('translate') ? mfn_opts_get('translate-categories','Categories') : __('Categories','betheme');
?>

<!-- #Content -->
<div id="Content">
	<div class="content_wrapper clearfix">

		<!-- .sections_group -->
		<div class="sections_group">
```

```
                    <div class="extra_content">
                          <?php
                                if( category_description() ){
                                      echo '<div class="section the_content
category_description">';
                                            echo '<div class="section_wrapper">';
                                                  echo '<div
class="the_content_wrapper">';
                                                        echo category_description();
                                                  echo '</div>';
                                            echo '</div>';
                                      echo '</div>';
                                } else {
                                      mfn_builder_print( mfn_ID(), true );
                                }
                          ?>
                    </div>

                    <div class="section">
                          <div class="section_wrapper clearfix">


                                <!-- #Filters -->
                                <?php if( ! is_singular() && get_post_type()=='post' &&
get_option( 'page_for_posts' ) ): ?>
                                      <div id="Filters" class="column one <?php if(
$blog_layout=='masonry' ) echo 'isotope-filters'; ?>">


                                            <ul class="filters_buttons">
<li class="label"><?php echo $translate['filter']; ?></li>
<li class="categories"><a class="open" href="#"><i class="icon-docs"></i><?php echo
$translate['categories']; ?><i class="icon-down-dir"></i></a></li>
<li class="tags"><a class="open" href="#"><i class="icon-tag"></i><?php echo
$translate['tags']; ?><i class="icon-down-dir"></i></a></li>
<li class="authors"><a class="open" href="#"><i class="icon-user"></i><?php echo
$translate['authors']; ?><i class="icon-down-dir"></i></a></li>
<li class="reset"><a class="close" data-rel="*" href="<?php echo
get_permalink(mfn_ID()); ?>"><i class="icon-cancel"></i><?php echo $translate['all'];
?></a></li>
```

```
                                        </ul>


                                        <div class="filters_wrapper">
                                              <ul class="categories">
                                                    <?php
                                       if( $categories = get_categories() ){
                                              foreach( $categories as $category )
{echo '<li><a data-rel=".category-'. $category->slug .'" href="'.
get_term_link($category) .'">'. $category->name .'</a></li>';
                                                                        }
                                                                  }
                                                    ?>
            <li class="close"><a href="#"><i class="icon-cancel"></i></a></li>
                                        </ul>
                                        <ul class="tags">
                                              <?php
                                                          if( $tags = get_tags()
){
                                        foreach( $tags as $tag ){
echo '<li><a data-rel=".tag-'. $tag->slug .'" href="'. get_tag_link($tag) .'">'. $tag-
>name .'</a></li>';
                                                                        }
                                                                  }
                                                    ?>
      <li class="close"><a href="#"><i class="icon-cancel"></i></a></li>
                                        </ul>
                                        <ul class="authors">
                                              <?php
                                                          $authors = get_users();
                                        if( is_array($authors) )
                  foreach( $authors as $ka => $author ){
                              $remove = true;


                  if( in_array( 'author', $author->roles ) ) $remove = false;
      if( in_array( 'editor', $author->roles ) ) $remove = false;


            if( in_array( 'administrator', $author->roles ) ) $remove = false;
```

```
            ( $remove ) unset( $authors[$ka] );
                                                                    }

                                                foreach( $authors as $auth ){
echo '<li><a data-rel=".author-'. $auth->data->user_login .'" href="'.
get_author_posts_url($auth->ID) .'">'. $auth->data->display_name .'</a></li>';

                                                                    }

                                                                }
                                                            ?>
<li class="close"><a href="#"><i class="icon-cancel"></i></a></li>
                                                        </ul>
                                                </div>

                                        </div>
                                    <?php endif; ?>

                                    <div class="column one column_blog">
                                        <div class="blog_wrapper isotope_wrapper">

<div class="posts_group lm_wrapper <?php echo implode(' ', $blog_classes); ?>">
<?php echo mfn_content_post( false, false, $load_more ); ?>
                                                </div>

                                                <?php
                                                    // pagination
if( function_exists( 'mfn_pagination' ) ):
echo mfn_pagination( false, $load_more );

                                                else:
                                                    ?>

                                                                <div class="nav-
next"><?php next_posts_link(__('&larr; Older Entries', 'betheme')) ?></div>
                                                                <div class="nav-
previous"><?php previous_posts_link(__('Newer Entries &rarr;', 'betheme')) ?></div>
```

```
                                                    <?php
                                                endif;
                                            ?>

                                        </div>
                                    </div>

                                </div>
                            </div>

                        </div>



                        <!-- .four-columns - sidebar -->
                        <?php get_sidebar( 'blog' ); ?>

                </div>
        </div>

        <?php get_footer(); ?>
```

Content.post

```
if( ! function_exists('mfn_content_post') ){
        function mfn_content_post( $query = false, $layout = false, $load_more = false ){
                global $wp_query;
                $output = '';


                $translate['published']   = mfn_opts_get('translate') ?
mfn_opts_get('translate-published','Published by') : __('Published by','betheme');
                $translate['at']                = mfn_opts_get('translate') ?
mfn_opts_get('translate-at','at') : __('at','betheme');
                $translate['categories']  = mfn_opts_get('translate') ?
mfn_opts_get('translate-categories','Categories') : __('Categories','betheme');
                $translate['like']              = mfn_opts_get('translate') ?
mfn_opts_get('translate-like','Do you like it?') : __('Do you like it?','betheme');
```

```
            $translate['readmore']          = mfn_opts_get('translate') ?
mfn_opts_get('translate-readmore','Read more') : __('Read more','betheme');


            if( ! $query ) $query = $wp_query;


            if ( $query->have_posts() ){
                  while ( $query->have_posts() ){
                        $query->the_post();


                        $post_class =  array('post-item','isotope-item','clearfix');
                        if( ! mfn_post_thumbnail( get_the_ID() ) ) $post_class[] =
'no-img';
                        if( post_password_required() ) $post_class[] = 'no-img';
                        $post_class[] = 'author-'. get_the_author_meta( 'user_login'
);
                        $post_class = implode(' ', get_post_class( $post_class ));


                        $output .= '<div class="'. $post_class .'">';


                              $output .= '<div class="date_label">'. get_the_date()
.'</div>'; // style: timeline




// photo ---------------------------------------------------------------------
                              if( ! post_password_required() ){
                                    $output .= '<div class="image_frame post-photo-
wrapper scale-with-grid">';
                                          $output .= '<div class="image_wrapper">';
                                                $output .= mfn_post_thumbnail(
get_the_ID(), false, false, $load_more );
                                          $output .= '</div>';
                                    $output .= '</div>';
                              }




// desc ----------------------------------------------------------------------
```

```
                                $output .= '<div class="post-desc-wrapper">';
                                        $output .= '<div class="post-desc">';

                                // meta ------------------------------------
                                            if( mfn_opts_get( 'blog-meta' ) ){


$output .= '<div class="post-meta clearfix">';


                                                                $output .= '<div
class="author-date">';


$output .= '<span class="vcard author post-author">';


                                                $output .= '<span class="label">'.
$translate['published'] .' </span>';
                                                                            $output .= '<i
class="icon-user"></i> ';
$output .= '<span class="fn"><a href="'. get_author_posts_url( get_the_author_meta(
'ID' ) ) .'">'. get_the_author_meta( 'display_name' ) .'</a></span>';
                                $output .= '</span> ';


                            $output .= '<span class="date">';
                                                                            $output .=
'<span class="label">'. $translate['at'] .' </span>';
                                                                            $output .= '<i
class="icon-clock"></i> ';
                                                                            $output .=
'<span class="post-date updated">'. get_the_date() .'</span>';
                                                                        $output .= '<span
class="date"></span>';


                                                                $output .= '</div>';


                                                                $output .= '<div
class="category">';
                                                                        $output .= '<span
class="cat-btn">'. $translate['categories'] .' <i class="icon-down-dir"></i></span>';
```

```
                                                $output .= '<div
class="cat-wrapper">'. get_the_category_list() .'</div>';
                                                $output .= '</div>';

                                        $output .= '</div>';
                                }

                        // title -----------------------------------
                                $output .= '<div class="post-title">';

                                        if( get_post_format() == 'quote' ){


                                        $output .= '<blockquote><a href="'.
get_permalink() .'">'. get_the_title() .'</a></blockquote>';

                                        } elseif( get_post_format() ==
'link' ){


                                                $output .= '<i class="icon-
link"></i>';
                                                $output .= '<div class="link-
wrapper">';
                                                        $output .= '<h4>'.
get_the_title() .'</h4>';
                                                        $link =
get_post_meta(get_the_ID(), 'mfn-post-link', true);
                                                        $output .= '<a
target="_blank" href="'. $link .'">'. $link .'</a>';
                                                $output .= '</div>';

                                        } else {

                                                $output .= '<h2 class="entry-
title" itemprop="headline"><a href="'. get_permalink() .'">'. get_the_title()
.'</a></h2>';
                                        }

                                $output .= '</div>';
```

```
                                // content -------------------------------------
                                        $output .= '<div class="post-excerpt">'.
get_the_excerpt() .'</div>';

                                // footer --------------------------------------
                                        $output .= '<div class="post-footer">';



                                                $output .= '<div class="button-
love"><span class="love-text">'. $translate['like'] .'</span>'. mfn_love() .'</div>';

                                                $output .= '<div class="post-
links">';

            if( comments_open() ) $output .= '<i class="icon-comment-empty-fa"></i> <a
href="'. get_comments_link() .'" class="post-comments">'. get_comments_number()
.'</a>';

                                                        $output .= '<i class="icon-
doc-text"></i> <a href="'. get_permalink() .'" class="post-more">'.
$translate['readmore'] .'</a>';

                                                $output .= '</div>';


                                        $output .= '</div>';


                                    $output .= '</div>';

                                $output .= '</div>';


                            $output .= '</div>';


                }



//                  $output .= '<div class="TILIM">'. $wp_query->max_num_pages
.'</div>';

            }


            return $output;

      }
```

```
}
function mfn_meta_field_input( $field, $meta ){
		global $MFN_Options;

		if( isset( $field['type'] ) ){
			echo '<tr valign="top">';

				// Field Title & SubDescription
				echo '<th scope="row">';
					if( key_exists('title', $field) ) echo $field['title'];
					if( key_exists('sub_desc', $field) ) echo '<span
class="description">'. $field['sub_desc'] .'</span>';
				echo '</th>';

				// Muffin Options Field & Description
				echo '<td>';
					$field_class = 'MFN_Options_'.$field['type'];
					require_once( $MFN_Options-
>dir.'fields/'.$field['type'].'/field_'.$field['type'].'.php' );

					if( class_exists( $field_class ) ){
						$field_object = new $field_class( $field, $meta
);
						$field_object->render( true );
					}

				echo '</td>';

			echo '</tr>';
		}
	}
}if( $section_sidebar ){
			echo '<div class="four columns section_sidebar">';
			echo '<div class="widget-area clearfix">';
			dynamic_sidebar( $sidebars[$section['attr']['sidebar']] );
                                        echo '</div>';
                                    echo '</div>';
```

```
                }

            echo '</div>';
        echo '</div>';
?>
```