Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Дорофеєва Олександра Андрійовича*
(ПІБ)

академічної групи *121-18-2*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: ***Розробка веб-додатка платформи аграрних знань за допомогою php 7.4 та Symfony 4.***

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	*проф. Бердник М.Г*			
розділів:				
спеціальний	*проф. Бердник М.Г.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	доц.І.А.Шедловський			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-18-2* *Дорофеєв О.А.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи ***Розробка веб-додатка платформи аграрних знань за допомогою php 7.4 та Symfony 4.***

затверджена наказом ректора НТУ «ДП» від *18.05.2022 р.* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *проф. Бердник М.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Дорофеєв О.А.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 87 с., 6 рис., 0 табл., 1 дод., 25 джерел.

Об'єкт розробки: web-додаток для навчання у сфері аграрного бізнесу.

Мета кваліфікаційної роботи: створення web-додатку для покращення навичок працівників аграрного бізнесу, підвищення прибутків компаній або окремих осіб, знаходження партнерів та обміну навичкми.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні web-додатка, що надає можливість електронного зберігання даних про інформаційний блок, ведення бази даних, підбиття підсумків з проведених курсів та вебінарів, отримання звітів і формування супутньої ділової документації.

Актуальність даного програмного продукту визначається поширенням аграрної сфери в усіх куточках країни, покращенням комунікації та попитом на знання у зацікавлених цією сферою людей.

Список ключових слів: FRAMEWORK, ПАТЕРН, DOCKER, WEB-ДОДАТОК, MODEL, VIEW, CONTROLLER, АДМІНІСТРАТИВНА ПАНЕЛЬ, DATABASE

ABSTRACT

Explanatory note: 87 pages, 6 figures, 0 tables, 1 appendix, 25 sources.

Object of development: web-application for training in the field of agrarian business.

The purpose of the qualification work: to create a web-application to improve the skills of agricultural workers, increase the profits of companies or individuals, find partners and share skills.

The introduction considers the analysis and current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the subject branch is analyzed, the urgency of the task and the purpose of development are determined, the statement of the task is formulated, the requirements to the software implementation, technologies and software are specified.

The second section analyzes the available solutions, selected platforms for development, designed and developed the program, describes the program, algorithm and structure of its operation, as well as calling and loading the program, determines the input and output data, describes the parameters of hardware.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

Of practical importance is the creation of a web-application that provides electronic storage of information about the information unit, database maintenance, summarizing the courses and webinars, receiving reports and the formation of related business documents.

The relevance of this software product is determined by the expansion of the agricultural sector in all parts of the country, improving communication and the demand for knowledge from people interested in this field.

Keyword list: FRAMEWORK, PATTERN, DOCKER, WEB APP, MODEL, VIEW, CONTROLLER, ADMINISTRATIVE PANEL, DATABASE

ЗМІСТ

ВСТУП

Тематика даної кваліфікаційної роботи присвячена розробці навчального web-додатку в сфері аграрної діяльності.

Аграрна промисловість є вкрай важливою для України в умовах європейської інтеграції. Відповідно до Конституції України, «земля є основним національним багатством, яке перебуває під особливою охороною держави» (стаття 14).

За своїми фізичними, мінералогічними, хімічними, агрохімічними властивостями українські чорноземи вважаються найкращими у світі. В Україні чорноземи займають площу 60,4 млн. га, з них майже 42 млн. га (близько 69%) припадає на сільгоспугіддя. Причому 78% сільгоспугідь — рілля. Іншими словами, розорано в країні близько 33 млн. га чорноземів. У різних джерелах цифри різняться, але незначно.

Експерти стверджують, що Україна може забезпечити повноцінним харчуванням приблизно 500 млн. людей. Цьому сприяють і кліматичні умови: досить м'яка зима (середня температура в різних регіонах коливається від -4°C до -12°C), помірно тепле літо (від +18°C до +25°C) і близько 230 сонячних днів на рік створюють практично ідеальні умови вирощування всіх основних сільгоспкультур.

Основною метою проекту є підвищення прибутків малих та середніх робітників аграрної сфери діяльності.

Проект допомагає збільшити прибуток за рахунок впровадження інноваційних технологій і методів ведення бізнесу, які економлять час і зусилля, а також є екологічно чистими.

Результатом виконання даної роботи є web-додаток, що значно спрощує отримання знань за рахунок великого вибору видів навчальних матеріалів, комунікацію робітників аграрної сфери, а також пошук партнерів та однодумців.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDE - Integrated Development Environment.

HTML - Гіпертекстова мова розмітки для створення веб-сторінок.

CSS - Cascading Style Shee

ОС - Операційні системи.

БД - Бази даних.

MVC - Model View Controller.

JS – JavaScript.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості з предметної галузі

Аграрний бізнес дуже поширений на території України за рахунок її плодородючих заемль. Для підвищення ефективності його діяльності потребується:

1. Розвиток ринку:

- розширення можливостей створення додаткової вартості продукції;
- залучення малих і середніх фермерів на високопродуктивні ринки шляхом інтеграції ринку;
- удосконалювання методів та технологій виробництва та розвит оквідносин з постачальниками.

2. Розвиток трудових ресурсів:

- доведення рівня університетської освіти до вимог бізнесу;
- покращення та розвиток навичок випускників університетів до рівня, якого вимагає плодоовочевий сектор.

3. Доступ до фінансових ринків:

поліпшення доступу до інвестицій та фінансових можливостей шляхом створення зв'язків з інвестиційними компаніями та компаніями з приватним капіталом.

4. Використання інноваційного фонду:

- створення фінансових стимулів для просування новітніх технологій і можливостей;
- проведення конкурсу бізнес планів з зустрічним фінансуванням на їх впровадження;
- надання фінансових гарантій структурам, які будуть заохочувати інвестиції у життєздатні середні підприємства та фінансувати сільськогосподарське виробництво плодоовочевої продукції.

1.2 Призначення розробки та галузь застосування

Web-додаток аграрної навчальної платформи призначений для облегшення користувачам пошуку корисної інформації, нових знайомст у сфері бізнесу та партнерів. Платформа надає обширні можливості пошоку різноманітних знань в аграрній сфері, а також має велику кількість експертів та партерів з якими можна зв’язатися при потребі.

Додаток навчальної аграрної платформи стане в нагоді усюди де є розвиток землеробства. Він допоможе не тільки початківця у покращені своїх знань, а й робітникам з досвідом які хочуть розвивати навички в аграрном бізнесі далі.

Завдяки обширним можливостям платформи, вона поширюється на більшісь популярних галузів виробництво, на приклад овочівництво, бджільництво, еко – технології та інші.

Можна зробити висновок, що навчальна аграрна платформа буде корисна у веливій вкількості різноманітних стрін аграрного бізнесу.

1.3 Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Підставами для розробки (виконання кваліфікаційної роботи) є:

- освітня програма 121 Інженерія програмного забезпечення;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18.05.2022 р;
- завдання для кваліфікаційної роботи на тему: «Розробка веб-додатка платформи аграрних знань за допомогою php 7.4 та Symfony 4.».

1.4 Постановка завдання

Метою дипломної роботи є розробка й створення на основі сучасних комп’ютерних технологій веб-додатка платформи аграрних знань.

Завдання:

1. Розробити веб-додаток з унікальним контентом в сфері аграрного навчання.
2. Розробити зручний доступ донавчальної платорми.
3. Розробити адміністративну сторінку порталу.

Розроблюваний веб-додаток повинен забезпечувати виконання таких функцій:

— надання доступу до інформаційно-навчальних публікацій;

— здійснення зручного і швидкого доступу до кабінету користувача;

— можливість здійснення пошуку інформаційно-навчальних публікацій за ключовими словами;

— авторизація на сторінці адміністратора;

— розміщення публікації на сайті.

1.5 Вимоги до програми або програмного виробу

Гармонійний та грамотний додаток працює на компанію та приваблює клієнтів чи покупців. Вимоги до створення сайту визначають напрямок організації та контингент користувачів. Виділяються моменти, що належать до ресурсів, незалежно від особливостей. Серед них:

- приємний дизайн, що запам'ятовується;
- зручний інтерфейс;
- навігація;
- наявність посилань;
- реєстрація;
- пошук;
- контакти;
-відображення в будь-якому браузері та за допомогою різних пристроїв, висока конверсія.

Ці параметри зроблять сайт таким, що запам'ятовується для користувача, який захоче звертатися до ресурсу.

Щоб продукт мав ці характеристики, потрібно правильно і професійно підійти до створення. Вимоги до створення сайту залежать від різновиду ресурсу. Так, сайти компаній, що пропонують конкретні послуги споживачеві, мають контакти та замовлення зворотного зв'язку. Інтернет-магазини та організації, що спеціалізуються на онлайн-продажах, активніше використовуються за наявності в них калькулятора та порівняння товарів та цін. У цьому полягає зручність поводження з продуктом. Така практика стає запорукою повторного звернення та позитивних відгуків покупців.

Дизайн визначається спрямованістю роботи. Корпоративні сайти прийнято витримувати у фірмовому діловому стилі компанії. Логотипи, новини, інформація про рекламні акції — те, що переважає у таких типах ресурсів. Тут важливим моментом стає текст, з мінімумом графіки та зображень.

Сайт-візитку вирізняє простий дизайн, який відповідає за функціональність. Значок це дія. Наочність та відданість тематики – запорука успішної роботи сайту та компанії.

Зручність сприйняття інформації визначає різновид верстки. Блоки та таблиці наочні користувачам, але викликають проблеми під час завантаження сторінки.

Фіксована та гумова верстка торкаються моменту відображення інформації на конкретному моніторі.

При розробці сайтів, адаптованих під мобільні програми, використовується адаптивна верстка.

Визначити актуальність верстки під силу професіоналам, яким варто довірити розробку.

Ключовим моментом сайту є інформація, якою він наповнений. Унікальний контент, адаптований під роботу пошукових систем та корисний для людей, — ось те, що зробить ресурс рентабельним та популярним серед користувачів. Текстове наповнення стає визначальним під час виведення сайту в топ. Епоха статей, орієнтованих на роботів, пішла в минуле. Сьогодні лише корисна інформація виводить ресурс на вагомі позиції в пошукових системах.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

Будь-який сайт або веб-додаток створюється з конкретною метою, він «заточений» під певні функції.

Веб-додаток має в собі декілька функцій:

1. Інформаційна.

Це основна функція: користувачі приходять до мережі за інформацією. Популярність будь-якого проекту залежить від того, наскільки повно і змістовно він її надасть, наскільки матеріали веб-ресурсу актуальні, як часто оновлюються.

Інформація, що розміщується на сайтах, включає:

- загальні дані про компанію чи власника (на авторських проектах);
- портфоліо - своєрідний звіт про виконані роботи;
- опис продукції, послуг;
- блок опитувань для збирання цінних відомостей від клієнтів;
- інформаційні статті на допомогу споживачам (ось навіщо потрібен блог на сайті комерційної спрямованості);
- інформаційна інформація для зв'язку: контакти, адреса, карта проїзду;
- відгуки партнерів та споживачів;
- перелік партнерів, найкращих клієнтів;
- контактну форму для швидкого зворотного зв'язку;
- інформаційний блог для публікації новин та корисних статей;
- сертифікати, ліцензії, іншу документацію, що підтверджує надійність та експертність.

Вирішальні чинники успішності проекту: як швидко користувачі одержують необхідну інформацію; у вигляді вона представлена; її простота та доступність для будь-якого споживача. Від цього залежить рішення прочитати текст до кінця, зв'язатися з менеджерами, зателефонувати, виконати іншу очікувану дію.

Наскільки повно компанія/власник особистого проекту заявлять про себе, знайдуть можливість залучити та утримати відвідувачів, залежить успіх та розкрутка проекту, вибір способів просування сайту.

Частина відповідальності за інформацію залежить від якості текстів, які написали копірайтери для веб-сайту/додатку.

2. Іміджева функція.

Друга за значимістю. Від якісності створення власного іміджу залежить відношення партнерів та споживачів. Вони сформують позитивне чи негативне ставлення до веб-ресурсу компанії або особистому веб-сайту.

На імідж впливає безліч деталей: розкрученість у соціальних мережах, часто згадка бренду в інтернеті. Знову відповідальність за іміджеві тексти або пости для соціальних мереж лягає на плечі копірайтерів, результат їх роботи сприяє просуванню сайту статтями.

3. Комунікативна функція.

Необхідно надати відвідувачам можливість спілкування між собою та з адміністрацією сайту (представниками компанії):

- зручність для контактів, великий вибір можливості: зателефонувати, написати на електронну пошту, скористатися онлайн-консультантом, зв'язатися з менеджером, приїхати в офіс;

- своєчасний відгук на коментарі користувачів. Зазвичай запереченнями чи виправданнями відповідають негативні відгуки, а позитивні залишають без відповіді.

До покращення комунікативної функції теж причетні копірайтери, яким доручають написання коментарів та відгуків.

4. Рекламна функція

Веб-сайт/додаток сам є рекламним інструментом, що дозволяє за короткий термін залучити велику кількість відвідувачів. Для цього потрібна грамотна оптимізація та просування сайтів у пошукових системах. Крім цього, розкрутка в соціальних мережах та на інших майданчиках мережі.

Своєрідною рекламою є все, що належить до діяльності компанії: продукти, послуги, сервіс, представлення інформації у такому вигляді, щоб вона привернула увагу цільової аудиторії та інших потенційних споживачів.

Залучають інші рекламні інструменти, замовляють копірайтерам:

- комерційних пропозицій;
- листів для розсилки;
- продають тексти;
- блоків для лендінгів;
- рекламних текстів;
- пропозицій, які розміщуються на дошках оголошень.

Сайт дозволяє публікувати необмежену кількість інформації, що рекламує успішніше, ніж матеріали невеликих рекламних блоків у буклетах, газетах, журналах, прайсах, листівках. Знижуються витрати на іншу рекламну продукцію.

2.2 Опис використаної архітектури та шаблонів проектування

Найпопулярнішою архітектурою програмного забезпечення, безумовно, є Model-View-Controller, або MVC.

MVC ділить будь-яку велику програму на три частини:

- модель;
- вид;
- контролер.

Кожен із цих компонентів створений для роботи з певним аспектом програми та має різні цілі.

Модель містить всю пов’язану з даними логіку, з якою працює користувач, як-от схеми та інтерфейси проекту, бази даних та їх поля.

Наприклад, об’єкт клієнта буде отримувати інформацію про клієнта з бази даних, маніпулювати або оновлювати їхній запис у базі даних або використовувати його для відтворення даних.

Подання містить інтерфейс користувача та презентацію програми.

Наприклад, перегляд клієнта включатиме всі компоненти інтерфейсу користувача, такі як текстові поля, спадні меню та інші речі, з якими взаємодіє користувач.

І, нарешті, контролер містить всю пов’язану з бізнесом логіку та обробляє вхідні запити. Це інтерфейс між моделлю та представленням.

Наприклад, контролер клієнта оброблятиме всі взаємодії та вводи з представлення клієнта та оновлюватиме базу даних за допомогою моделі клієнта. Той самий контролер буде використовуватися для перегляду даних клієнтів.

Спочатку браузер надсилає запит контролеру. Потім контролер взаємодіє з моделлю для надсилання та отримання даних.

Потім контролер взаємодіє з представленням для відтворення даних. Представлення турбується лише про те, як подати інформацію, а не про остаточну презентацію. Це буде динамічний HTML-файл, який відтворює дані на основі того, що йому надсилає контролер.

Нарешті, представлення надішле свою остаточну презентацію контролеру, а контролер надішле ці остаточні дані на вихід користувача.

Важливо те, що Вид і Модель ніколи не взаємодіють один з одним. Єдина взаємодія, яка відбувається між ними, - через контролер.

Це означає, що логіка програми та інтерфейс ніколи не взаємодіють один з одним, що полегшує написання складних програм.

Патерн MVC є основою проекту аграрної навчальної платформи який пов’язує усі частини його струкатури між собою.

Для розумінння приведемо приклад взявши за основу модель сутності “Item”.

Код:

```
<?php
declare(strict_types=1);

namespace App\Entity;

use App\Traits\TopItem;
use ReflectionClass;
use Symfony\Component\DependencyInjection\ParameterBag\ParameterBagInterface;
use Doctrine\Common\Collections\{
```

```
    Collection,
    ArrayCollection,
};
use Doctrine\ORM\Mapping as ORM;
use Knp\DoctrineBehaviors\Contract\Entity\TimestampableInterface;
use Knp\DoctrineBehaviors\Model\Timestampable\TimestampableTrait;
use App\Repository\ItemRepository;
use App\Service\FileManager\FileManagerInterface;
use Symfony\Component\Validator\Constraints as Assert;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;

/
 * @ORM\Entity(repositoryClass=ItemRepository::class)
 * @ORM\InheritanceType("JOINED")
 * @ORM\DiscriminatorColumn(name="discr", type="string")
 * @ORM\DiscriminatorMap({"article" = "Article", "course" = "Course", "webinar" = "Webinar", "other" = "Other", "occurrence" = "Occurrence" ,"news" = "News"})
 * @UniqueEntity(fields={"slug"})
 */
abstract class Item implements TimestampableInterface
{
    use TimestampableTrait;
    use TopItem;

    public const ARTICLE = 'article';
    public const COURSE = 'course';
    public const WEBINAR = 'webinar';
    public const OTHER = 'other';
    public const OCCURRENCE = 'occurrence';
    public const NEWS = 'news';

    /
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /
     * @ORM\Column(type="boolean",options={"default" : false})
     */
    private $isActive = false;

    /
     * @ORM\Column(type="boolean")
     */
    private $registrationRequired = false;

    /
     * @ORM\ManyToOne(targetEntity=FeedbackForm::class, inversedBy="items")
     */
    private $feedbackForm;

    /
     * @ORM\OneToMany(targetEntity=Comment::class, mappedBy="item")
     */
    private $comments;

    /
     * @ORM\Column(type="boolean", nullable=true)
     */
    private $commentsAllowed = false;
```

```
    /
     * @ORM\Column(type="integer", nullable=true)
     */
    private $viewsAmount = 0;

    /
     * @ORM\ManyToMany(targetEntity=Tags::class, inversedBy="items")
     */
    private $tags;

    /
     * @ORM\ManyToMany(targetEntity=Partner::class, inversedBy="items")
     */
    private $partners;

    /
     * @ORM\ManyToMany(targetEntity=Expert::class, inversedBy="items")
     */
    private $experts;

    /
     * @ORM\ManyToOne(targetEntity=Category::class, inversedBy="items")
     */
    private $category;

    /
     * @ORM\ManyToMany(targetEntity=News::class, mappedBy="similar")
     */
    private $similarNewsRelation;

    /
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $slug;

    /
     * @ORM\Column(type="float", nullable=true)
     * @Assert\Range(min = 0, max = 5)
     */
    private $rating;

    /
     * @ORM\Column(type="float", nullable=true)
     */
    private $newRating;

    /
     * @ORM\Column(type="integer", nullable=true)
     * @Assert\PositiveOrZero()
     */
    private $oldUserCount;

    /
     * @ORM\Column(type="datetime", nullable=true)
     */
    private $startDate;

/
     * @ORM\ManyToMany(targetEntity=Crop::class, inversedBy="items")
     */
    private $crops;

    public function __construct()
```

```
{
    $this->isActive = false;
    $this->registrationRequired = false;
    $this->comments = new ArrayCollection();
    $this->tags = new ArrayCollection();
    $this->partners = new ArrayCollection();
    $this->experts = new ArrayCollection();
    $this->crops = new ArrayCollection();
    $this->similarNewsRelation = new ArrayCollection();
}

public function getId(): ?int
{
    return $this->id;
}

public function getIsActive(): bool
{
    return $this->isActive ?? false;
}

public function setIsActive(bool $isActive): self
{
    $this->isActive = $isActive;

    return $this;
}

public function getRegistrationRequired(): bool
{
    return $this->registrationRequired ?? false;
}

public function setRegistrationRequired(bool $registrationRequired): self
{
    $this->registrationRequired = $registrationRequired;

    return $this;
}

public function getFeedbackForm(): ?FeedbackForm
{
    return $this->feedbackForm;
}

public function setFeedbackForm(?FeedbackForm $feedbackForm): self
{
    $this->feedbackForm = $feedbackForm;

    return $this;
}

public function getViewsAmount(): int
{
    return $this->viewsAmount ?? 0;
}

public function setViewsAmount(?int $viewsAmount): self
{
    $this->viewsAmount = $viewsAmount;

    return $this;
}
```

```
public function increaseViewsAmount(): self
{
    $this->setViewsAmount($this->getViewsAmount() + 1);

    return $this;
}

public function increaseOldUserCount(): self
{
    $this->setOldUserCount($this->getOldUserCount() + 1);

    return $this;
}

/
 * @return Collection|self[]
 */
public function getSimilarNewsRelation(): Collection
{
    return $this->similarNewsRelation;
}

public function addSimilarNewsRelation(?Item $similar): self
{
    if (!$this->similarNewsRelation->contains($similar)) {
        $this->similarNewsRelation[] = $similar;
    }

    return $this;
}

public function rNGfm99RfAqifzEoqjYY9NPszcDnpChRWu(self $similar): self
{
    $this->similarNewsRelation->removeElement($similar);

    return $this;
}

/
 * @return Collection|Comment[]
 */
public function getComments(): Collection
{
    return $this->comments;
}

public function addComment(Comment $comment): self
{
    if (!$this->comments->contains($comment)) {
        $this->comments[] = $comment;
        $comment->setItem($this);
    }

    return $this;
}

public function removeComment(Comment $comment): self
{
    if ($this->comments->removeElement($comment)) {
        // set the owning side to null (unless already changed)
        if ($comment->getItem() === $this) {
            $comment->setItem(null);
```

```
            }
        }

        return $this;
    }

    public function getCommentsAllowed(): bool
    {
        return $this->commentsAllowed ?? false;
    }

    public function setCommentsAllowed(bool $commentsAllowed): self
    {
        $this->commentsAllowed = $commentsAllowed;

        return $this;
    }

    /
     * @return Collection|Tags[]
     */
    public function getTags(): Collection
    {
        return $this->tags;
    }

    public function addTag(Tags $tag): self
    {
        if (!$this->tags->contains($tag)) {
            $this->tags[] = $tag;
        }

        return $this;
    }

    public function removeTag(Tags $tag): self
    {
        $this->tags->removeElement($tag);

        return $this;
    }

/
     * @return Collection|Partner[]
     */
    public function getPartners(): Collection
    {
        return $this->partners;
    }

    public function addPartner(Partner $partner): self
    {
        if (!$this->partners->contains($partner)) {
            $this->partners[] = $partner;
        }

        return $this;
    }

    public function removePartner(Partner $partner): self
    {
        $this->partners->removeElement($partner);
```

```
    return $this;
}

/
 * @return Collection|Expert[]
 */
public function getExperts(): Collection
{
    return $this->experts;
}

public function getExpertsCount(): int
{
    return $this->experts->count();
}

public function addExpert(Expert $expert): self
{
    if (!$this->experts->contains($expert)) {
        $this->experts[] = $expert;
    }

    return $this;
}

public function removeExpert(Expert $expert): self
{
    $this->experts->removeElement($expert);

    return $this;
}

public function getCategory(): ?Category
{
    return $this->category;
}

public function setCategory(?Category $category): self
{
    $this->category = $category;

    return $this;
}

/
 * @return mixed
 */
public function getSlug()
{
    return $this->slug;
}

/
 * @param mixed $slug
 */
public function setSlug($slug): void
{
    $this->slug = $slug;
}

/
 * @return mixed
 */
```

```
public function getRating()
{
    return number_format((float)$this->rating, 3) ?? 0;
}

/
 * @param mixed $rating
 */
public function setRating($rating): void
{
    $this->rating = $rating;
}

public function getOldUserCount(): int
{
    return (int)$this->oldUserCount ?? 0;
}

/
 * @param mixed $oldUserCount
 */
public function setOldUserCount($oldUserCount)
{
    $this->oldUserCount = $oldUserCount;
}

/
 * @return mixed
 */
public function getNewRating()
{
    return $this->newRating;
}

/
 * @param mixed $newRating
 */
public function setNewRating($newRating): void
{
    $this->newRating = $newRating;
}

/
 * @return mixed
 */
public function getStartDate()
{
    return $this->startDate;
}

/
 * @param mixed $startDate
 */
public function setStartDate($startDate): self
{
    $this->startDate = $startDate;

    return $this;
}

public function getClassName()
{
    return (new ReflectionClass($this))->getShortName();
```

```
    }

    /
     * @return Collection|Crop[]
     */
    public function getCrops(): Collection
    {
        return $this->crops;
    }

    public function addCrop(Crop $crop): self
    {
        if (!$this->crops->contains($crop)) {
            $this->crops[] = $crop;
        }

        return $this;
    }

    public function removeCrop(Crop $crop): self
    {
        $this->crops->removeElement($crop);

        return $this;
    }

/
     * Get item copy.
     *
     * @param ParameterBagInterface $parameters Parameters container.
     * @param FileManagerInterface $fileManager Files manager, used for files copy
     *
     * @return self Entity copy.
     */
    public function getCopy(
        ParameterBagInterface $parameters,
        FileManagerInterface $fileManager
    ): self {
        $result = new static();
         $result->setIsActive($this->getIsActive());
        $result->setSlug("{$this->getSlug()}-copy");
        $result->setRegistrationRequired($this->getRegistrationRequired());
        $result->setFeedbackForm($this->getFeedbackForm());
        $result->setCommentsAllowed($this->getCommentsAllowed());
        $result->setCategory($this->getCategory());
        $result->setStartDate($this->getStartDate());

        foreach ($this->getTags() as $tag) {
            $result->addTag($tag);
        }
        foreach ($this->getPartners() as $partner) {
            $result->addPartner($partner);
        }
        foreach ($this->getExperts() as $expert) {
            $result->addExpert($expert);
        }
        foreach ($this->getCrops() as $crop) {
            $result->addCrop($crop);
        }

        return $result;
    }
```

```
    /
     * Get item type.
     *
     * @return string Item type.
     */
    abstract public function getTypeItem(): string;

    /**
     * @param $filterNameList
     *
     * @return array
     */
    public static function getItemTypeByFilterName($filterNameList): array
    {
        $result = [];
        foreach ($filterNameList as $filterName) {
            switch ($filterName) {
                case 'course':
                    $result[] = Course::class;
                    break;
                case 'webinar':
                    $result[] = Webinar::class;
                    break;
                case 'article':
                    $result[] = Article::class;
                    break;
                case 'other':
                    $result[] = Other::class;
                    break;
                case 'occurrence':
                    $result[] = Occurrence::class;
                    break;
                case 'news':
                    $result[] = News::class;
                    break;
            }
        }

        return $result;
    }
}
```

У цьому прикладі можна побачити наступні частини:

- оголошення полів та констант, які супроводжеються коментарями для обробки інформації і роботою з базою даних завдяки доктрині;
- геттери і сеттери цих полів і константнт, які дозволяють звертатися до них и тим самим розробник може записати інформаціюу чи отримати її з бази даних;
- методи для обробки полів і констант.

Деякі прості методи з нескладною логікою до яких потрібно швидко та часто звертатися також можуть бути записапні у моделі.

За допомогою цих складових моделі нею можна без проблем користуватися та у контроллері та виодити чи записувати дані з бази в інтерфейс користувача (view).

2.3 Опис використаних технологій та мов програмування

2.3.1 Мова програування PHP

Абревіатура PHP (Personal Home Page) спочатку означала персональну домашню сторінку. Але тепер це рекурсивний акронім для Hypertext Preprocessor. (Це рекурсивне в тому сенсі, що саме перше слово є абревіатурою, тому повне значення не слідує за абревіатурою).

Перша версія PHP була запущена 26 років тому. Зараз це версія 8, випущена в листопаді 2020 року, але версія 7 залишається найбільш широко використовуваною.

PHP працює на движку Zend, який є найпопулярнішою реалізацією. Існують також деякі інші реалізації, як-от папуга, HPVM (Hip Hop Virtual Machine) і Hip Hop, створені Facebook.

PHP в основному використовується для створення веб-серверів. Він працює у браузері, а також може працювати в командному рядку. Отже, якщо вам не хочеться показувати свій код у браузері, ви можете показати його в терміналі.

2.3.2 Symfony фреймворк для веб-розробки

Symfony — це фреймворк, який є платформою програмування для веб-розробників для створення додатків. Завдяки йому ми можемо визначити структуру нової програми та загальний механізм її функціонування. Він також надає набір готових компонентів і бібліотек, які дозволяють виконувати конкретні завдання. Symfony — одна з найпопулярніших фреймворків, заснована на мові програмування PHP і доступна через відкритий вихідний код. Що робить Symfony дійсно популярним серед розробників, так це те, що він

значно скорочує час, необхідний для створення програмного забезпечення. Більше того, Symfony пропонує численні переваги з точки зору бізнесу.

2.3.3 Основні переваги фреймворку Symfony

У сучасному світі швидкість дій має вирішальне значення в більшості аспектів функціонування людини. Особливо це стосується техніки. Схоже і в роботі програмістів – оптимізація швидкості роботи програми є одним із ключових кроків у створенні ефективного програмного забезпечення. З Symfony не доведеться турбуватися про це. Ця структура приділяє велику увагу ефективності та швидкості роботи. На даний момент це одна з найшвидших фреймворків на основі PHP.

Незалежно від складності ваших потреб, Symfony, безумовно, адаптується до них. Це забезпечить, серед іншого, утиліту Dispatcher подій, яка дозволить вам розширити функції вже написаного коду.

Крім того, Symfony дозволяє створювати програмне забезпечення трьома способами:

Повний стек – це дозволить вам створити велику програму, багату різними функціями.

Цеглинка за цеглиною – дозволяє створити програмну функцію за функцією залежно від того, що вам зараз потрібно.

Мікрофреймворк – ви можете створювати конкретні функції у вибраних проектах без необхідності програмувати все з нуля та без встановлення всього фреймворка. З-поміж наявних ви виберете лише ті фрагменти, які вам дійсно потрібні.

Все це робить фреймворк Symfony ідеальним для створення різноманітних типів додатків із широким спектром додатків та пристосованих до певної діяльності.

Варто підкреслити, що надзвичайно корисною особливістю Symfony є те, що кожен елемент є плагіном і кожен з них додає окрему функціональність до

всього фреймворку. Завдяки цьому його можна використовувати в іншому проекті і навіть бути доступним широкій спільноті розробників.

Більше того, сама система дозволяє вносити великі зміни в Symfony, навіть модифікуючи його ядро. Таким чином, функціонування всього фреймворка можна розширити за потреби, без необхідності переналаштовувати його з нуля.

Величезною перевагою Symfony є те, що проект підтримує SensioLab разом із активною спільнотою розробників, які мають великий досвід. Завдяки цьому каркас віщує добре майбутнє і точно не буде застоюватися. Регулярні тести та часті оновлення, безсумнівно, гарантують, що він не втратить свою сучасність протягом тривалого часу.

Переваги Symfony, які ми згадували вище, такі як швидкість і гнучкість, роблять його однією з найкращих фреймворків PHP. Немає жодних проблем з адаптацією його до найбільш перевірених практик і застосовних стандартів в індустрії програмування. Більше того, до нього впроваджуються все більше інноваційних рішень. Завдяки цьому він ефективно допомагає програмістам при створенні сучасних додатків, а також підвищує їх креативність в роботі.

Функції, про які ми згадували вище, такі як швидкість і гнучкість Symfony, роблять його одним із найкращих фреймворків PHP. Немає жодних проблем з адаптацією його до найбільш перевірених практик і застосовних стандартів в індустрії програмування. Більше того, до нього впроваджуються все більше інноваційних рішень. Завдяки цьому він ефективно допомагає програмістам при створенні сучасних додатків, а також підвищує їх креативність в роботі.

Також великою перевагою фреймворку Symfony є його функція керування кешуванням. Усе тому, що MySQL виконує кожен запит, прочитаний Symfony. Важливо, що всі інструкції зберігаються в MySQL і можуть бути успішно використані в майбутньому. Заслуговує на увагу також параметр під назвою кешування фрагмента, який, зберігаючи фрагменти сторінки, зменшує кількість запитів до бази даних, якщо це необхідно. Це дуже корисне рішення, напр. при створенні кошика для покупок, статус входу користувача або можливість введення коментарів у блозі.

2.3.4 Мова програмування JavaScript

JavaScript — це мова сценаріїв або програмування, яка дозволяє вам реалізувати складні функції на веб-сторінках — щоразу, коли веб-сторінка робить щось більше ніж відображає статичну інформацію, яку ви можете подивитися — відображаючи своєчасні оновлення вмісту, інтерактивні карти, анімовані 2D/ 3D-графіка, прокручування відео-автоматів тощо — ви можете посперечатися, що JavaScript, ймовірно, задіяний. Це третій шар шарів стандартних веб-технологій, дві з яких (HTML і CSS).

JavaScript — це мова сценаріїв, яка дозволяє створювати динамічно оновлюваний вміст, керувати мультимедіа, анімувати зображення та майже все інше.

Основна мова JavaScript на стороні клієнта складається з деяких загальних функцій програмування, які дозволяють виконувати такі дії, як:

Зберігайте корисні значення всередині змінних. Але ще більш захоплюючим є функціональність, побудована на основі мови JavaScript на стороні клієнта. Так звані інтерфейси прикладного програмування (API) надають вам додаткові суперздатності для використання у вашому коді JavaScript.

API — це готові набори будівельних блоків коду, які дозволяють розробнику реалізовувати програми, які інакше було б важко або неможливо реалізувати. Вони роблять те саме для програмування, що й готові меблі для будівництва будинку — набагато простіше взяти готові панелі й прикрутити їх разом, щоб зробити книжкову полицю, ніж самостійно розробити дизайн, піти й знайти виправте деревину, виріжте всі панелі до потрібного розміру та форми, знайдіть шурупи правильного розміру, а потім з’єднайте їх, щоб зробити книжкову полицю.

API-інтерфейси браузера вбудовані у ваш веб-переглядач і можуть надавати дані з навколишнього комп’ютерного середовища або виконувати корисні складні речі.

Наприклад:

- API DOM (Document Object Model) дозволяє маніпулювати HTML і CSS, створювати, видаляти та змінювати HTML, динамічно застосовувати нові стилі до вашої сторінки тощо. (як ми бачили вище в нашій простій демонстрації), наприклад, це DOM в дії;

- API геолокації отримує географічну інформацію. Ось як Google Maps може знайти ваше місцезнаходження та нанести його на карту;

- API Canvas і WebGL дозволяють створювати анімовану 2D і 3D графіку. Люди роблять дивовижні речі за допомогою цих веб-технологій — див. Експерименти Chrome і webglsamples;

- API аудіо та відео, такі як HTMLMediaElement і WebRTC, дозволяють робити дійсно цікаві речі з мультимедіа, наприклад, відтворювати аудіо та відео прямо на веб-сторінці, або захоплювати відео зі своєї веб-камери та відображати його на чужому комп'ютері (спробуйте нашу просту демонстраційну демонстрацію Snapshot щоб зрозуміти ідею).

2.3.5 Бібліотека JQuery

jQuery — це швидка, невелика і багатофункціональна бібліотека JavaScript. Це значно спрощує такі речі, як обхід і маніпуляції з документами HTML, обробку подій, анімацію та Ajax, завдяки простому у використанні API, який працює в багатьох браузерах.

jQuery полегшує життя веб-розробнику. Він надає безліч вбудованих функцій, за допомогою яких можна легко і швидко виконувати різноманітні завдання.

Вибір DOM: jQuery надає селектори для отримання елемента DOM на основі різних критеріїв, таких як назва тегу, ідентифікатор, назва класу css, назва атрибута, значення, n-й дочірній елемент в ієрархії тощо.

Маніпулювання DOM: Ви можете маніпулювати елементами DOM за допомогою різних вбудованих функцій jQuery. Наприклад, додавання або видалення елементів, зміна вмісту html, класу css тощо.

Спеціальні ефекти: Ви можете застосувати спеціальні ефекти до елементів DOM, як-от показати або приховати елементи, зникнення або зникнення видимості, ефект ковзання, анімацію тощо.

Події: бібліотека jQuery містить функції, еквівалентні подіям DOM, як-от click, dblclick, mouseenter, mouseleave, blur, keyup, keydown тощо. Ці функції автоматично вирішують проблеми між браузерами.

Ajax: jQuery також включає прості у використанні функції AJAX для завантаження даних із серверів без перезавантаження всієї сторінки.

Підтримка між браузерами: бібліотека jQuery автоматично вирішує проблеми між браузерами, тому користувачеві не доведеться турбуватися про це. jQuery підтримує IE 6.0+, FF 2.0+, Safari 3.0+, Chrome і Opera 9.0+.

2.3.6 Переваги jQuery

Легко навчитися: jQuery легко вивчити, оскільки він підтримує кодування в одному стилі JavaScript.

Пишіть менше, робіть більше: jQuery надає багатий набір функцій, які підвищують продуктивність розробників, пишучи менше і читабельний код.

Чудова документація API: jQuery надає чудову онлайн-документацію API.

Підтримка між браузерами: jQuery забезпечує відмінну міжбраузерну підтримку без написання додаткового коду.

Ненав’язливий: jQuery ненав’язливий, що дозволяє розділяти проблеми, розділяючи код html і jQuery.

2.3.7 Мова розмітки HTML

HTML, або Hypertext Markup Language, — це мова розмітки для Інтернету, яка визначає структуру веб-сторінок.

Це один з основних будівельних блоків кожного веб-сайту, тому важливо навчитися, якщо ви хочете зробити кар’єру у веб-розробці.

Склад абревіатури HTML:

гіпертекст: текст (часто з вбудованими, наприклад зображеннями), організований для з'єднання пов'язаних елементів;

розмітка: посібник зі стилю для верстки будь-чого, що буде надруковано у паперовому чи друкованому форматі;

мова: мова, яку розуміє комп'ютерна система та використовує для інтерпретації команд.

HTML визначає структуру веб-сторінок. Однієї цієї структури недостатньо, щоб веб-сторінка виглядала добре та інтерактивно. Таким чином, ви будете використовувати допоміжні технології, такі як CSS і JavaScript, щоб зробити свій HTML красивим і додати інтерактивності відповідно.

Оскільки HTML визначає розмітку для певної веб-сторінки, потрібно, щоб текст, зображення чи інші вбудовані елементи відображалися певним чином.

Наприклад, можна зробіии так, щоб деякий текст був великим, інший текст був маленьким, а якийсь був жирним, курсивом або у формі маркера.

HTML має "теги", які дозволяють це зробити. Отже, існують теги для створення заголовків, абзаців, слів, виділених жирним шрифтом, слів, виділених курсивом тощо.

2.3.8 CSS стилі

CSS (каскадні таблиці стилів) дозволяє створювати веб-сторінки, які чудово виглядають.

Документ зазвичай є текстовим файлом, структурованим за допомогою мови розмітки — HTML є найпоширенішою мовою розмітки, але ви також можете зустріти інші мови розмітки, такі як SVG або XML.

Презентувати документ користувачеві означає перетворити його у форму, яку може використовувати ваша аудиторія. Браузери, такі як Firefox, Chrome або Edge, призначені для візуального представлення документів, наприклад, на екрані комп’ютера, проектора чи принтера.

CSS можна використовувати для дуже простих стилів тексту документа — наприклад, для зміни кольору та розміру заголовків і посилань. Його можна

використовувати для створення макета — наприклад, для перетворення окремого стовпця тексту на макет із основною областю вмісту та бічною панеллю для пов’язаної інформації. Його навіть можна використовувати для таких ефектів, як анімація.

2.3.9 Docker контейнери

Слово «Docker» відноситься до кількох речей, включаючи проект спільноти з відкритим кодом; інструменти з проекту з відкритим кодом; Docker Inc., компанія, яка в першу чергу підтримує цей проект; та інструменти, які компанія офіційно підтримує. Той факт, що технології та компанія мають однакову назву, може ввести в оману.

IT-програмне забезпечення «Docker» — це технологія контейнеризації, яка дозволяє створювати та використовувати контейнери Linux.

Спільнота Docker з відкритим кодом працює над покращенням цих технологій, щоб принести користь усім користувачам.

Компанія Docker Inc. спирається на роботу спільноти Docker, робить її більш безпечною та ділиться цими досягненнями з більшою спільнотою. Потім він підтримує вдосконалені та посилені технології для корпоративних клієнтів.

За допомогою Docker можна розглядати контейнери як надзвичайно легкі модульні віртуальні машини. І ви отримуєте гнучкість із цими контейнерами – ви можете створювати, розгортати, копіювати та переміщувати їх із середовища в середовище, що допомагає оптимізувати ваші програми для хмари.

Технологія Docker використовує ядро Linux і функції ядра, такі як Cgroups і простори імен, для відокремлення процесів, щоб вони могли працювати незалежно. Ця незалежність — це мета контейнерів — можливість запускати кілька процесів і програм окремо один від одного, щоб краще використовувати вашу інфраструктуру, зберігаючи безпеку, яку ви мали б із окремими системами.

Інструменти контейнера, включаючи Docker, забезпечують модель розгортання на основі образів. Це полегшує спільний доступ до програми або

набору служб з усіма їх залежностями в кількох середовищах. Docker також автоматизує розгортання програми (або комбінованих наборів процесів, з яких складається програма) всередині цього контейнерного середовища.

Ці інструменти, створені на основі контейнерів Linux, що робить Docker зручним і унікальним, надають користувачам безпрецедентний доступ до програм, можливість швидкого розгортання та контроль над версіями та розповсюдженням версій.

Підхід Docker до контейнеризації зосереджується на можливості видаляти частину програми для оновлення або відновлення, не видаляючи всю програму. На додаток до цього підходу, заснованого на мікросервісах, ви можете спільно використовувати процеси між кількома додатками приблизно так само, як це робить сервісно-орієнтована архітектура (SOA).

Кожен файл зображення Docker складається з серії шарів, які об’єднані в одне зображення. Шар створюється, коли зображення змінюється. Щоразу, коли користувач вказує команду, наприклад запустити або копіювати, створюється новий шар.

Docker повторно використовує ці шари для створення нових контейнерів, що прискорює процес побудови. Проміжні зміни розподіляються між зображеннями, що ще більше покращує швидкість, розмір та ефективність. Також для багатошаровості притаманний контроль версій: щоразу, коли відбувається нова зміна, у вас по суті є вбудований журнал змін, що надає вам повний контроль над зображеннями контейнера.

Мабуть, найкраща частина багатошаровості - це здатність відкочуватися. Кожне зображення має шари. Не подобається поточна ітерація зображення? Поверніть його до попередньої версії. Це підтримує гнучкий підхід до розробки та допомагає зробити безперервну інтеграцію та розгортання (CI/CD) реальністю з точки зору інструментів.

Запуск, підготовка та доступність нового обладнання займали дні, а рівень зусиль і накладних витрат був обтяжливим. Контейнери на основі Docker можуть скоротити розгортання до секунд. Створивши контейнер для кожного

процесу, ви можете швидко поділитися цими процесами з новими додатками. А оскільки операційній системі не потрібно завантажуватися, щоб додати або перемістити контейнер, час розгортання значно коротший. У поєднанні з коротким часом розгортання ви можете легко та економічно ефективно створювати та знищувати дані, створені вашими контейнерами, без турбот.

Отже, технологія Docker — це більш детальний, контрольований підхід на основі мікросервісів, який приділяє більше значення ефективності.

Docker сам по собі може керувати окремими контейнерами. Коли ви починаєте використовувати все більше контейнерів і контейнерних додатків, розбитих на сотні частин, керування та оркестровка можуть стати складними. Зрештою, вам доведеться зробити крок назад і згрупувати контейнери, щоб надавати послуги — мережу, безпеку, телеметрію тощо — для всіх ваших контейнерів. Ось тут і з’являється Kubernetes.

За допомогою Docker ви не отримуєте ту саму функціональність, подібну до UNIX, яку ви отримуєте з традиційними контейнерами Linux. Це включає в себе можливість використовувати такі процеси, як cron або syslog, у контейнері разом із додатком. Існують також обмеження на такі речі, як очищення дочірніх процесів після завершення дочірніх процесів — те, з чим зазвичай обробляються традиційні контейнери Linux. Ці проблеми можна пом’якшити, змінивши файл конфігурації та налаштувавши ці можливості з самого початку, але це може бути неочевидним на перший погляд.

Крім того, існують інші підсистеми та пристрої Linux, які не мають простору імен. До них належать пристрої SELinux, Cgroups та /dev/sd*. Це означає, що якщо зловмисник отримує контроль над цими підсистемами, хост буде скомпрометований. Щоб залишатися легким, спільне використання ядра хоста з контейнерами відкриває цю можливість уразливості безпеки. Це відрізняється від віртуальних машин, які набагато жорсткіше відокремлені від хост-системи.

Демон Docker також може бути проблемою безпеки. Щоб використовувати та запускати контейнери Docker, ви, швидше за все, будете використовувати

демон Docker, постійне середовище виконання для контейнерів. Демону Docker потрібні привілеї root, тому слід особливо уважно ставитися до того, хто отримує доступ до цього процесу і де він знаходиться.

2.4 Опис структури програми та алгоритмів її функціонування

Web-додаток начальної аграрної платформи є монолітом иак як це полегшує написання сайтів та додатків при невеликій кількості розробників і надає легкий доступ до усіх частин коду. Сам проект написаний на фреймворку Symfony та має структуру за патерном MVC.

Завдяки структурі MVC та фреймворку Symfony створення проекту стає значно швидшим та більш гнучким. Також застосовується Docker та Composer.

Composer є маршрутизатором з допомогою якого класси з різних частин MVC можуть взаємодіяти між собою. Ключове слово для їх підключення “use”. Завдяки цьому використання полів та методів інших классів стає можливим всього лише за допомого оголошення цього классу.

Адміністративна панель є також окремою частиною проекту яка реалізована за допомогою бандла “EasyAdmin”. Він надає свій інтервейс та базові логічні взаємодії одразу після підключення та має усе необхідне для створення повноцінної багатофункціональної адміністративної панелі.

2.5 Опис розробленого програмного продукту

Розробка веб-додатку “АгроВікі” заснована PHP фреймвоку Symfony який побудований за допомогою патерну MVC. У проекті застсовуються такі мов як:

- PHP;
- JavaScript;
- HTML;
- CSS.

Проект поділений на такі основні частини:

- головна сторінка;

- сторінка навчання;
- адміністративна панель.

Головна сторінка надає можливість швитко вибрати найпопулярнішу та найважливішу інформацію яка розміщенна у веб-додатку (рис. 2.1).

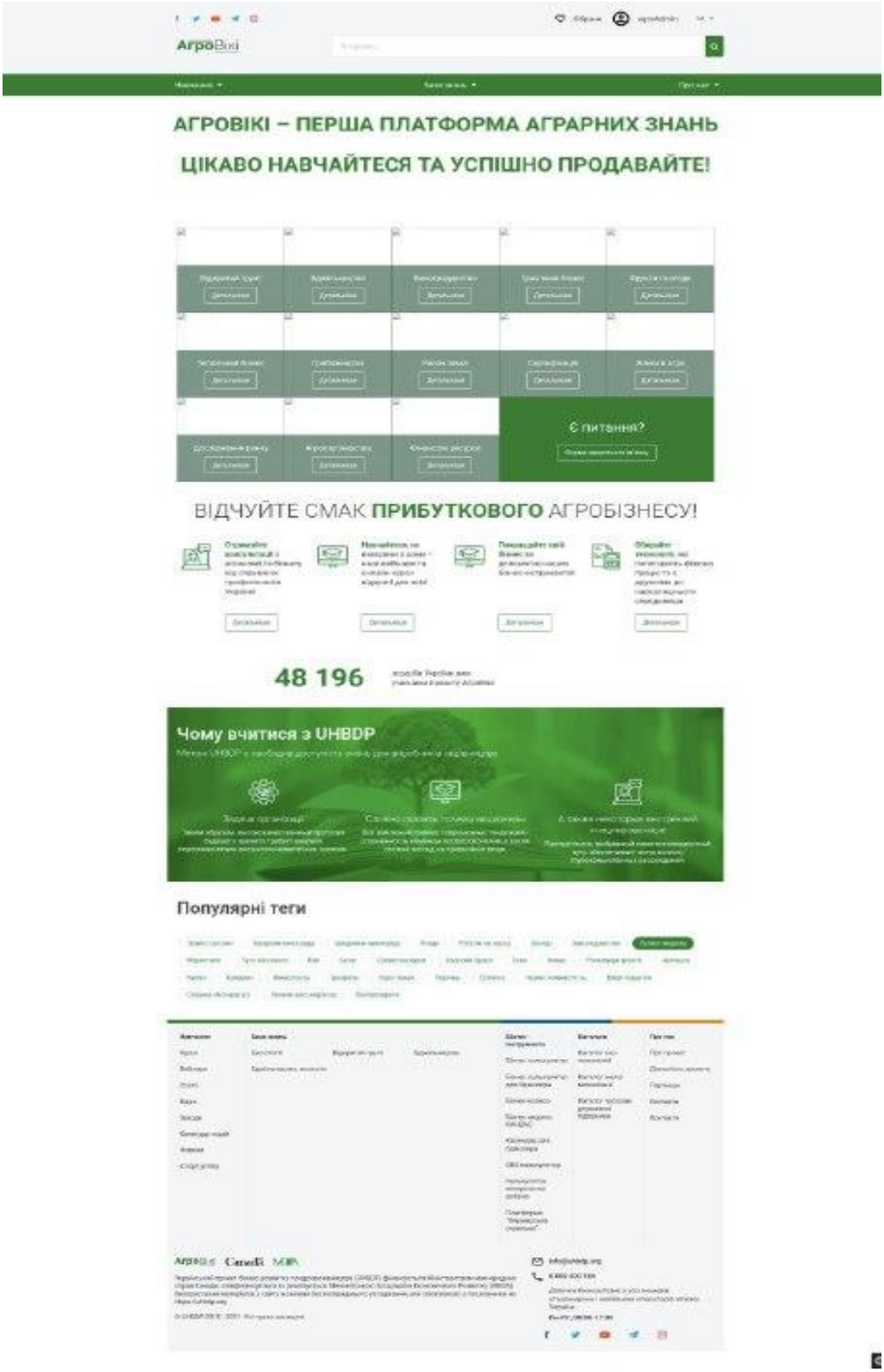

Рис. 2.1. Головна сторінка

Особистий кабінет дозволяє користувачу швидко та зручно редагувати свій особитий кабінет, підписуватись на курсі та вебінари, підтримувати зв’язок зі спонсорами та адміністраторами сайту (рис. 2.2).

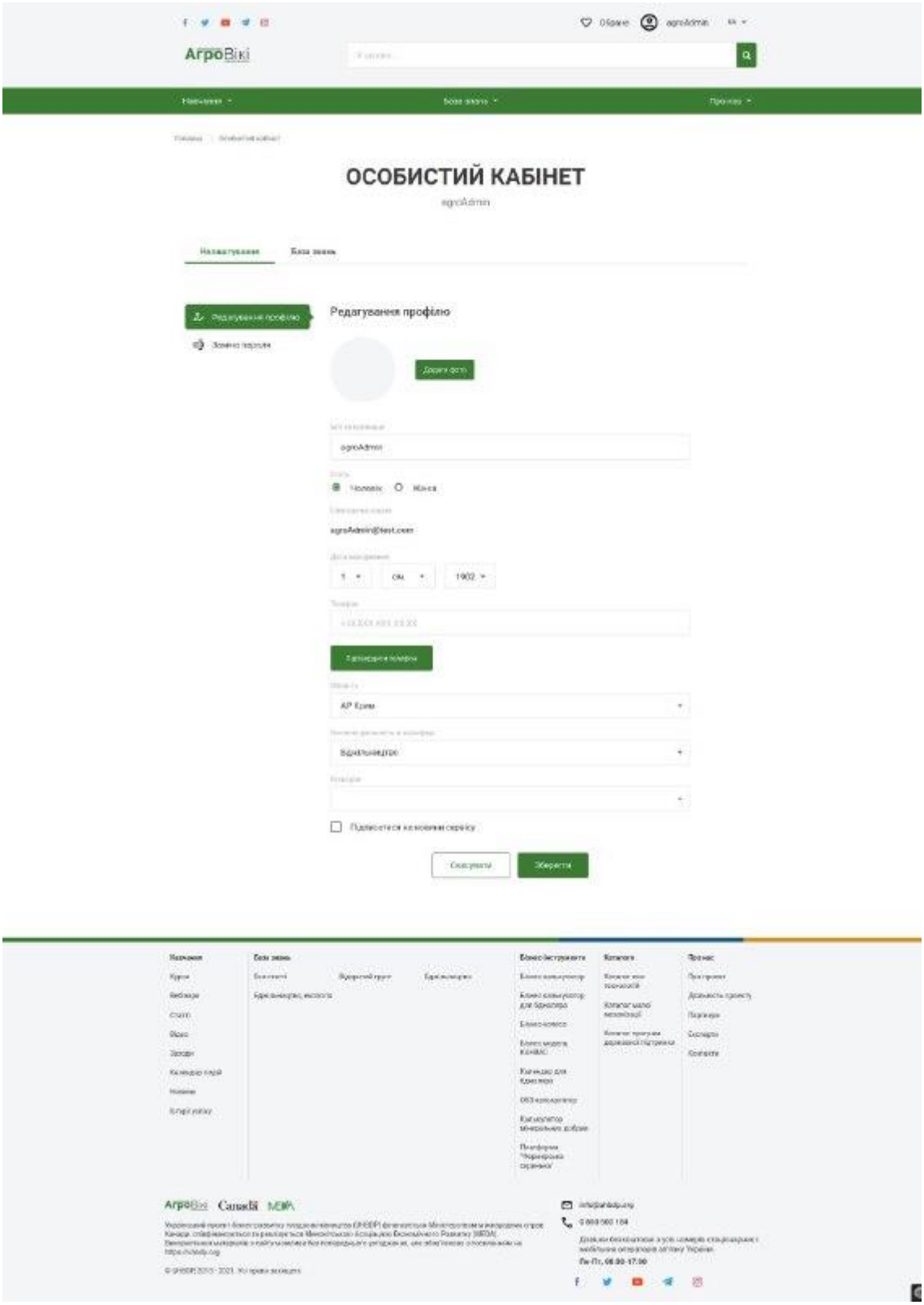

Рис. 2.2. Особистий кабінет

Розділ навчання надає доступ до усієї бази інформації яка мається у веб-додатку. Також у цьому розділі та й на усьому сайті є пошук який дозволяє шукати потрібну вам інформацію по всьому додатку (рис. 2.3).

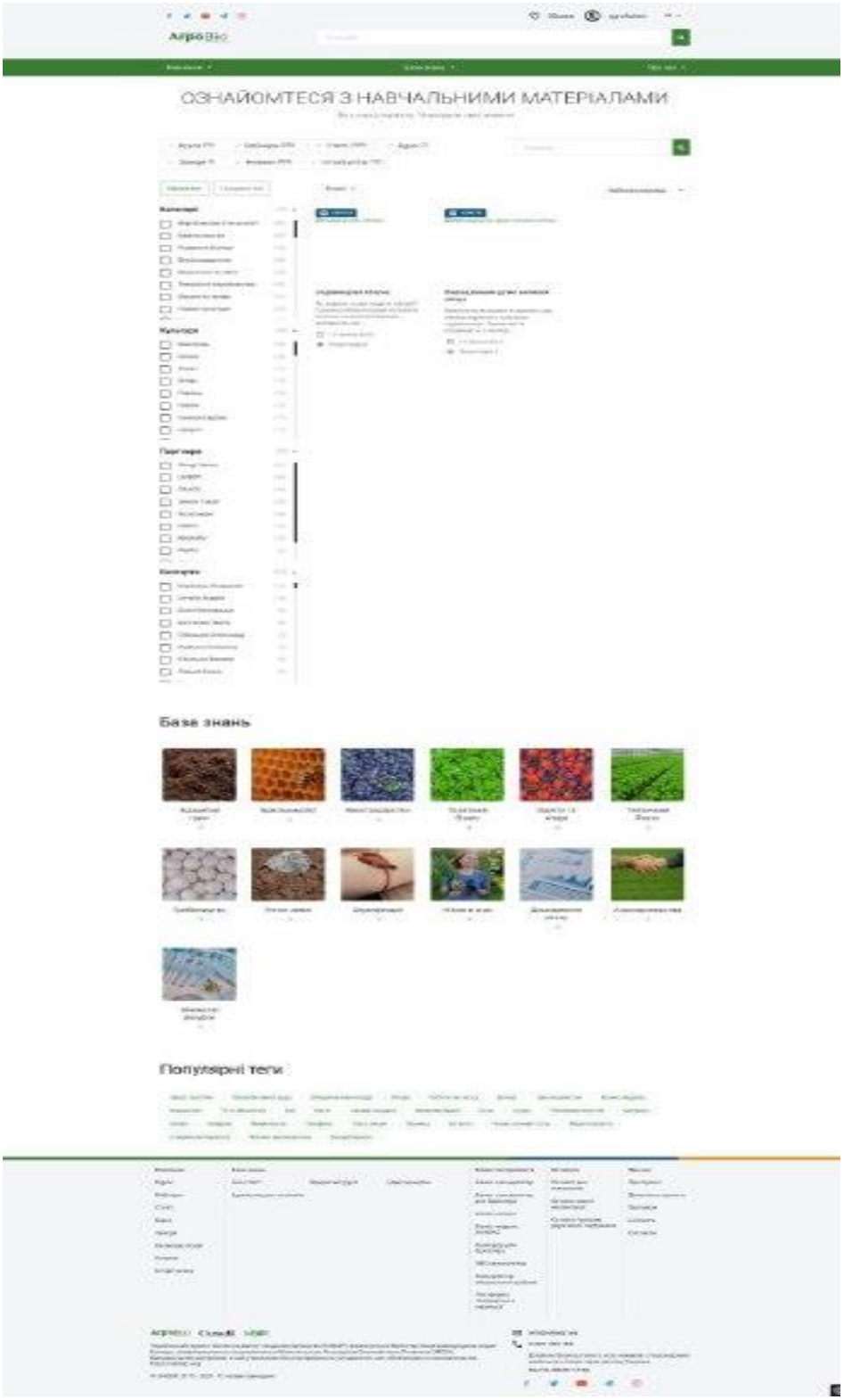

Рис 2.3. Навчальні матеріали

Адмін панель (рис. 2.4) написана за допомогою бандла EasyAdmin, вона дозволяє:

- відстежувати івенти користувачів додатку;
- створювати нові заходи та повідомлення;
- здійснювати комунікацію з користувачами.

Рис 2.4. Адміністратина панель

2.5.1 Використані технічні засоби

Програмний продукт був розроблений на ПК з операційною системою Windows. ПК мав такі параметри: Intel Core 7 8th Gen, RAM 32 ГБ.

2.5.2 Опис інтерфейсу користувача

Інтерфейс користувача має базовий каркас для усіх сторінок на сайті, а саме хедер (рис. 2.5) и футер (рис. 2.6).

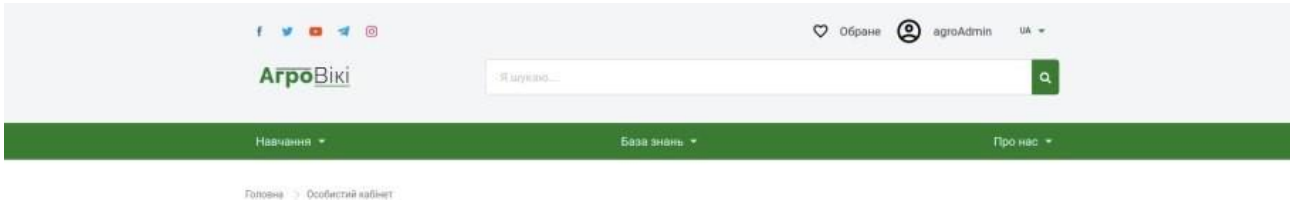

Рис 2.5. Хедер

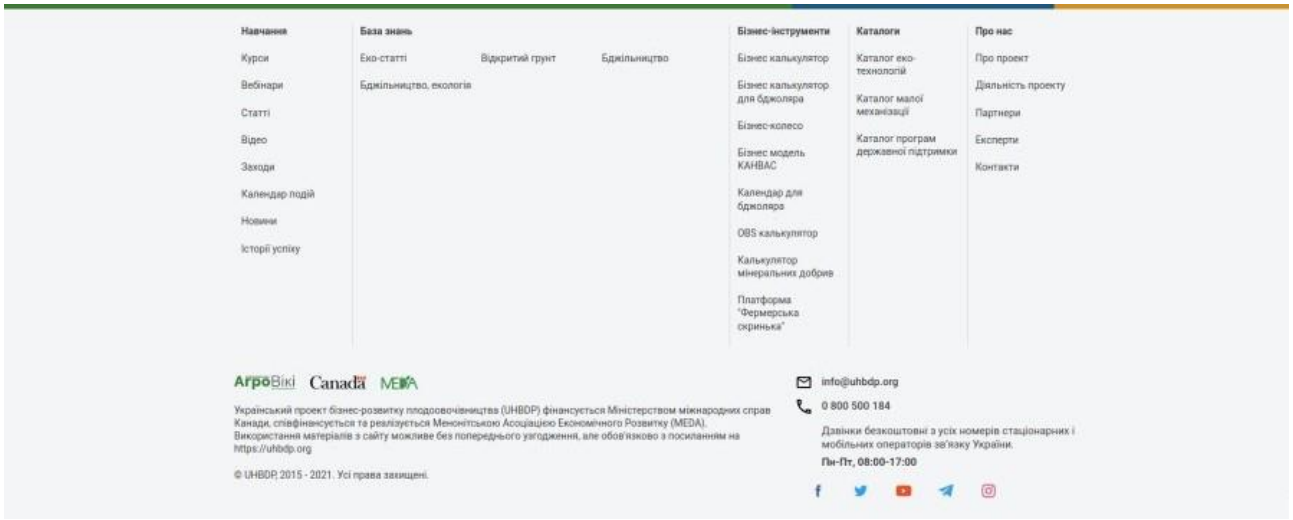

Рис. 2.6. Футер

А також основну частину сторінок, які були представленні на рис.2.1 – 2.3.

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

Основна мета економічного поділу полягає у визначенні доцільності реалізації запропонованого десктопного докладання шляхом оцінки його ефективності після впровадження в організацію.

Кінцевою метою є створення аграрног додатку.
Основне призначення цієї розробки – поліпшення роботи аграрного бізнесу.

3.1 Оцінка інвестиційних витрат

При розробці заходів у галузі інформаційних систем та технологій інвестиційні витрати розраховуються як капітальні вкладення.

Капітальні вкладення є інвестиції, створені задля відтворення основних засобів.

Капітальні вкладення можна обчислити за такою формулою:
К_р=З_про+З_пп+З_мат+З_зп+З_сн+З_пр,
де
З_об - витрати, пов'язані з придбанням обладнання або експлуатацією техніки, грн.;
З_пп - Витрати спеціальні програмні продукти, які необхідні для розробки проекту, грн.;
З_мат - витрати на господарсько-операційні потреби, грн.;
З_ЗП - загальний фонд оплати праці розробників, грн.;
З_сн - витрати на виплату страхових внесків, грн.;
З_пр - інші витрати, грн.

Для даних розрахунків потрібно виділити основні етапи проекту, докладно подати список роботи кожному з етапів, враховуючи послідовність виконання.

Внаслідок роботи над ВКР не проводилася закупівля додаткового обладнання та його експлуатація. У результаті витрати на придбання чи експлуатацію прирівнюються до нуля, тобто. З_про = 0.

Спеціальні програмні продукти, використані під час розробки, є безкоштовними і представлені у відкритому доступі, в такий спосіб, витрати на програмне забезпечення дорівнюють нулю, тобто. Зпп = 0.

Далі проводиться розрахунок матеріальних витрат за такою формулою:

З_мат = 0З_з+З0_м+З_ел+З_тр+З_(пр.мат) ,

де

З_с - витрати на сировину, грн.;

З_м - витрати на матеріали, грн.;

З_ел - Витрати на електроенергію, грн.;

З_тр - витрати на транспорт, грн.;

З_(пр.мат) - інші матеріальні витрати, грн.

У процесі розробки програми використовувався персональний комп'ютер із блоком живлення, номінальною потужністю 600 Ватт, тоді:

З_ел = (600 * k) / 1000 * С * Т,

де

k – коефіцієнт корисної роботи;

С - вартість кіловат-години, грн.;

Т - час використання ПК, год.

При вартості 1 кіловат-години 4,98 грн:

З_ел = (600 * 1,2) / 1000 * 4,98 * 300 = 1075,68 (грн.)

До інших матеріальних витрат відноситься доступ до Інтернету. Тарифний план становить 580 грнлів/місяць, отже З_(пр.мат) = 1160 грн.

Таким чином, З_с = 0 грн.; З_м = 0 грн.; З_ел = 1075,68 грн.; З_тр = 0 грн.; З_пр=1160 грн., Отже,

З_мат = 0З_з+З0_м+З_ел+З_тр+З_пр=0+0+1075,68+0+1160=2235,68 (грн.)

Наступним кроком є розрахунок витрат за оплату праці розробнику проекту. При розрахунку заробітної плати використовуватиметься погодинна форма оплати праці:

З_вр = Т * С,

де

Т – фактично відпрацьований час працівником, год;

З – годинна тарифна ставка працівника, грн./год.

Витрати на оплату праці дорівнюють сумі заробітних плат усіх працівників проекту.

Категорія працівника Трудомісткість розробки, чол./год. Годинна ставка, грн. Сума, грн. Сума з урахуванням страхових внесків, грн.

Розробник 300450 135000 175500

Таким чином, витрати на оплату праці складають З_ЗП = 135000 грн.

На фонд заробітної плати провадиться нарахування страхових внесків на обов'язкове соціальне страхування – це обов'язкові відрахування за встановленими законодавством нормами (обов'язкове пенсійне страхування, страхування у разі тимчасової непрацездатності та материнства, обов'язкове медичне страхування):

З_сн=З_зп*К_сн

На момент 2021 тариф на страхові внески становить 30% від загального фонду оплати праці, дана ставка регулюється державою.

Таким чином,

З_сн = 135000 * 0,3 = 40500 (грн.).

Інші витрати З_пр приймемо за 0.

Таким чином, капітальні витрати, необхідні для реалізації проекту, рівні:

К_р = 0 +0 +2235,68 +135500 +40500 +0 = 178235,68 (грн.)

3.2 Оцінка експлуатаційних витрат

Експлуатаційні витрати є річні поточні витрати, пов'язані з експлуатацією впровадженого проекту.

Експлуатаційні витрати розраховуються за формулою:

З_експл=З_мат+З_зп+З_с

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка веб-додптку для для навчання, взаємодії та пошуку партнерів робітникиів аграрного бізнесу

Додаток являє собою платформу, що реалізує автоматизований збір, обробку і маніпулювання даними і включає технічні засоби обробки даних.

База даних, яка забезпечує зберігання інформації і являє собою проіменовану сукупність даних, організованих за певними правилами, що включає загальні принципи опису, зберігання і маніпулювання даними. Додаток призначений для покращення діяльності працівників аграрної сфери і надає можливість виконувати наступні дії:

- додавання, видалення і редагування інформації сторінок за допомогою адміністративної панелі;
- перегляд інформації про багату кількість галузей аграрної сфери в Україні, їх партнерів та експертів;
- здійснення пошуку необхідної інформації;
- можливість входу в систему з різними рівнями доступу до даних: користувач, адміністратор;
- можливість зміни користувача у ході роботи програми;
- здійснення контролю введених даних: перевірка на відповідність типів, на введення обов'язкових полів даних, а також, на введення тільки можливих значень, прочитуваних із необхідних таблиць;
- можливість перегляду інформації з таблиць в режимі реального часу.

Актуальність поставленої задачі обумовлюється широким попитом на аграрний бізнес у нашій країні.

Розроблений додаток призначений для застосування в будь-якому куточку України та світу де є інтернет мережа.

Створений додаток дозволить оптимізувати та спростити дії по веденнюаграрного бізнесу; скоротити час напошук партнерів та єкспертів.

Структура програми являє собою веб-додаток, написаний на мові програмування високого рівня PHP, що взаємодіє з базою даних «MySQL» за допомогою технології Doctrine. База даних розроблена мовою SQL.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (300 чол-год), підраховані витрати на створення програмного забезпечення (178235,68 грн.) і гаданий період розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://www.php.net
2. https://symfony.com/
3. https://ukraine-inform.com/news-ekonomika-agro-industry/103866-stan-agrarno-promislovost-v-ukra-n-problemi-y-tendenc-rozvitku.html
4. https://nachasi.com/
5. https://uk.wikipedia.org
6. https://web-creator.ru/articles/about_frameworks
7. https://www.docker.com
8. https://docs.gitlab.com/runner/executors/docker.html
9. https://jquery.com
10. https://www.w3schools.com/jquery/default.asp
11. https://www.npmjs.com/package/jquery
12. https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics
13. https://css-tricks.com
14. https://getcomposer.org/
15. https://packagist.org
16. https://github.com/composer/composer
17. https://twig.symfony.com/
18. http://devilbox.org/
19. https://ajax.systems/
20. https://easyadmin.readthedocs.io/en/latest/
21. https://getbootstrap.com
22. https://vuejs.org
23. https://sass-lang.com/
24. https://yarnpkg.com
25. https://designpatternsphp.readthedocs.io/en/latest/README.html

ДОДАТОК А
Код програми

```
<?php

namespace App\Controller;

use App\Helper\SeoHelper;
use App\Service\SeoService;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use App\Entity\{Article, Category, Crop, Expert, Item, Partner, Tags,
TypePage};
use Doctrine\ORM\NonUniqueResultException;
use Doctrine\ORM\NoResultException;
use Knp\Component\Pager\PaginatorInterface;

/
 * Class ItemController
 * @package App\Controller
 */
class ItemController extends AbstractController
{
    private const QUERY_PARAMETER_SEARCH = 'search';
    private const QUERY_PARAMETER_FILTER_CATEGORY = 'category';
    private const QUERY_PARAMETER_FILTER_CROP = 'crop';
    private const QUERY_PARAMETER_FILTER_PARTNER = 'partner';
    private const QUERY_PARAMETER_FILTER_EXPERT = 'expert';
    private const QUERY_PARAMETER_FILTER_TYPE = 'type';
    private const QUERY_PARAMETER_SORT_BY = 'sortBy';
    private const QUERY_PARAMETER_SORT_VALUES = [
        'viewsAmount',
        'createdAt',
    ];
    private const QUERY_PARAMETER_PAGE = 'page';
    private const LIST_PAGE_SIZE = 24;

    /
     * @Route("/courses-and-webinars", name="courses_and_webinars")
     *
     * @param PaginatorInterface $paginator
     * @param Request $request
     * @param SeoService $seoService
```

```
     * @return Response|string
     * @throws NoResultException
     * @throws NonUniqueResultException
     */
    public function list(PaginatorInterface $paginator, Request $request,
SeoService $seoService): Response
    {
        $categoriesBlock = $this->getDoctrine()-
>getRepository(Category::class)->getHomePageCategories();
        $topTags = $this->getDoctrine()->getRepository(Tags::class)-
>getTopAll($request->getLocale());

        $countAll = $this->getDoctrine()->getRepository(Item::class)-
>countActiveItems();

        $pageTypeArticle = $this->getDoctrine()-
>getRepository(TypePage::class)->findOneBy(['code' =>
TypePage::TYPE_ARTICLE]);
        $pageTypeEcoArticle = $this->getDoctrine()-
>getRepository(TypePage::class)->findOneBy(['code' =>
TypePage::TYPE_ECO_ARTICLES]);
        $countArticle = $this->getDoctrine()->getRepository(Article::class)-
>getCountArticle([$pageTypeArticle->getId(), $pageTypeEcoArticle-
>getId()]);

        $pageTypeSuccessStories = $this->getDoctrine()-
>getRepository(TypePage::class)->findOneBy(['code' =>
TypePage::TYPE_SUCCESS_STORIES]);
        $countSuccessStories = $this->getDoctrine()-
>getRepository(Article::class)->getCountArticle([$pageTypeSuccessStories-
>getId()]);

        $coursesCount = $countAll['coursesCount'];
        $articlesCount = $countArticle;
        $webinarsCount = $countAll['webinarsCount'];
        $othersCount = $countAll['othersCount'];
        $occurrenceCount = $countAll['occurrenceCount'];
        $newsCount = $countAll['newsCount'];

        $all = $request->query->all();

        $search = null;
        if (!empty($all[self::QUERY_PARAMETER_SEARCH])) {
            $search =
htmlspecialchars($all[self::QUERY_PARAMETER_SEARCH]);
```

```
        }

        $categoryFilter = null;
        if (!empty($all[self::QUERY_PARAMETER_FILTER_CATEGORY])) {
            $categoryFilter = explode(',',
($all[self::QUERY_PARAMETER_FILTER_CATEGORY]));
        }

        $cropFilter = null;
        if (!empty($all[self::QUERY_PARAMETER_FILTER_CROP])) {
            $cropFilter = explode(',',
($all[self::QUERY_PARAMETER_FILTER_CROP]));
        }

        $partnerFilter = null;
        if (!empty($all[self::QUERY_PARAMETER_FILTER_PARTNER])) {
            $partnerFilter = explode(',',
($all[self::QUERY_PARAMETER_FILTER_PARTNER]));
        }

        $expertFilter = null;
        if (!empty($all[self::QUERY_PARAMETER_FILTER_EXPERT])) {
            $expertFilter = explode(',',
($all[self::QUERY_PARAMETER_FILTER_EXPERT]));
        }

$typeFilter = null;
        if (!empty($all[self::QUERY_PARAMETER_FILTER_TYPE])) {
            $typeFilter = explode(',',
($all[self::QUERY_PARAMETER_FILTER_TYPE]));
        }

        $sortBy = null;
        if (!empty($all[self::QUERY_PARAMETER_SORT_BY])) {
            $sortBy = $all[self::QUERY_PARAMETER_SORT_BY];
        }

        $page = null;
        if (!empty($all[self::QUERY_PARAMETER_PAGE])) {
            $page = $all[self::QUERY_PARAMETER_PAGE];
        }

        $typePage = $this->getDoctrine()->getRepository(TypePage::class)-
>findOneBy(['code' => 'news']);
```

```
        $categories = $this->getDoctrine()->getRepository(Category::class)-
>getAllSorted($typeFilter);
        $crops = $this->getDoctrine()->getRepository(Crop::class)-
>getAllSorted($typeFilter);
        $partners = $this->getDoctrine()->getRepository(Partner::class)-
>getAllSorted($typeFilter);
        $experts = $this->getDoctrine()->getRepository(Expert::class)-
>getAllSorted($typeFilter);

        $items = $this->getDoctrine()->getRepository(Item::class)-
>getFilteredItems(
            $categoryFilter,
            $cropFilter,
            $partnerFilter,
            $expertFilter,
            $typeFilter,
            $sortBy,
            $search,
            $request->getLocale(),
            $typePage
        );

        $items = $paginator->paginate(
            $items,
            $request->query->getInt(self::QUERY_PARAMETER_PAGE, 1),
            self::LIST_PAGE_SIZE
        );

        if ($request->isXmlHttpRequest()) {
            return new JsonResponse([
                'content' => $this->render('blocks/item-panel/list-items.html.twig',
                    [
                        'items' => $items,
                    ])->getContent(),
                'filter' => $this->render('blocks/item-panel/filter-items.html.twig',
                    [
                        'categories' => $categories,
                        'crops' => $crops,
                        'partners' => $partners,
                        'experts' => $experts,
                        'appliedQueryParams' => [
                            self::QUERY_PARAMETER_FILTER_CATEGORY =>
$categoryFilter,
                            self::QUERY_PARAMETER_FILTER_CROP =>
$cropFilter,
```

```
                    self::QUERY_PARAMETER_FILTER_PARTNER =>
$partnerFilter,
                    self::QUERY_PARAMETER_FILTER_EXPERT =>
$expertFilter,
                ],
            ])->getContent(),
        ], Response::HTTP_OK);
    }

    $seo = null;
    if (is_array($typeFilter) && count($typeFilter) === 1) {
        $seo = $seoService
            ->setPage(SeoHelper::PAGE_COURSES_AND_WEBINARS,
['type' => $typeFilter[0]])
            ->getSeo();
    }

    $metaRobots = null;
    if ($page < 2 && (
        (is_array($typeFilter) && count($typeFilter) !== 1)
        !is_null($categoryFilter)
        !is_null($partnerFilter)
        !is_null($sortBy))
    ) {
        $metaRobots = 'noindex, nofollow';
    }

return $this->render('item/full.html.twig',
        [
            'seo' => $seo,
            'metaRobots' => $metaRobots,
            'items' => $items,
            'categoriesBlock' => $categoriesBlock,
            'topTags' => $topTags,
            'categories' => $categories,
            'crops' => $crops,
            'partners' => $partners,
            'experts' => $experts,
            'coursesCount' => $coursesCount,
            'articlesCount' => $articlesCount,
            'webinarsCount' => $webinarsCount,
            'otherCount' => $othersCount,
            'occurrenceCount' => $occurrenceCount,
            'newsCount' => $newsCount,
            'countSuccessStories' => $countSuccessStories,
```

```
            'appliedQueryParams' => [
                self::QUERY_PARAMETER_SEARCH => $search,
                self::QUERY_PARAMETER_FILTER_CATEGORY =>
$categoryFilter,
                self::QUERY_PARAMETER_FILTER_CROP => $cropFilter,
                self::QUERY_PARAMETER_FILTER_PARTNER =>
$partnerFilter,
                self::QUERY_PARAMETER_FILTER_EXPERT =>
$expertFilter,
                self::QUERY_PARAMETER_FILTER_TYPE => $typeFilter,
                self::QUERY_PARAMETER_SORT_BY => $sortBy,
                self::QUERY_PARAMETER_PAGE => $page,
            ],
            'filter' => [
                'sortValues' => self::QUERY_PARAMETER_SORT_VALUES,
            ],
            'listAjaxUrl' => $this->generateUrl('courses_and_webinars',
                [
                    self::QUERY_PARAMETER_SEARCH =>
'SEARCH_VALUE',
                    self::QUERY_PARAMETER_FILTER_CATEGORY =>
'CATEGORY_VALUES',
                    self::QUERY_PARAMETER_FILTER_CROP =>
'CROP_VALUES',
                    self::QUERY_PARAMETER_FILTER_PARTNER =>
'PARTNER_VALUES',
                    self::QUERY_PARAMETER_FILTER_EXPERT =>
'EXPERT_VALUES',
                    self::QUERY_PARAMETER_FILTER_TYPE =>
'TYPE_VALUES',
                    self::QUERY_PARAMETER_SORT_BY => 'SORT_VALUE',
                    self::QUERY_PARAMETER_PAGE => 'PAGE_VALUE',
                ]),
        ]);
    }
}
```

```
<?php

namespace App\Controller;

use Symfony\{
    Bundle\FrameworkBundle\Controller\AbstractController,
    Component\EventDispatcher\EventDispatcherInterface,
    Component\Routing\Annotation\Route,
    Component\HttpFoundation\Request,
    Component\HttpFoundation\Response,
    Component\HttpKernel\HttpKernelInterface,
    Component\HttpKernel\Exception\NotFoundHttpException,
    Component\Security\Core\User\UserInterface,
    Component\Security\Core\Security,
};
use App\{Entity\Article,
    Entity\Comment,
    Entity\Category,
    Entity\Page,
    Entity\Tags,
    Form\CommentType,
    Event\Item\RequestEvent as ItemRequestEvent,
    Helper\SeoHelper,
    Service\SeoService};

/
 * Class ArticleController
 * @package App\Controller
 */
class ArticleController extends AbstractController
{
    private const COMMENTS_PAGE_SIZE = 5;

    private const PAGE_TYPE_CODE = 'article';

    protected EventDispatcherInterface $eventDispatcher;

    protected HttpKernelInterface $kernel;

    protected Security $security;

    public function __construct(
        EventDispatcherInterface $eventDispatcher,
        HttpKernelInterface $kernel,
        Security $security
```

```
    ) {
        $this->eventDispatcher = $eventDispatcher;
        $this->kernel = $kernel;
        $this->security = $security;
    }

    /
     * @Route("/article/{slug}", name="article_detail")
     *
     * @param Request $request
     * @param UserInterface|null $user
     * @param $slug
     *
     * @return \Symfony\Component\HttpFoundation\Response
     */
    public function detail(
        Request $request,
        SeoService $seoService,
        ?UserInterface $user,
        $slug
    ): Response {
        $categories = $this->getDoctrine()->getRepository(Category::class)-
>findActiveCategories();
        $pageList = $this->getDoctrine()->getRepository(Page::class)-
>findPageByTypeName('business_tools');
        //todo add reviews when it will be ready
        / @var Article|null $article */
        $article = $this->getDoctrine()->getRepository(Article::class)-
>findOneBy(['slug' => $slug]);
        if (is_null($article) or $article->getIsActive() != true) {
            throw new NotFoundHttpException();
        }
        if ($article->getSimilar()->isEmpty()) {
            $similarItemList = $this
                ->getDoctrine()
                ->getRepository(Article::class)
                ->getSimilar(self::PAGE_TYPE_CODE,$article-
>getItemCropsAndCategoriesIds());
        } else {
            $similarItemList = $article->getSimilar();
        }
        / @var Comment[] $comments */
        $commentsRepository = $this
            ->getDoctrine()
            ->getRepository(Comment::class);
```

```
        $commentsFilter = [
            'item' => $article->getId(),
        ];
        $comments = $commentsRepository->findBy(
            $commentsFilter,
            ['createdAt' => 'DESC'],
            self::COMMENTS_PAGE_SIZE
        );
        $commentsTotalCount = $commentsRepository-
>getCount($commentsFilter);

        $event = new ItemRequestEvent(
            $this->kernel,
            $request,
            HttpKernelInterface::MASTER_REQUEST
        );
        $event->setItem($article);
        $this->eventDispatcher->dispatch($event);

        $seo = $seoService
            ->setPage(SeoHelper::PAGE_ARTICLE)
            ->getSeo(['title' => $article->getTitle(), 'category' => $article-
>getCategory()])
        ;

        $lastModified = SeoHelper::formatLastModified($article-
>getUpdatedAt());

        $response = new Response();
        $response->headers->set('Last-Modified', $lastModified);

return $this->render('article/detail.html.twig', [
            'seo' => $seoService->merge($seo, $article->getSeo()),
            'categories' => $categories,
            'topTags' => $article->getTags(),
            'tagTitleFlag' => true,
            'pagesList' => $pageList,
            'article' => $article,
            'similar' => $similarItemList,
            'user' => $user,
            'comments' => [
                'exist' => $comments,
                'form' => $this->createForm(CommentType::class)->createView(),
                'pageSize' => self::COMMENTS_PAGE_SIZE,
                'totalCount' => $commentsTotalCount,
```

```
            ],
        ], $response);
    }
}
```

```
<?php

namespace App\Controller\Admin;

use EasyCorp\Bundle\EasyAdminBundle\Config\{Assets, Actions, Action,
Crud, Filters};
use EasyCorp\Bundle\EasyAdminBundle\Field\{IdField, ImageField,
TextField, TextareaField};
use EasyCorp\Bundle\EasyAdminBundle\Controller\AbstractCrudController;
use App\Entity\Options;
use Symfony\Component\Filesystem\Filesystem;
use Symfony\Component\HttpFoundation\RequestStack;

class OptionsCrudController extends BaseCrudController
{
    public static function getEntityFqcn(): string
    {
        return Options::class;
    }

    protected $requestStack;

    public function __construct(RequestStack $requestStack)
    {
        $this->requestStack = $requestStack;
    }

    public function configureFields(string $pageName): iterable
    {
        $kernelDir = $this->get('parameter_bag')->get('kernel.project_dir');
        $dir = $kernelDir . '/public/upload/options';
        $fs = new Filesystem();
        $fs->mkdir($dir);
        yield IdField::new('id')->hideOnForm();
        yield TextField::new('code')
            ->setLabel('Код налаштування')
```

```
            ->setColumns('col-sm-6 col-md-4')
            ->setFormTypeOption('disabled', 'disabled');
        $value = TextField::new('value')
            ->setColumns('col-sm-6 col-md-4')
            ->setLabel('Значення');

        if (Crud::PAGE_EDIT == $pageName) {
            $this->checkValueIsRequiredField()
                ? yield $value->setRequired(false)
                : yield $value->setRequired(true);
        } else {
            yield $value;
        }

        yield TextField::new('description')
            ->setColumns('col-sm-6 col-md-4')
            ->setLabel('Опис');
        $img = ImageField::new('imageName')
            ->setUploadDir('public/upload/options')
            ->setUploadedFileNamePattern('[timestamp]-
[contenthash].[extension]')
            ->setBasePath('upload/options')
            ->setColumns('col-sm-6 col-md-4')
            ->setLabel('Зображення');
        yield $img->setRequired(false);
    }

    public function configureCrud(Crud $crud): Crud
    {
        return $crud
            ->setDefaultSort(['id' => 'ASC'])
            ->setEntityLabelInPlural('admin.dashboard.option.list_title')
            ->setEntityLabelInSingular('admin.dashboard.option.edit_button_title')
            ->setPageTitle(Crud::PAGE_NEW,
'admin.dashboard.option.new_page_title')
            ->setPageTitle(Crud::PAGE_EDIT,
'admin.dashboard.option.edit_page_title')
            ->setSearchFields(['code', 'description', 'value'])
            ->setFormThemes(
                [
                    '@A2lixTranslationForm/bootstrap_4_layout.html.twig',
                    '@FOSCKEditor/Form/ckeditor_widget.html.twig',
                    '@EasyAdmin/crud/form_theme.html.twig',
                ]
            )
```

```
            ->showEntityActionsInlined();
    }

    public function configureActions(Actions $actions): Actions
    {
        return parent::configureActions($actions)
            ->disable(Action::NEW, Action::DELETE);
    }

    public function configureFilters(Filters $filters): Filters
    {
        return $filters
            ->add('code')
            ->add('value')
            ->add('description');
    }

    public function checkValueIsRequiredField()
    {
        $request = $this->requestStack->getCurrentRequest();
        $optionId = $request->query->get('entityId');
        $option = $this->getDoctrine()->getRepository(Options::class)-
>findOneBy(['id' => $optionId]);

        return str_starts_with($option->getCode(), 'homepage_footer_');
    }
}

<?php
declare(strict_types=1);

namespace App\Repository;

use DateInterval;
use DatePeriod;
use Throwable;
use DateTime;
use Doctrine\ORM\QueryBuilder;
use Doctrine\ORM\Query\Expr\Join;
use Doctrine\Persistence\ManagerRegistry;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Symfony\Component\Security\Core\{
    Exception\UnsupportedUserException,
    User\PasswordUpgraderInterface,
    User\UserInterface,
```

```
};
use App\Repository\Traits\{
    FilterApplierTrait,
    CountPerCreatedDateTrait,
    UserHelperTrait,
};
use App\Repository\Helper\AliasMap;
use App\Entity\{Crop, Region, User, Market\Commodity,
Market\UserFavorite};
use function get_class;
use function in_array;

/
 * @method User|null find($id, $lockMode = null, $lockVersion = null)
 * @method User|null findOneBy(array $criteria, array $orderBy = null)
 * @method User[]    findAll()
 * @method User[]    findBy(array $criteria, array $orderBy = null, $limit =
null, $offset = null)
 */
class UserRepository extends ServiceEntityRepository implements
PasswordUpgraderInterface
{
    use FilterApplierTrait;
    use CountPerCreatedDateTrait;
    use UserHelperTrait;

    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, User::class);
    }

    /
     * Used to upgrade (rehash) the user's password automatically over time.
     */
    public function upgradePassword(UserInterface $user, string
$newEncodedPassword): void
    {
        if (!$user instanceof User) {
            throw new UnsupportedUserException(sprintf('Instances of "%s" are
not supported.', get_class($user)));
        }

        $user->setPassword($newEncodedPassword);
        $this->_em->persist($user);
        $this->_em->flush();
```

```
}

/
 * Get count "custom" version.
 *
 * @param array $filter Filter.
 *
 * @return  int                     Users count.
 */
public function countAlt(array $filter): int
{
    $alias = 'user';
    $builder = $this
        ->createQueryBuilder($alias)
        ->select("count($alias.id)");

    $this->applyFilter($builder, $alias, $filter);

    try {
        return $builder
            ->getQuery()
            ->getSingleScalarResult();
    } catch (Throwable $exception) {
        return 0;
    }
}

public function countToday($gander)
{
    $dateToday = new DateTime('today');
    $dateTomorrow = new DateTime('tomorrow');
    $builder = $this
        ->createQueryBuilder('user')
        ->select('count(user.id)');
    if (is_null($gander)) {
        $builder->andWhere('user.gender is NULL');
    } else {
        $builder->andWhere('user.gender = ' . (int)$gander);
    }

    $builder->andWhere($builder->expr()->between(
        'user.createdAt',
        ':from',
        ':to'
    ))
```

```
            ->setParameters(
                [
                    'from' => $dateToday->format('Y-m-d H:i:s'),
                    'to' => $dateTomorrow->format('Y-m-d H:i:s')
                ]
            );
        try {
            return $builder
                ->getQuery()
                ->getSingleScalarResult();
        } catch (Throwable $exception) {
            return 0;
        }
    }

    /
     * Get items count per time.
     *
     * @param DateTime $from Date from.
     * @param DateTime $to Date to.
     * @param array $filter Filter.
     *
     * @return  array                   Data, where
     *                              key is date and
     *                              value is items count for that day.
     */
    public function getRegistrationsCountPerTime(DateTime $from, DateTime
$to, array $filter = []): array
    {
        return $this->getItemsCountPerCreatedDate($from, $to, $filter);
    }

/
     * Get users count per regions.
     *
     * @param string $locale Locale.
     *
     * @return  array                   Data, where
     *                              key is region and
     *                              value is users count for that regions.
     */
    public function getCountPerRegions(string $locale): array
    {
        $aliasUser = 'user';
        $aliasRegion = 'region';
```

```
        $aliasRegionTranslation = "{$aliasRegion}_translation";
        $data = $this
            ->createQueryBuilder($aliasUser)
            ->leftJoin(
                Region::class,
                $aliasRegion,
                Join::WITH,
                "$aliasUser.region = $aliasRegion.id"
            )
            ->leftJoin(
                "$aliasRegion.translations",
                $aliasRegionTranslation,
                Join::WITH,
                "$aliasUser.region = $aliasRegion.id"
            )
            ->select([
                "$aliasRegionTranslation.name as regionName",
            ])
            ->andWhere("$aliasRegionTranslation.locale = :locale")
            ->setParameter('locale', $locale)
            ->getQuery()
            ->getResult();
        $result = [];

        foreach ($data as $item) {
            $regionName = $item['regionName'];
            $result[$regionName] = $result[$regionName] ?? 0;
            $result[$regionName]++;
        }

        return $result;
    }

    /
     * Market list filter provider.
     *
     * @param User|null $user User.
     * @param string|null $order Order.
     * @param array $filter Filter.
     *
     * @return  QueryBuilder              Query builder.
     */
    public function marketListFilter(?User $user, ?string $order, array $filter =
[]): QueryBuilder
    {
```

```
    $aliasMap = (new AliasMap())
        ->setAlias(User::class, 'user');
    $rootAlias = $aliasMap->getAlias(User::class);
    $queryBuilder = $this
        ->createQueryBuilder($rootAlias)
        ->addSelect($rootAlias);

    $this->applyUserSubEntitiesQuery($queryBuilder, $aliasMap);
    $this->applyUsersFavoritesQuery($queryBuilder, $user, $aliasMap);

    $this->applyActivityFilter($queryBuilder, $filter);
    $this->applyMarketListOrder($queryBuilder, $order);
    $this->applyMarketListFilter($queryBuilder, $filter, $aliasMap);

    return $queryBuilder;
}

/
 * Get favorites count for given user.
 *
 * @param User|null $user User.
 * @param array $filter Filter.
 *
 * @return  int                    Favorites count.
 */
public function getTotalCount(?User $user, array $filter = []): int
{
    $queryBuilder = $this->marketListFilter($user, null, $filter);
    $alias = $queryBuilder->getRootAliases()[0];

    return count($queryBuilder
        ->select("$alias.id")
        ->groupBy("$alias.id")
        ->getQuery()
        ->getResult());
}

/
 * Apply activity filter.
 *
 * @param QueryBuilder $queryBuilder Query builder.
 * @param array $filter Filter.
 *
 * @return  void
 */
```

```
    private function applyActivityFilter(QueryBuilder $queryBuilder, array
$filter): void
    {
        $activity = (array)($filter['activity'] ?? []);

        if (in_array(true, $activity, true) && in_array(false, $activity, true)) {
            return;
        }
        if (in_array(false, $activity, true)) {
            $this->applyMarketUserInactivityFilter($queryBuilder);
        } else {
            $this->applyMarketUserActivityFilter($queryBuilder);
        }
    }

/
     * Apply market list order.
     *
     * @param QueryBuilder $queryBuilder Query builder.
     * @param string|null $order Order.
     *
     * @return  void
     */
    private function applyMarketListOrder(QueryBuilder $queryBuilder, ?string
$order): void
    {
        $alias = $queryBuilder->getRootAliases()[0];

        switch ($order) {
            case 'id':
                $queryBuilder->orderBy("$alias.id", 'asc');
                break;
            case null:
            default:
        }
    }

    /
     * Apply products list filters.
     *
     * @param QueryBuilder $queryBuilder Query builder.
     * @param array $filter Filter.
     * @param AliasMap $aliasMap Alias map.
     *
     * @return  void
```

```
     */
    private function applyMarketListFilter(
        QueryBuilder $queryBuilder,
        array        $filter,
        AliasMap     $aliasMap
    ): void
    {
        $alias = $queryBuilder->getRootAliases()[0];
        $inFavorite = (bool)($filter['inFavorite'] ?? false);
        $search = trim($filter['search'] ?? '');
        $filterToApply = [];

        if (isset($filter['region'])) {
            $queryBuilder->andWhere(
                $queryBuilder->expr()->eq(
                    "{$aliasMap->getAlias(User::class)}.region",
                    ':regionId'
                )
            );
            $queryBuilder->setParameter('regionId', $filter['region']);
        }
        if (isset($filter['crop'])) {
            $aliasMap->setAlias(Crop::class, 'crop');
            $queryBuilder->leftJoin(
                "{$aliasMap->getAlias(User::class)}.crops",
                $aliasMap->getAlias(Crop::class)
            );

            $queryBuilder->andWhere(
                $queryBuilder->expr()->eq(
                    "{$aliasMap->getAlias(Crop::class)}.id",
                    ':cropId'
                )
            );
            $queryBuilder->setParameter('cropId', $filter['crop']);
        }
        if (isset($filter['created_at'])) {
            $queryBuilder
                ->andWhere($queryBuilder->expr()->lte("{$aliasMap-
>getAlias(User::class)}.createdAt", ':date'))
                ->setParameter('date', $filter['created_at']);
        }
        if (isset($filter['gender'])) {
            $filterToApply['gender'] = $filter['gender'];
        }
```

```
        if (isset($filter['roles'])) {
            $this->setRolesFilter($queryBuilder, (array)$filter['roles']);
        }
        if (isset($filter['id'])) {
            $filterToApply['id'] = $filter['id'];
        }
        if (isset($filter['!id'])) {
            $filterToApply[] = [
                'field' => 'id',
                'condition' => '!=',
                'value' => $filter['!id'],
            ];
        }
        if ($inFavorite) {
            $queryBuilder->andWhere("{$aliasMap-
>getAlias(UserFavorite::class)}.id IS NOT NULL");
        }
        if (isset($filter['commodities'])) {
            $commodities = (array)($filter['commodities'] ?? []);
            $aliasMap->setAlias(Commodity::class, 'userCommodities');

            $queryBuilder
                ->leftJoin("$alias . commodities", $aliasMap-
>getAlias(Commodity::class))
                ->andWhere("{$aliasMap->getAlias(Commodity::class)}.id
IN(:userCommodities)")
                ->setParameter('userCommodities', count($commodities) > 0 ?
$commodities : ['none']);
        }
        if (strlen($search) > 0) {
            $queryBuilder
                ->andWhere(
                    $queryBuilder->expr()->orX(
                        $queryBuilder->expr()->like("$alias . name", ':searchString'),
                        $queryBuilder->expr()->eq("$alias . email", ':searchEmail'),
                        $queryBuilder->expr()->eq("$alias . id", ':searchId'),
                    )
                )
                ->setParameter('searchString', " % $search % ")
                ->setParameter('searchEmail', $search)
                ->setParameter('searchId', (int)$search);
        }

        $this->applyFilter($queryBuilder, $alias, $filterToApply);
    }
```

```
/**
     * Set user roles filter
     *
     * @param QueryBuilder $queryBuilder Query builder.
     * @param string[] $roles Roles filter.
     *
     * @return  void
     */
    private function setRolesFilter(QueryBuilder $queryBuilder, array $roles):
void
    {
        if (count($roles) === 0) {
            return;
        }
        $alias = $queryBuilder->getRootAliases()[0];
        $orX = $queryBuilder->expr()->orX();
        foreach ($roles as $role) {
            $orX->add($queryBuilder->expr()->like("$alias . roles", "'%$role%'"));
        }

        $queryBuilder->andWhere($orX);
    }
}

<?php
declare(strict_types=1);

namespace App\Repository;

use App\Entity\TypePage;
use DateTime;
use Doctrine\ORM\NonUniqueResultException;
use Doctrine\ORM\NoResultException;
use Doctrine\ORM\Query;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;
use App\Repository\Traits\{
    FilterApplierTrait,
    CountPerCreatedDateTrait,
};
use App\Entity\Article;
use function count;

/
```

```
 * @method Article|null find($id, $lockMode = null, $lockVersion = null)
 * @method Article|null findOneBy(array $criteria, array $orderBy = null)
 * @method Article[]    findAll()
 * @method Article[]    findBy(array $criteria, array $orderBy = null, $limit =
null, $offset = null)
 */
class ArticleRepository extends ServiceEntityRepository
{
    use FilterApplierTrait;
    use CountPerCreatedDateTrait;

    private const ARTICLES_ALIAS = 'articles';
    private const REGIONS_ALIAS = 'regions';
    private const TRANSLATIONS_ALIAS = 'translations';

    private int $maxCount;

    public function __construct(ManagerRegistry $registry)
    {
        $this->maxCount = 9;
        parent::__construct($registry, Article::class);
    }

    /
     * @param string $code
     *
     * @return int|mixed|string
     */
    public function getSimilar(string $code, array $data)
    {

        $q = $this->createQueryBuilder('a')
            ->leftJoin('a.typePage', 'tp')
            ->andWhere('tp.code = :code')
            ->andWhere('a.isActive = true')
            ->setParameter('code', $code);
        if (array_key_exists('cropsIds', $data)) {
            $q->leftJoin('a.crops', 'cr')
                ->andWhere('cr.id IN (:cropsIds)')
                ->setParameter('cropsIds', $data['cropsIds']);
        }
        if (array_key_exists('categoryId', $data)) {
            $q->leftJoin('a.category', 'cat')
                ->andWhere('cat.id = :categoryId')
                ->setParameter('categoryId', $data['categoryId']);
```

```
        }
        if (array_key_exists('id', $data)) {
            $q->andWhere('a.id != :id')
                ->setParameter('id', $data['id']);
        }


        $q->orderBy('a.createdAt', 'DESC')
            ->setMaxResults($this->maxCount);

        return $q->getQuery()->getResult();
    }

    /
     * @param $pageType
     * @param bool $isActive
     * @return int
     * @throws NonUniqueResultException
     */
    public function getCountArticle(array $pageTypes =
[TypePage::TYPE_ARTICLE], bool $isActive = true): int
    {
        $count = count($pageTypes);
        $query = $this->createQueryBuilder('a');
        $query
            ->select('count(a.id) as count')
            ->andWhere(
                $query->expr()->eq('a.typePage', $pageTypes[0])
            );
        if ($count > 1) {
            for ($i = 1; $i < $count; $i++) {
                $query->orWhere(
                    $query->expr()->eq('a.typePage', $pageTypes[$i])
                );
            }
        }
        return $query
            ->andWhere('a.isActive = :isActive')
            ->setParameter('isActive', $isActive)
            ->getQuery()
            ->getOneOrNullResult()['count'];
    }


    /
```

```
     * @return int|mixed|string
     */
    public function getTopArticles() //todo chacnge this to improve forming
collection of similar articles
    {
        return $this->createQueryBuilder('i')
            ->andWhere('i.top = true')
            ->andWhere('i.isActive = true')
            ->orderBy('i.id', 'DESC')
            ->setMaxResults($this->maxCount)
            ->getQuery()
            ->getResult();
    }

/
     * Get items count per regions.
     *
     * @param array $filter Articles filter.
     *
     * @return  array                   Data, where
     *                              key is region ID and
     *                              value is region articles count
     */
    public function getItemsPerRegionsCount(array $filter = []): array
    {
        / @var Article[] $articles */
        $regionsAlias = self::REGIONS_ALIAS;
        $articlesAlias = self::ARTICLES_ALIAS;
        $articlesBuilder = $this
            ->createQueryBuilder($articlesAlias)
            ->leftJoin("$articlesAlias.region", $regionsAlias)
            ->select([
                "$articlesAlias.id as article",
                "$regionsAlias.id as region",
            ]);
        $result = [];

        foreach ($filter as $key => $value) {
            $condition = is_array($value)
                ? "$articlesAlias.$key IN (:$key)"
                : "$articlesAlias.$key = :$key";

            $articlesBuilder->andWhere($condition);
            $articlesBuilder->setParameter($key, $value);
        }
```

```
    $articlesData = $articlesBuilder
        ->getQuery()
        ->getResult(Query::HYDRATE_ARRAY);

    foreach ($articlesData as $articleData) {
        $regionId = (int)$articleData['region'];
        $result[$regionId] = $result[$regionId] ?? 0;
        $result[$regionId]++;
    }

    return $result;
}

/
 * Get items by filter parameters
 *
 * @param array $filter Filter.
 * @param string $locale Locale.
 * @param array $order Order.
 * @param int|null $limit
 * @param int|null $offset
 *
 * @return  Article[]               Articles set.
 */
public function findByAlt(
    array  $filter,
    string $locale,
    array  $order = [],
    ?int   $limit = null,
    ?int   $offset = null
): array
{
    $articlesAlias = self::ARTICLES_ALIAS;
    $translationsAlias = self::TRANSLATIONS_ALIAS;
    $articlesBuilder = $this
        ->createQueryBuilder(self::ARTICLES_ALIAS)
        ->andWhere("$articlesAlias.isActive = true")
        ->join("$articlesAlias.translations", $translationsAlias);

    foreach ($filter as $key => $value) {
        switch ($key) {
            case 'content':
                $articlesBuilder->andWhere("
                (
```

```
                (
                    $translationsAlias.title LIKE :content OR
                    $translationsAlias.content LIKE :content
                ) AND
                $translationsAlias.locale = :locale
            )");
            $articlesBuilder->setParameter('content', "%$value%");
            $articlesBuilder->setParameter('locale', $locale);
            break;
        default:
            if (is_array($value) && in_array(null, $value)) {
                unset($value[array_search(null, $value)]);

                $articlesBuilder->andWhere("
                (
                    $articlesAlias.$key IN (:$key) OR
                    $articlesAlias.$key IS NULL
                )
                ");
                $articlesBuilder->setParameter($key, $value);
            } elseif (is_array($value)) {
                $articlesBuilder->andWhere("$articlesAlias.$key IN (:$key)");
                $articlesBuilder->setParameter($key, $value);
            } else {
                $articlesBuilder->andWhere("$articlesAlias.$key = :$key");
                $articlesBuilder->setParameter($key, $value);
            }
    }
}

if (count($order) > 0) {
    $sortValue = array_key_first($order);
    $order = array_values($order)[0];

    switch ($sortValue) {
        case 'title':
            $sort = "$translationsAlias.title";
            brea

k;
        default:
            $sort = "$articlesAlias.$sortValue";
    }

    $articlesBuilder->orderBy($sort, $order);
```

```
    }

    if ($limit) {
        $articlesBuilder->setMaxResults($limit);
    }
    if ($offset) {
        $articlesBuilder->setFirstResult($offset);
    }

    return $articlesBuilder
        ->getQuery()
        ->getResult();
}

/
 * @throws NonUniqueResultException
 * @throws NoResultException
 */
public function getCountSuccessesStory()
{
    return $this->createQueryBuilder('a')
        ->select('count(a.id)')
        ->join('a.typePage', 'pt')
        ->andWhere("pt.code = 'success_stories'")
        ->andWhere('a.isActive = true')
        ->getQuery()
        ->getSingleScalarResult();
}

public function findArticleByTypeName($typeCode, $catId)
{
    return $this->createQueryBuilder('a')
        ->leftJoin('a.typePage', 'tp')
        ->leftJoin('a.category', 'c')
        ->andWhere('c.id = :catId')
        ->andWhere('tp.code = :code')
        ->andWhere('a.isActive = true')
        ->setParameter('code', $typeCode)
        ->setParameter('catId', $catId)
        ->setMaxResults(20)
        ->orderBy('a.createdAt','DESC')
        ->getQuery()
        ->getResult();
}
```

```
    public function findArticleOnlyByTypeName($typeCode,int $maxCount =
null)
    {
        return $this->createQueryBuilder('a')
            ->leftJoin('a.typePage', 'tp')
            ->andWhere('tp.code = :code')
            ->andWhere('a.isActive = true')
            ->setParameter('code', $typeCode)
            ->setMaxResults($maxCount ? : 20)
            ->getQuery()
            ->getResult();
    }

    /**
     * Get items count per time.
     *
     * @param DateTime $from Date from.
     * @param DateTime $to Date to.
     * @param array $filter Filter.
     *
     * @return  array                   Data, where
     *                              key is date and
     *                              value is items count for that day.
     */
    public function getCountPerTime(DateTime $from, DateTime $to, array
$filter = []): array
    {
        return $this->getItemsCountPerCreatedDate($from, $to, $filter);
    }
}

<?php
declare(strict_types=1);

namespace App\Repository;

use App\Entity\Category;
use App\Entity\News;
use DateTime;
use Doctrine\ORM\NonUniqueResultException;
use Doctrine\ORM\Query;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;
use App\Repository\Traits\{
    FilterApplierTrait,
```

```
    CountPerCreatedDateTrait,
};
use App\Entity\Article;

/
 * @method News|null find($id, $lockMode = null, $lockVersion = null)
 * @method News|null findOneBy(array $criteria, array $orderBy = null)
 * @method News[]    findAll()
 * @method News[]    findBy(array $criteria, array $orderBy = null, $limit =
null, $offset = null)
 */
class NewsRepository extends ServiceEntityRepository
{
    use FilterApplierTrait;
    use CountPerCreatedDateTrait;

    private const NEWS_ALIAS = 'news';
    private const REGIONS_ALIAS = 'regions';
    private const TRANSLATIONS_ALIAS = 'translations';

    private int $maxCount;

    public function __construct(ManagerRegistry $registry)
    {
        $this->maxCount = 9;
        parent::__construct($registry, News::class);
    }

    /
     * @return int|mixed|string
     */
    public function getSimilar(array $data)
    {

        $q = $this->createQueryBuilder('n')
            ->andWhere('n.isActive = true');
        if (array_key_exists('cropsIds', $data)) {
            $q->leftJoin('n.crops', 'cr')
                ->andWhere('cr.id IN (:cropsIds)')
                ->setParameter('cropsIds', $data['cropsIds']);
        }
        if (array_key_exists('categoryId', $data)) {
            $q->leftJoin('n.category', 'cat')
                ->andWhere('cat.id = :categoryId')
                ->setParameter('categoryId', $data['categoryId']);
```

```
        }
        if (array_key_exists('id', $data)) {
            $q->andWhere('n.id != :id')
                ->setParameter('id', $data['id']);
        }


        $q->orderBy('n.createdAt', 'DESC')
            ->setMaxResults($this->maxCount);

        return $q->getQuery()->getResult();
    }

    /
     * @param $pageType
     * @param bool $isActive
     * @return int
     * @throws NonUniqueResultException
     */
    public function getCountArticle($pageType, bool $isActive = true): int
    {
        return $this->createQueryBuilder('a')
            ->select('count(a.id) as count')
            ->andWhere('a.typePage = :typePage')
            ->andWhere('a.isActive = :isActive')
            ->setParameter('typePage', $pageType)
            ->setParameter('isActive', $isActive)
            ->getQuery()
            ->getOneOrNullResult()['count'];
    }


    /
     * @return int|mixed|string
     */
    public function getTopArticles() //todo change this to improve forming
collection of similar articles
    {
        return $this->createQueryBuilder('i')
            ->andWhere('i.top = true')
            ->andWhere('i.isActive = true')
            ->orderBy('i.id', 'DESC')
            ->setMaxResults($this->maxCount)
            ->getQuery()
            ->getResult();
```

```
    }

    /
     * Get items count per regions.
     *
     * @param array $filter Articles filter.
     *
     * @return  array                   Data, where
     *                                  key is region ID and
     *                                  value is region articles count
     */
    public function getItemsPerRegionsCount(array $filter = []): array
    {
        / @var Article[] $articles */
        $regionsAlias = self::REGIONS_ALIAS;
        $articlesAlias = self::NEWS_ALIAS;
        $articlesBuilder = $this
            ->createQueryBuilder($articlesAlias)
            ->leftJoin("$articlesAlias.region", $regionsAlias)
            ->select([
                "$articlesAlias.id as article",
                "$regionsAlias.id as region",
            ]);
        $result = [];

        foreach ($filter as $key => $value) {
            $condition = is_array($value)
                ? "$articlesAlias.$key IN (:$key)"
                : "$articlesAlias.$key = :$key";

$articlesBuilder->andWhere($condition);
            $articlesBuilder->setParameter($key, $value);
        }

        $articlesData = $articlesBuilder
            ->getQuery()
            ->getResult(Query::HYDRATE_ARRAY);

        foreach ($articlesData as $articleData) {
            $regionId = (int)$articleData['region'];
            $result[$regionId] = $result[$regionId] ?? 0;
            $result[$regionId]++;
        }

        return $result;
```

```
}

/**
 * Get items by filter parameters
 *
 * @param array $filter Filter.
 * @param string $locale Locale.
 * @param array $order Order.
 * @param int|null $limit
 * @param int|null $offset
 *
 * @return  News[]                Articles set.
 */
public function findByAlt(
    array  $filter,
    string $locale,
    array  $order = [],
    ?int   $limit = null,
    ?int   $offset = null
): array
{
    $articlesAlias = self::NEWS_ALIAS;
    $translationsAlias = self::TRANSLATIONS_ALIAS;
    $articlesBuilder = $this
        ->createQueryBuilder(self::NEWS_ALIAS)
        ->andWhere("$articlesAlias.isActive = true")
        ->join("$articlesAlias.translations", $translationsAlias);

    foreach ($filter as $key => $value) {
        switch ($key) {
            case 'content':
                $articlesBuilder->andWhere("
                (
                    (
                        $translationsAlias.title LIKE :content OR
                        $translationsAlias.content LIKE :content
                    ) AND
                    $translationsAlias.locale = :locale
                )");
                $articlesBuilder->setParameter('content', "%$value%");
                $articlesBuilder->setParameter('locale', $locale);
                break;
            default:
                if (is_array($value) && in_array(null, $value)) {
                    unset($value[array_search(null, $value)]);
```

```
                $articlesBuilder->andWhere("
                (
                    $articlesAlias.$key IN (:$key) OR
                    $articlesAlias.$key IS NULL
                )
                ");
                $articlesBuilder->setParameter($key, $value);
            } elseif (is_array($value)) {
                $articlesBuilder->andWhere("$articlesAlias.$key IN (:$key)");
                $articlesBuilder->setParameter($key, $value);
            } else {
                $articlesBuilder->andWhere("$articlesAlias.$key = :$key");
                $articlesBuilder->setParameter($key, $value);
            }
        }
    }

    if (count($order) > 0) {
        $sortValue = array_key_first($order);
        $order = array_values($order)[0];

        switch ($sortValue) {
            case 'title':
                $sort = "$translationsAlias.title";
                break;
            default:
                $sort = "$articlesAlias.$sortValue";
        }

        $articlesBuilder->orderBy($sort, $order);
    }

    if ($limit) {
        $articlesBuilder->setMaxResults($limit);
    }
    if ($offset) {
        $articlesBuilder->setFirstResult($offset);
    }

    return $articlesBuilder
        ->getQuery()
        ->getResult();
}
```

```
    public function getNewsByCategory(Category $category)
    {
      return $this->createQueryBuilder('n')
          ->where('n.isActive = true')
          ->andWhere('n.category = :cat')
          ->setParameter('cat',$category)
          ->setMaxResults(20)
          ->getQuery()->getResult();
    }

public function getLastModified()
    {
       return $this->createQueryBuilder('n')
          ->select('n.updatedAt')
          ->orderBy('n.updatedAt', 'ASC')
          ->setMaxResults(1)
          ->getQuery()
          ->getOneOrNullResult();
    }
}
```

ДОДАТОК Б
ВІДГУК
на кваліфікаційну роботу бакалавра
на тему:
"Розробка веб-додатка платформи аграрних знань за допомогою php 7.4 та Symfony 4. "
студента групи 121-18-2 Дорофеєва Олександра Андрійовича

ДОДАТОК В
ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

1. Дипломна_робота.doc
2. Веб-додаток.loc
3. Презентація.ppt

РЕЦЕНЗІЯ
на кваліфікаційну роботу бакалавра
на тему:
"Розробка веб-додатка платформи аграрних знань за допомогою php 7.4 та Symfony 4. "
студента групи 121-18-2 Дорофеєва Олександра Андрійовича