Одеський національний університет імені І.І.Мечникова

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних систем та технологій

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

Розробка сайту структурного підрозділу університету

(назва українською)

Development of the site of the structural unit of the university

(назва англійською)

Виконав: студент денної форми навчання
спеціальності 123 Комп'ютерна інженерія.

(шифр і назва напряму підготовки, спеціальності)

Яблонський Кирило Михайлович

(прізвище, ім'я, по-батькові)

Керівник д.т.н., проф. Гунченко Ю.О.

(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент ст. викл. Мартинович Л.Я

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№ 8 від «8» 06 2022 р.

Завідувач кафедри

_____ Ю.О. Гунченко

(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК № ___

протокол № __ від «___» ____ 2022р.

Оцінка _____/_____/_____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ Н.Ф. Казакова

(підпис) (прізвище, ініціали)

Одеса - 2022

АНОТАЦІЯ

У дипломній роботі розробляється тема «Розробка сайту структурного підрозділу університету». Мета роботи – розробити веб-сайт на якому було би зручно редагувати та додавати інформацію щодо звітності роботи Ради Молодих Вчених. Проведено аналіз потреб користувачів, розглянуті вже існуючі подібні ресурси, та на основі цього визначено необхідні функції та інші вимоги до системі.

Основним пріоритетом при розробці стала необхідність створити систему максимально доступною та зручною для користувача. Виходячи з цього було прийнято рішення реалізовувати систему як веб-ресурс з використанням PHP фреймворку Laravel.

Реалізовано базові можливості: CRUD всіх сутностей сайту, можливість в майбутньому легко додавати нові мови до сайту. Весь створений контент зберігається у серверній базі даних MySQL. Використані переваги фреймворку Laravel: механізм міграцій, захист від кросбраузерних атак.

В результаті отриманий додаток вийшов легко масштабованим і відкритим для подальшого розвитку.

Ключові слова: розробка сайту, ОНУ, MVC, PHP, Laravel, MySQL.

ANNOTATION

The thesis "Development of the site of the structural unit of the university" is developed in the thesis. The purpose of the work is to develop a website where it would be convenient to edit and add information on the reporting of the Council of Young Scientists. The analysis of user needs is carried out, already existing similar resources are considered, and on this basis the necessary functions and other requirements to the system are defined.

The main priority in the development was the need to create a system as accessible and user-friendly as possible. Based on this, it was decided to implement the system as a web resource using the Laravel PHP framework.

Implemented basic features: CRUD of all site entities, the ability to easily add new languages to the site in the future. All created content is stored in the MySQL back-end database. The advantages of the Laravel framework were used: migration mechanism, protection against cross-browser attacks.

As a result, the resulting application was easily scalable and open for further development.

Keywords: website development, ONU, MVC, PHP, Laravel, MySQL.

АННОТАЦИЯ

В дипломной работе разрабатывается тема "Разработка сайта структурного подразделения университета". Цель работы – разработать веб-сайт, на котором было бы удобно редактировать и добавлять информацию о отчетности работы Совета Молодых Ученых. Проведен анализ потребностей пользователей, рассмотрены уже существующие подобные ресурсы, и на основе этого определены необходимые функции и другие требования к системе.

Основным приоритетом при разработке стала необходимость создать систему максимально доступной и удобной для пользователя. Исходя из этого, было принято решение реализовывать систему как веб-ресурс с использованием PHP фреймворка Laravel.

Реализованы базовые возможности: CRUD всех сущностей сайта, возможность в будущем легко добавлять новые языки на сайт. Весь созданный контент хранится в серверной базе данных MySQL. Использованы преимущества фреймворка Laravel: механизм миграций, защита от кроссбраузерных атак.

В результате полученное приложение получилось легко масштабируемым и открытым для дальнейшего развития.

Ключевые слова: разработка сайта, ОНУ, MVC, PHP, Laravel, MySQL.

ЗМІСТ

Стор.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

МОН - Міністерство Освіти і Науки

РМВ - Рада Молодих вчених при ОНУ Мечникова

Трейт - метод повторного використання коду

Front end - це інтерфейс для взаємодії між користувачем і back end

Back end - все, що працює на сервері, тобто «не в браузері» або на комп'ютері, підключеному до мережі (зазвичай до Інтернету), який відповідає на повідомлення від інших комп'ютерів

Веб-додаток – клієнт-серверний додаток, в якому клієнт взаємодіє з сервером за допомогою браузера, а за сервер відповідає веб-сервер

БД – база даних

Email – адреса електронної пошти

Фреймворк – це програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту

CRUD - (англ. create read update delete) 4 основні функції управління даними «створення, читання, оновлення і вилучення»

ВСТУП

Веб-додаток - це клієнт-серверний додаток, (клієнтом є браузер, а в якості сервера виступає веб-сервер), при якому зберігання даних здійснюється головним чином на сервері, а обмін даними відбувається по мережі. З цього випливає, що для роботи з веб-додатком користувачеві необхідний доступ до інтернету. Серверна частина отримує запит від клієнта, виконує обчислення, після цього формує веб-сторінку і відправляє її клієнту через мережу з використанням протоколу HTTP / HTTPS.

Також останнім часом набирає велику популярність технологія WebSocket, яка не вимагає постійних запитів від клієнта до сервера, а створює двонаправлене з'єднання, при якому сервер може відправляти дані клієнта без запиту від останнього. Таким чином з'являється можливість динамічно управляти контентом в режимі реального часу.

До переваг веб-додатків можна віднести:

– веб-додатки не вимагають установки на телефоні, на відміну від мобільних, тому вони не займають місця на вашому смартфоні.

– для роботи необхідні тільки браузер і інтернет - користувачеві не потрібно встановлювати спеціальне програмне забезпечення для того, щоб користуватися веб-додатком.

– розробка клієнт-серверних додатків зазвичай обходиться дешевше, ніж мобільних.

– при внесенні оновлень в веб-додаток вони здійснюються на сайті автоматично.

– витрати тимчасових і грошових коштів нижче, ніж на нативні мобільні додатки.

А до недоліків веб-додатків можна віднести:

– доступ до додатку неможливий при відсутності інтернету;

– push-повідомлення можливі тільки при використанні сторонніх сервісів.

1 АНАЛІЗ ВИМОГ. ОГЛЯД ІСНУЮЧИХ МЕТОДІВ РІШЕНЬ

1.1 Аналіз вимог

Голова РМВ молодих вчених звернулася з завданням щодо розробки сайту цього підрозділу нашого університету. Сайт необхідний для показу складу, функцій, інформації про відділи і також для публікації новин та документів РМВ.

1.2 Огляд існуючих рішень

Було розглянуто нинішній вигляд виводу інформації про Раду. Це виявився один підрозділ головного сайту університету. Так він виглядає (див. рис. 1.1).

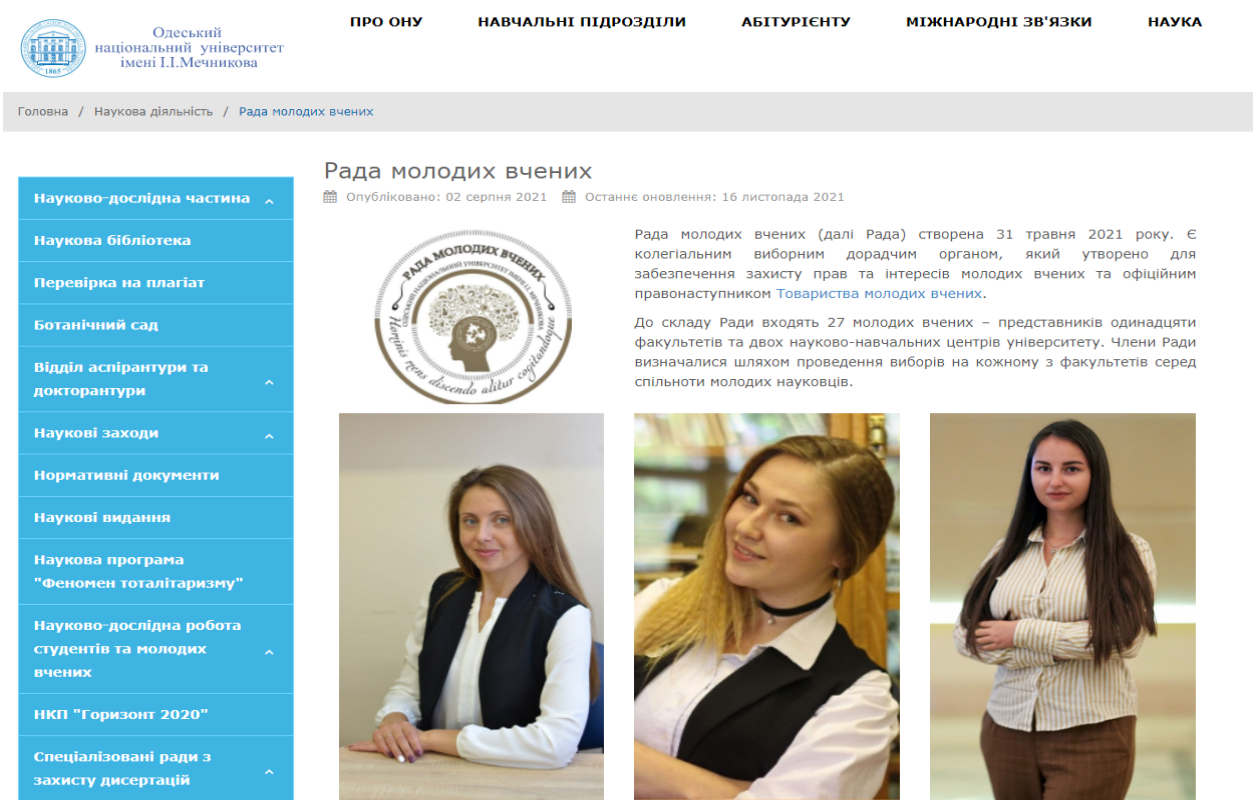

Рисунок 1.1 - підрозділ РМВ на головному сайті університету

У цього підрозділу немає місця для розміщення інформації щодо звітностності роботи РМВ.

Далі були переглянуті рішення для інших РМВ. Здебільшого інші РМВ розміщують інформацію на підрозділі головного сайту, але було знайдено РМВ, які розмістили свою інформацію на окремому сайті. А саме РМВ при МОН України (див. рис. 1.2), РМВ Київського Національного Університету (див. рис. 1.3) і РМВ Харківської медичної академії (див. рис. 1.4).

Переглянувши всі існуючі рішення представлення інформації для РМВ. Було вирішено взяти за основу дизайн РМВ при МОН України (див. рис. 1.2).

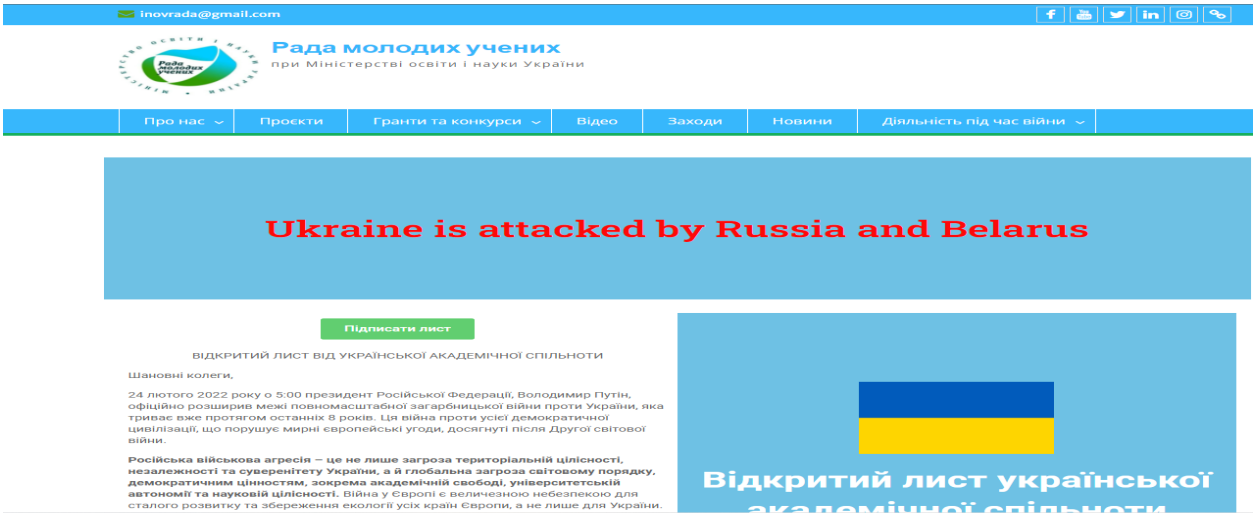

Рисунок. 1.2 - РМВ при МОН України

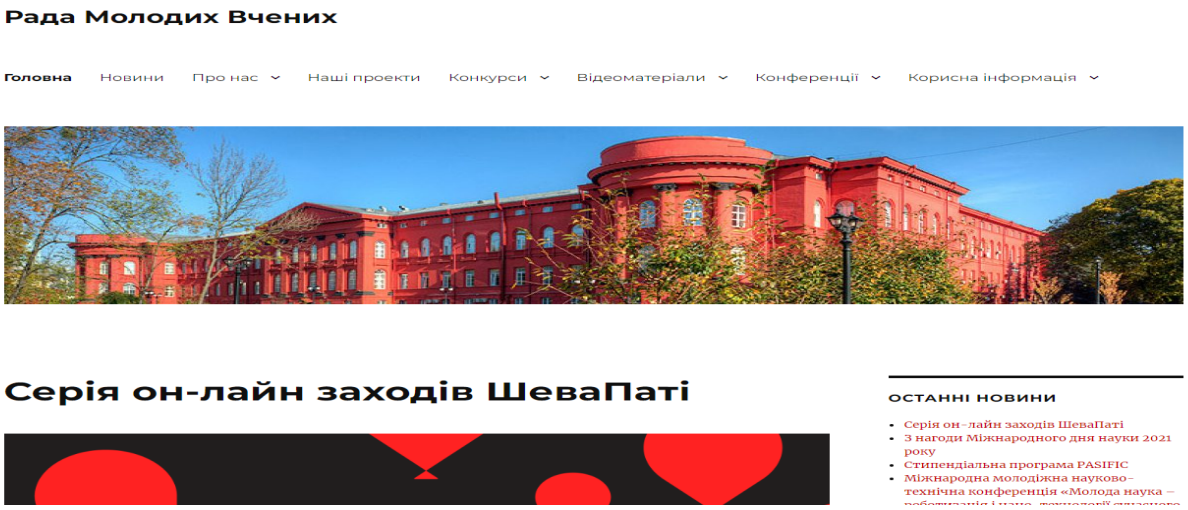

Рисунок 1.3 - РМВ Київського Національного Університету

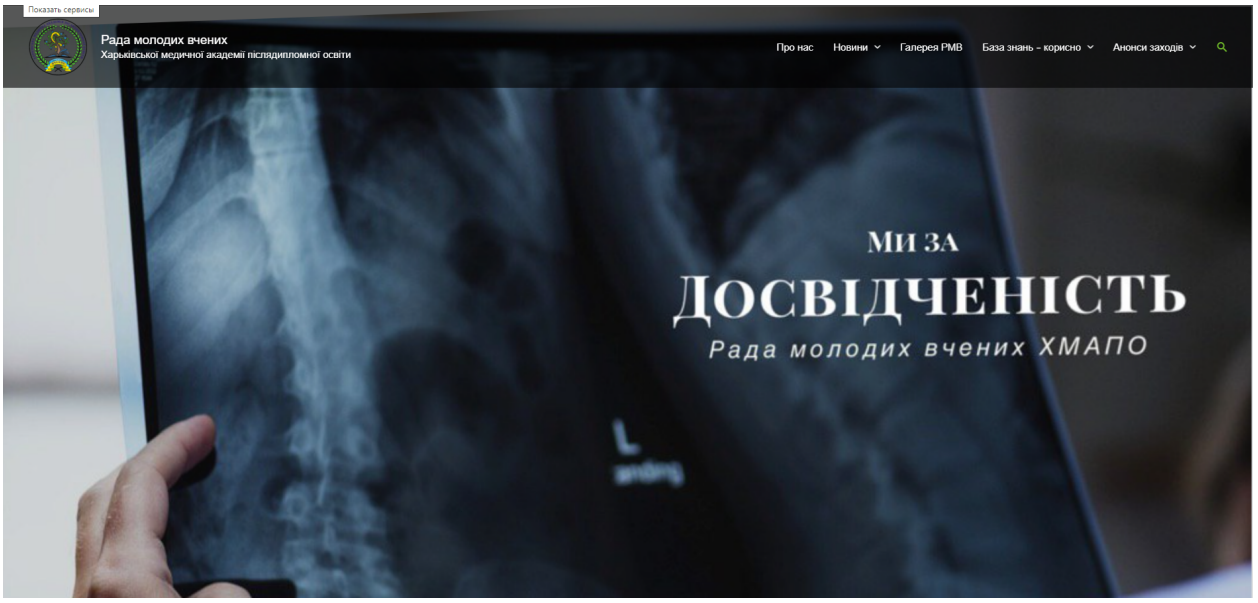

Рисунок 1.4 - РМВ Харківської медичної академії

1.3 Необхідні функції та інші вимоги до застосування

Основними функціями сайту РМВ є :

- Відображення та редагування інформації про діяльність;
- Відображення та редагування складу;
- Відображення та редагування інформації про відділи;
- Відображення та редагування новин;
- Відображення та додавання документів.

Всі ці функції мають працювати зручно, як для користувачів сайту, так і для адмінів. Для адмінів головною вимогою було редагування всіх сутностей сайту в адмін панелі. А для користувачів, зручний і мінімалістичний інтерфейс.

1.4 Висновки у розділі

У підрозділі 1.1 було визначено потреби користувачів, виявлено головні пріоритети розробки, а саме: програма повинна забезпечувати легкість доступу для користувача, масштабованість та безпеку даних. У підрозділі 1.2 було розглянуто три сайти для РМВ. У підрозділі 1.3 було сформульовано вимоги до системи.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ. ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

2.1 Визначення типу інформаційної системи

Потрібно розібратися з поняттям «інформаційна система». Інформаційна система (ІС) - «система, що організує обробку інформації про предметну область та її зберігання». Поняття ІС включає організаційні ресурси (людські, технічні, фінансові і т. д.), які забезпечують і поширюють інформацію. ІС призначена для своєчасного надання певним людям певної інформації. "Переважна більшість інформаційних систем працює в режимі діалогу з користувачем" [1]. За масштабом ІС поділяються на одиничні, групові, корпоративні та глобальні [1] (рис. 2.1).

Одиничні ІС реалізуються, як правило, на автономному персональному комп'ютері без використання комп'ютерної мережі. Така система може містити декілька простих додатків із спільним інформаційним фондом. Подібні комплекси можуть бути створені за допомогою таких локальних систем управління базами даних як Clipper, FoxPro, Paradox, MS Access тощо. Наприклад, "ІС: Бухгалтерія", АРМ.

Групові ІС орієнтовані на колективне використання інформації і найчастіше будуються на базі локальної обчислювальної мережі. При розробці таких додатків найчастіше використовуються сервери баз даних (SQL-сервери) для робочих груп. Серед найбільш відомих таких серверів є Oracle, InterBase, Sybase, тощо.

Корпоративні ІС призначені для великих компаній і можуть підтримувати територіально віддалені вузли і мережі. Як правило, вони мають ієрархічну клієнт-серверну структуру зі спеціалізацією серверів. При розробці таких систем можуть використовуватись ті ж сервери баз даних, що й при розробці групових ІС. Для корпоративних систем найбільш поширеними є сервери Oracle, DB2, Microsoft SQL Server.

Глобальні ІС охоплюють територію держави чи континенту. Прикладом такої інформаційної системи є глобальна мережа Інтернет [2].

Рисунок 2.1 – Класифікація ІС за масштабом

За характером обробки даних ІС поділяються на:

- інформаційно-довідкові (інформаційно-пошукові) ІС. У таких системах немає складних алгоритмів обробки даних, їх метою є пошук і видача інформації у зручному вигляді;
- ІС обробки даних (вирішальні ІС). Дані в них обробляються за складним алгоритмом.

Ще один критерій класифікації - архітектура, що використовується в основі (рис. 1.6):

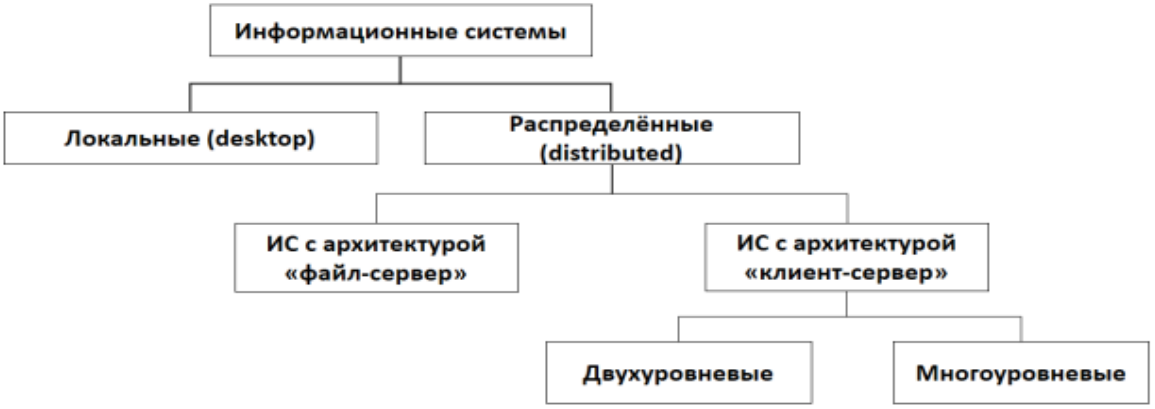

Рисунок 2.1 – Класифікація ІС з архітектури

Локальні ІС – всі компоненти знаходяться на одному комп'ютері.

Файл-серверні – на сервері дані лише витягуються з файлів, вся система управління базами даних (СУБД) та клієнтські додатки розташовуються на стороні користувача.

Клієнт-серверні – база даних (БД) та СУБД розташовуються на сервері, на стороні користувача лише клієнтські програми. Спочатку клієнт-серверні системи були дворівневими: «клієнт відповідав за відображення інтерфейсу користувача та виконання коду програми, а роль сервера зазвичай доручається СУБД. Проблеми виникли із ускладненням логіки предметної галузі – бізнес-правил, алгоритмів обчислень, умов перевірок тощо. Насамперед усі ці обов'язки покладалися на код клієнта та шукали відображення у вмісті інтерфейсних екранів. Чим складнішою ставала логіка, тим паче незграбним і важким сприйняття робився код». Все це призвело до появи багаторівневої системної архітектури, у межах якої користувацькі клієнтські програми не звертаються до СУБД безпосередньо, вони взаємодіють із проміжними рівнями.

Враховуючи теоретичні відомості, викладені вище, інформаційну систему «Рада молодих вчених» будемо створювати як корпоративну, інформаційно-довідкову систему з багаторівневою клієнт-серверною архітектурою.

2.2 Вибір шаблону проектування

Основною вимогою до системи є легкість її використання та доступність. Найбільш зручним рішенням, що задовольняє тип нашої системи, є Web-додаток. У такому додатку використовується трирівнева (тришарова) архітектура див. табл. 2.1:

Таблиця 2.1 - Тришарова архітектура

Шар	Функції
Подання	Надання послуг, відображення даних, обробка подій інтерфейсу користувача, обслуговування запитів HTTP, підтримка функцій командного рядка та API пакетного виконання
Домен	Бізнес-логіка програми
Джерело даних	Звернення до бази даних, обмін повідомленнями, управління транзакціями тощо.

Створювати сайт було вирішено з використанням паттерна трирівневої архітектури MVC [3]. Концепція патерну (шаблону) MVC (model – view - controller) передбачає поділ даних програми, інтерфейсу користувача і керуючої логіки на три окремих компоненти (рис. 2.2):

- Модель;
- Подання;
- Контролер.

Модель (Model) представляє дані предметної області уявленню і реагує команди контролера, змінюючи свій стан.

Подання (View) відповідає за відображення даних на предметній області (моделі) користувачеві, реагуючи на зміни моделі. У випадку Web програми, представлення – це HTML сторінка, яку бачить користувач.

Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель необхідність змін [4].

Реалізувати ці компоненти зручно, використовуючи класи ООП (об'єктно орієнтованого програмування).

Основна перевага цього патерну – незалежність компонентів. Вони можуть модифікуватися, віддалятися, змінюватись, не торкаючись один одного. Незалежно від змін, система в цілому залишатиметься робочою.

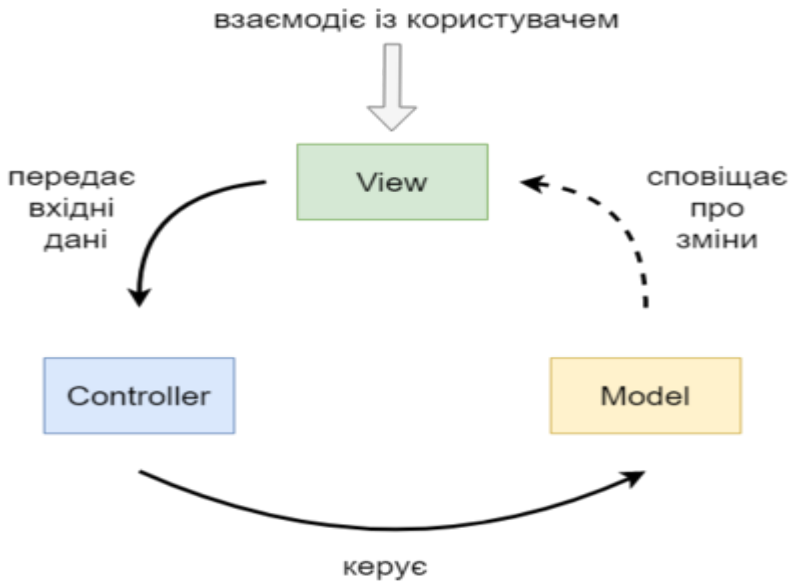

Рисунок 2.2 – Схема взаємодії компонентів MVC

Завдяки цьому, додаток набуває найважливіших в умовах технологій, що стрімко розвиваються, властивості – масштабованість і гнучкість. Наприклад, з'являється необхідність нових, раніше не використовуваних, даних предметної області – тоді ми просто додаємо нову модель, що описує їх, контролер, що передає запити на відображення цих даних і, можливо, додаємо у подання елемент, ці дані надає користувачеві. І ось нові дані вже в нашому додатку, причому вся колишня структура та функції не порушені, вони працюють, як і раніше.

Більшість програмістів спеціалізуються у певній вузькій області: або займаються розробкою графічних інтерфейсів, або розробляють бізнес-логіку програми. Використовуючи підхід MVC, можна домогтися поділу праці і тим самим збільшити ефективність і швидкість розробки.

2.3 Вибір мов програмування

2.3.1 Мова серверних сценаріїв

Мовою програмування серверної частини програми було обрано PHP 7.3 [5]. Насамперед через колосальну популярність у співтоваристві програмістів та відкритість вихідних кодів.

PHP є одним із лідерів серед мов, що застосовуються для створення динамічних веб-сайтів (рис. 2.3).

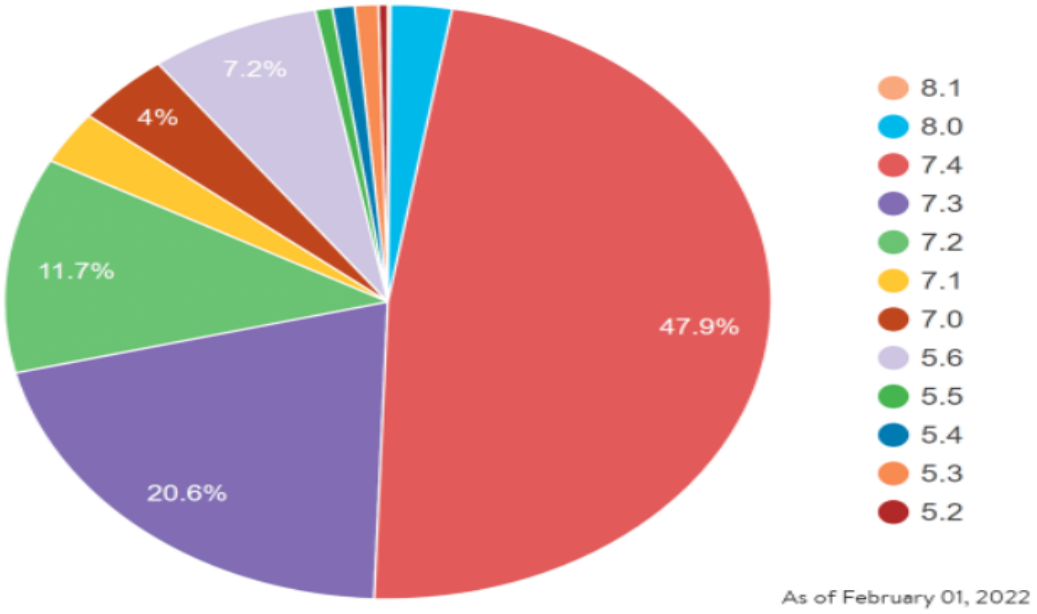

Рисунок 2.3 – Діаграма використання версій PHP

PHP («PHP: Hypertext Preprocessor») – широко використовувана мова сценаріїв загального призначення з відкритим вихідним кодом, яка особливо підходить для веб-розробки та може бути вбудована в HTML. Його синтаксис спирається на C, Java та Perl і простий у освоєнні. Мова підтримує концепцію об'єктно орієнтованого підходу (ООП). Основна мета мови – дозволити веб-розробникам швидко створювати динамічні веб-сторінки [6]. На даний момент 78.1% сайтів, мова платформи яких відома, використовують PHP як основну мову (рис. 2.4) [7].

Версія PHP 7 ґрунтується на експериментальній галузі PHP, яка спочатку називалася phpng (PHP Next Generation – наступне покоління), і розроблялася з упором на збільшення продуктивності та зменшення споживання пам'яті [5]. У версії додано можливість вказувати тип даних, що повертаються з функції, доданий контроль переданих типів для скалярних даних, а також нові оператори.

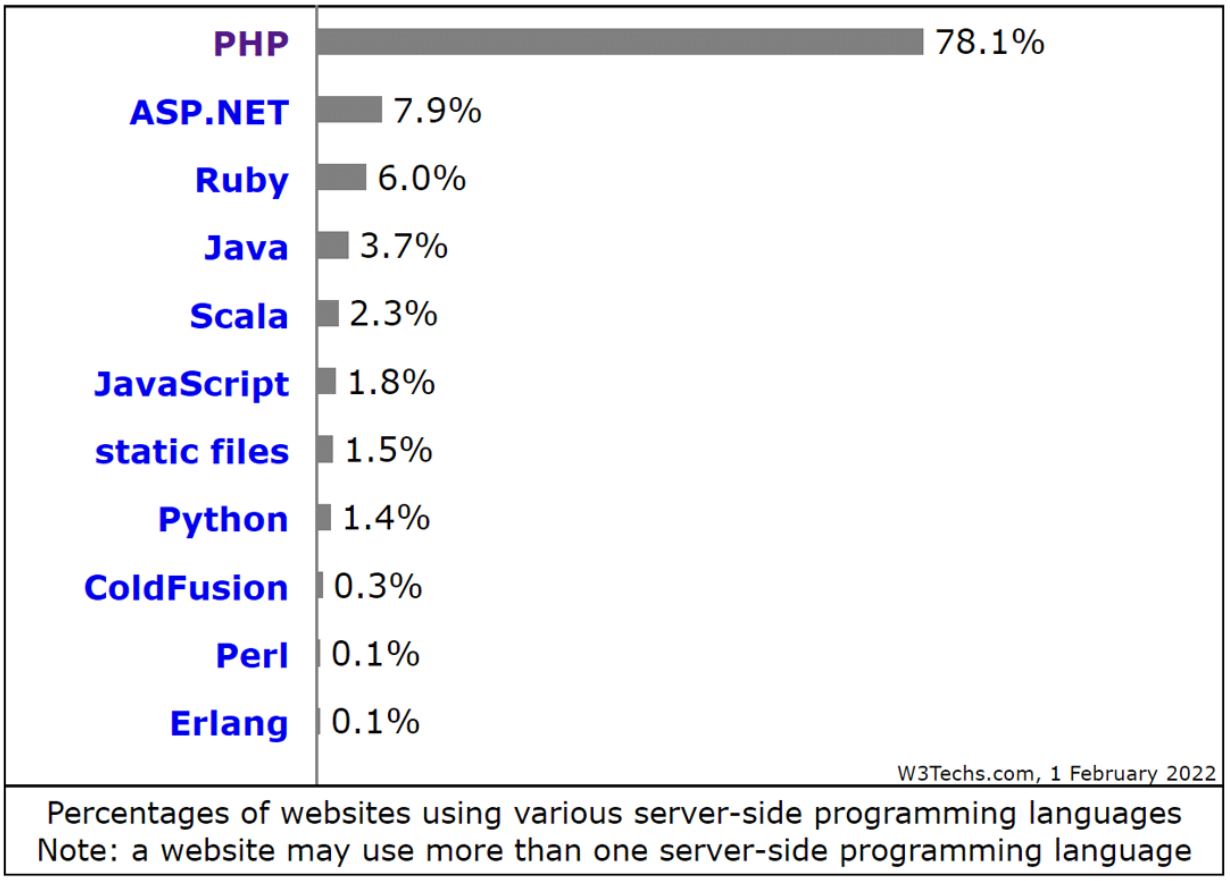

Рисунок 2.4 – Діаграма використання мов програмування для веб-сайтів

PHP підтримує доступ майже до будь-яких систем управління базами даних, таким як: MySQL, SQLite, PostgreSQL, Oracle, Microsoft SQL та ін. Саме з базою даних MySQL [8], нам і належить працювати.

2.3.2 Мова інтерфейсу користувача

Для представлення структури сторінки використовувався HTML5 [9] – тегова мова розмітки документів, що є набором елементів розмітки. Відмінність HTML5 від попередніх версій мови полягає у введенні нових тегів, що визначають поточну специфікацію елементів сторінки. А для опису властивостей елементів розмітки використовувався CSS3[10,11] – формальна мова опису зовнішнього вигляду документа, зокрема, CSS застосовується для завдання кольору заливки елементів, розмірів, розташування на сайті тощо. Основною метою розробки CSS є розділення опису структури веб сторінки (HTML) від опису зовнішнього вигляду цієї веб-сторінки. Такий поділ збільшує доступність документа, надає велику гнучкість та можливість управління його поданням, а також зменшити складність та повторюваність коду [12].

2.4 Вибір фреймворку

2.4.1 Навіщо використовувати фреймворк?

Незважаючи на те, що PHP в чистому вигляді можна використовувати для створення практично будь-якої програми, функціональність сайтів, що все зростає, вимагає структурування та організації самого процесу розробки. Найбільш природним рішенням цього завдання стали PHP-фреймворки.

Навіщо використовувати фреймворк замість чистого PHP для розробки програми? Декілька переваг використання фреймворків викладено нижче.

- PHP фреймворк значно скорочує терміни розробки. Не потрібно прописувати вручну підключення до баз даних, ajax-запити тощо.
- Спочатку проект строго структурований, тому ймовірність припуститися помилок в архітектурних рішеннях вкрай низька.
- Код добре документований та придатний для повторного використання.
- Фреймворки дають змогу легко масштабувати, розширювати проекти.

- Код стає лаконічним і зрозумілішим, що спрощує роботу. Особливо актуально, якщо технологія ведеться командою програмістів.
- Веб-програми більш безпечні, якщо написані за допомогою фреймворку. Адже розробнику не доводиться турбуватися про низькорівневу безпеку сайту.
- Забезпечує дотримання шаблону. У разі шаблону MVC (Model View-Controller).
- Сприяє застосуванню парадигми ООП (об'єктно-орієнтованого програмування).

2.4.2 Огляд популярних фреймворків: Laravel, Yii, Symfony.

Laravel (рис. 2.5) [17] відносно молодий фреймворк, перший реліз з'явився лише у 2011 році. Але за результатами опитування порталу SitePoint [18] він є найпопулярнішим серед веб-розробників.

Переваги фреймворку :

- Величезна екосистема, що виросла останніми роками вдвічі.
- Наявність простої та змістовної документації, навчальних ресурсів, форумів тощо.
- Наявність російськомовної спільноти, де вся документація представлена російською [18].
- Елегантний синтаксис, що спрощує такі завдання, як авторизація, керування кешуванням, чергами, сесіями.
- Підтримує Composer для керування пакетами.
- Підтримка власного шаблонизатора Blade. На відміну від інших шаблонних систем, Blade дозволяє використовувати PHP код уявленнях (view), тобто. прямо всередині HTML-розмітки. Код всередині HTML файлу перетворюється на чистий PHP на етапі обробки.
- Підтримка REST-Full маршрутизації.
- Добре підтримується модульне тестування.
- Існує безліч пакетів для розширення функціональності фреймворку.

- Багато хостинг компанії надають підтримку Laravel та пропонують хостинг рішення для додатків на Laravel.

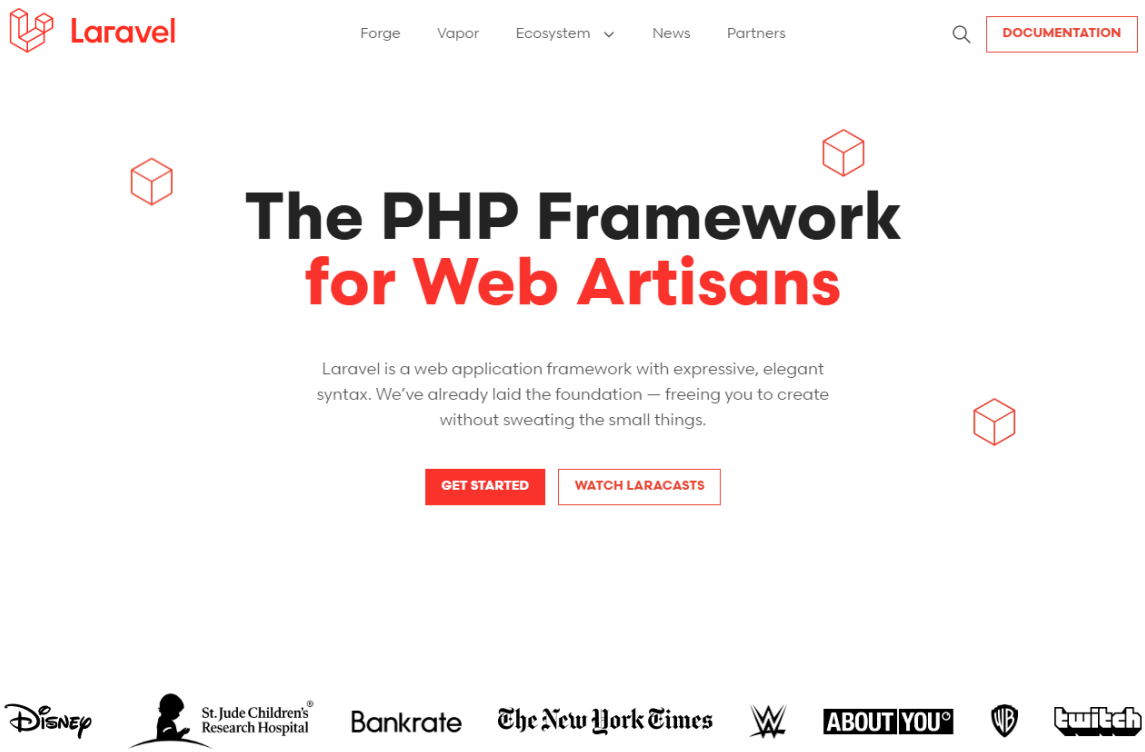

Рисунок 2.5 – Офіційний сайт Laravel

Недоліки фреймворку.

- Laravel програє іншим фреймворкам у продуктивності (кількість запитів, що обробляються фреймворком, за секунду). Порівняння продуктивності фреймворків для версії мови PHP 7.1 представлено. (рис. 2.6)
- Займає багато пам'яті порівняно з іншими фреймворками. (рис. 2.7).

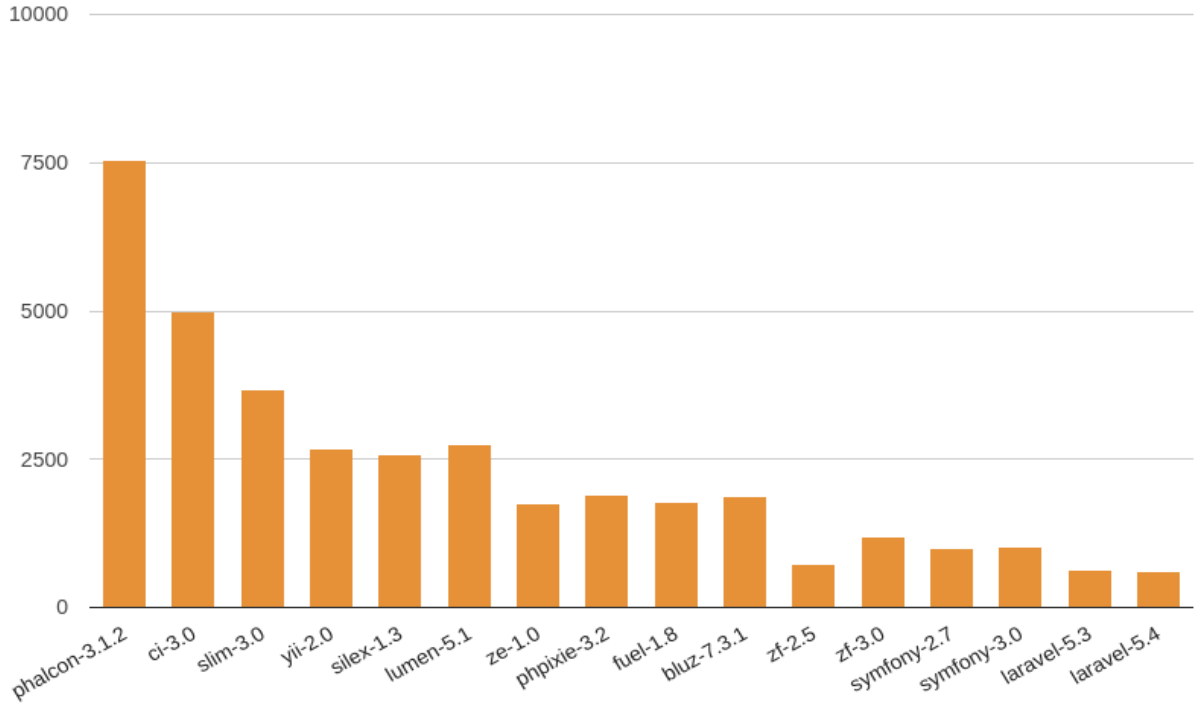

Рисунок 2.6 – Продуктивність фреймворків (запитів за секунду)

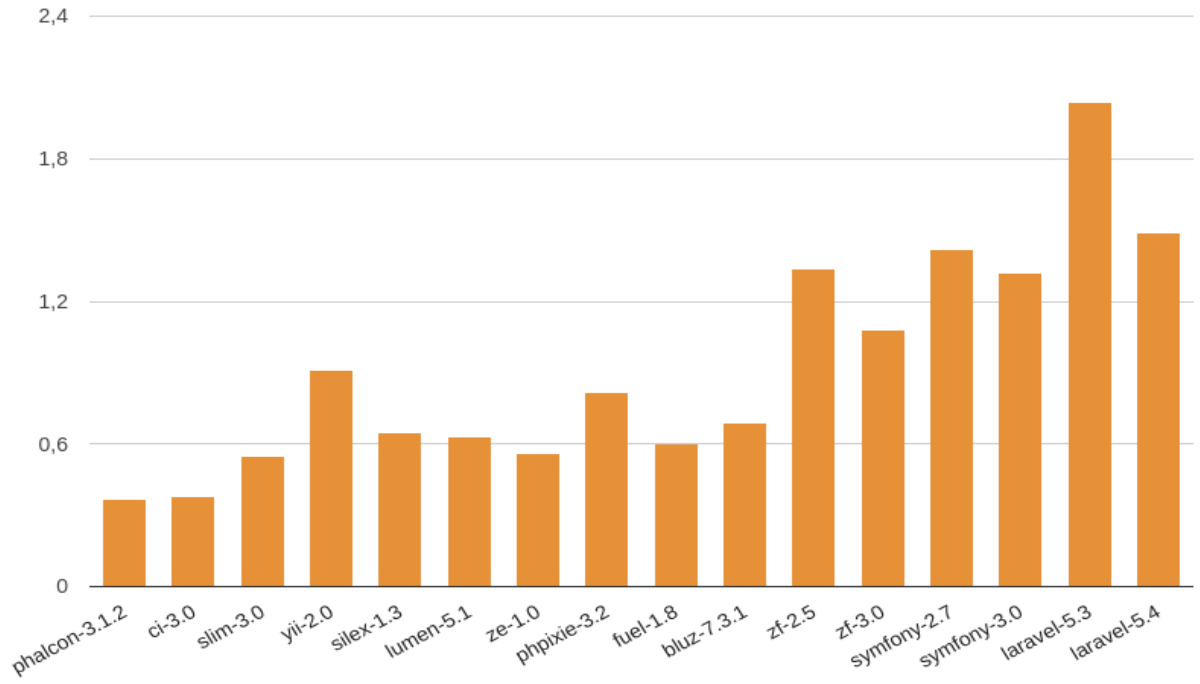

Рисунок 2.7 – Пам'ять, що займається фреймворками (Мб)

Незважаючи на те, що реліз третьої версії відбувся ще у 2015 році, саме друга версія Symfony (рис. 2.8) одноосібно утримує 3-е місце за популярністю серед фреймворків. Причина тут очевидна – швидкість роботи та загальна простота. Але щоб це не йшло у розріз із функціональністю, користувачеві пропонується вибрати одну з 3 версій для профільної роботи.

- Standard Edition – для знайомства та виконання спільних завдань. На ній заснований дистрибутив Hello World Edition, який містить одно скрипт оптимізації для подальшого використання в бенчмарках.
- Symfony CMF – адаптація для розробників, які працюють із CMS системами.
- REST Edition – оптимізація для роботи з REST-архітектурою (інтернет магазини, пошукові системи тощо).

Стереотипно вважається, що Symfony – це фреймворк для любителів командного рядка. Дійсно, вбудований інтерфейс SensioGeneratorBundle допоможе вам з одного рядка тексту отримати цілий скелет для коду. Переваги.

- Висока продуктивність завдяки кешування байт-коду. Надійність. В даний час є найбільш стабільним. Маса функцій розширення.
- Велика спільнота з документацією, навчальними ресурсами.
- Гарна підтримка; повністю сформований фреймворк.
- Система шаблонів Symfony Twig

Рисунок 2.8 – Логотип фреймворку Symfony

Недоліки.

- Надає неповну реалізацію моделі MVC, а лише модель і контролер.
- Досить складний у освоєнні.

Yii (рис. 2.9) є високопродуктивним, безпечним та швидким фреймворком для розробки додатків та веб-сайтів. Yii використовує менеджер PHP-залежностей для обробки різних залежностей і установок.

Рисунок 2.9 – Логотип фреймворку Yii

Переваги.

- Відмінно підходить для розробки програм у режимі реального часу Займає мінімальну кількість пам'яті.
- Швидко справляється з оптимізацією для маршрутизації URL-адрес, кеш-моделями, кодом.
- Добре підходить для багатомовних програм.
- Простий в установці.
- Легко налаштовується для потрібних завдань. Майже всі компоненти фреймворку розширюються.
- Система шаблонів Yii Default.
- Має велику спільноту з великою кількістю навчальних ресурсів.

Недоліки.

- Допускає повторення коду, на відміну інших MVC-фреймворків, у яких цю проблему вирішено.
- Сильна пов'язаність класів. Все в системі успадковується від CComponent.
- Інтеграція шаблонизатора (Twig, Smarty) досить слабка порівняно з нативними шаблонами.

2.4.3 Висновки щодо огляду фреймворків

Подібності.

- Всі три фреймворки є full-stack PHP фреймворками, що дозволяє створювати повністю готові веб-програми, починаючи від інтерфейсу (front-end) і закінчуючи серверними сценаріями (back-end).
- Є доступ до проектів із відкритим вихідним кодом. Їхній вихідний код можна знайти на GitHub.
- Фреймворки добре документовані та підтримуються великими спільнотами.
- Кожен із них підтримує ORM (Object Relationship Mapping). ORM – це «технологія програмування, яка пов'язує бази даних із концепціями об'єктно-орієнтованих мов програмування, створюючи віртуальну об'єктну базу даних» [20]. Наприклад, у Laravel підтримку ORM забезпечують "міграції".
- Всі три фреймворки використовують шаблонні для покращення front-end розробки та технічного обслуговування.
- Вони є надійними та безпечними для створення веб-додатків 2.0 [12].

Підтримка бази даних.

Кожен фреймворк оснащений підтримкою різних баз даних. Symfony 2 може працювати з масивом баз даних, таким як NoSQL та DynamoDB. Тому в даному аспекті є більш універсальним

Yii та Laravel також корисні, але забезпечують меншу сумісність. Наведена на рис. 2.10 таблиця показує, які бази даних підтримуються тим чи іншим фреймворком.

Framework	Laravel	Yii	Symfony 2
Database	SQLite MySQL PostgreSQL Redis Microsoft BI MongoDB	MySQL SQLite Microsoft BI Oracle PostgreSQL MongoDB	Microsoft BI MongoDB MySQL NoSQL PostgreSQL CouchDB DynamoDB GemFire GraphDB MemBase MemCacheDB Oracle Apache Jackrabbit

Рисунок 2.10 – Сумісність фреймворків із базами даних

Швидка розробка.

Для будь-якої компанії або клієнта важливо якомога швидше вивести додаток на ринок для задоволення споживчого попиту. Laravel швидко набирає обертів, закріплюючись у трійці основних PHP фреймворків. Крім того, Laravel є ідеальним для розробників-початківців. Він простий для розуміння – легко знайти документацію, навчальні уроки та відео, готові програми для прикладу в Інтернеті.

У свою чергу Yii піднімає продуктивність на новий рівень.

Ефективність.

Продуктивність будь-якої програми важлива, коли вона використовує критично важливі дані у режимі реального часу. Багато додатків мають

високу продуктивність і пред'являють високі вимоги до швидкості роботи. Багато проектах фреймворки грають вирішальну роль.

Незважаючи на те, що Laravel є найповільнішим фреймворком, існує безліч способів і ресурсів для прискорення Laravel додатків.

Це робить Laravel найбільш гнучким і дозволяє використовувати його в додатках з динамічним навантаженням.

2.4.4 Фреймворк Laravel 8

Для реалізації програми було обрано фреймворк Laravel, т.к. він абсолютно безкоштовний, має відкритий вихідний код і спочатку призначений для розробки з використанням архітектурної моделі Model View Controller.

Також це один із найпопулярніших фреймворків для розробки на PHP (див. рис. 2.11).

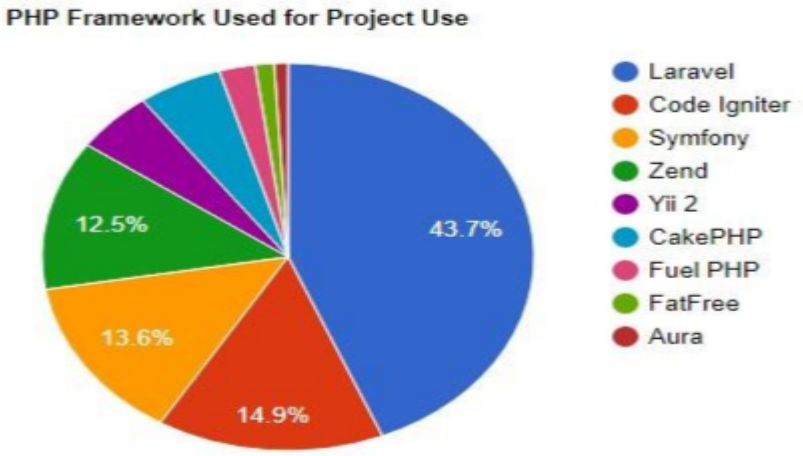

Рисунок 2.11 – Діаграма популярності фреймворків для PHP

Розробка на Laravel 8 чудово підходить для створення:

- CRM та ERP систем;
- корпоративних порталів;
- інтернет-магазинів;
- систем онлайн-бронювання.

Laravel є досить гнучким фреймворком і дозволяє вирішувати нестандартні завдання, структурувати веб-сайт відповідно до існуючої логіки та поставлених цілей.

Додаткові переваги Laravel.

- Великий функціонал.

Створення сайтів будь-якого рівня можливе завдяки величезній функціональності. Використовуючи цей фреймворк, можна реалізувати проекти, що надають можливість інтеграції необхідного функціоналу відповідно до індивідуальних вимог та особливостей конкретного бізнесу.

- Можливість створити гнучку адмін панель.

Можна реалізувати найзручніший варіант управління ресурсом, створюючи індивідуальну панель адміністратора під завдання конкретного веб-проекту.

- Безпека бази даних.

Можливість отримати несанкціонований доступ до бази даних, створеної з використанням Laravel, дуже складна. Високий рівень безпеки гарантує надійний захист від SQL-injection, атак типу XSS, CSRF.

- Регулярні випуски.

Вихідний код змінюється з урахуванням нововведень у PHP та потреб програмістів. Свіжі оновлення допомагають усунути проблеми, що раніше існували, і зробити фреймворк ще зручнішим.

- Популярність та активна спільнота.

Наявність великої спільноти відкриває простір для динамічної комунікації, обміну особистим досвідом та думками, вирішення різноманітних питань пов'язаних із проектуванням та підтримкою інтернет-ресурсів.

- Масштабованість.

Розробка на Laravel 8 передбачає можливість розширення функціоналу (інтеграції додаткових модулів) без істотних витрат на зміну поточної системи, а також ризик виникнення небажаних втрат для веб ресурсу.

2.5 Висновки у розділі

У розділах 2.1 – 2.4 визначено архітектуру та засоби розробки додатка. Прийнято рішення розробляти інформаційну систему «Рада молодих вчених» як клієнт-серверний додаток (web-сайт) з трирівневою архітектурою на основі моделі MVC та використовувати PHP-фреймворк Laravel. Використання фреймворку забезпечить суворе слідування моделі MVC, тобто гнучкість і масштабованість програми, а також низькорівневу безпеку програми.

3 РОЗРОБКА САЙТУ

3.1 Розробка бази даних

3.1.1 Концептуальне проектування

Для проектування бази даних необхідно визначити які саме дані нам потрібно там зберігати. У нашого сайту буде публічна сторона, та адмін панель. Публічна сторона вимагає створення шести сутностей, розглянемо їх детальніше.

Склад РМВ - це люди які входять до РМВ мають такі поля:

- ім’я;
- опис;
- пошта;
- номер телефону;
- фото;
- тип (посада в Раді);
- поле sort (буде необхідно для коректного сортування);
- поле active (буде необхідно для включення/виключення відображення);
- зовнішній ключ відділу;
- зовнішній ключ факультету.

Відділи:

- назва;
- опис;
- короткий опис;
- фото;
- поле sort;
- поле active;

Факультети:

- назва;
- опис;

- короткий опис;
- фото;
- поле sort;
- поле active.

Новини:

- назва;
- заголовок;
- опис;
- короткий опис;
- фото;
- поле active;
- зовнішній ключ наступної новини;
- зовнішній ключ попередньої новини.

Документи:

- назва;
- документ.

Контент:

- тип;
- назва;
- фото;
- додаткове фото;
- опис.

Отже на публічній частині сайту маємо шість сутностей: людина, відділ, факультет, новина, документ, контент.

Всі ці сутності будуть мати CRUD (create - створення, read - читання, update - оновлення, delete - видалення) у адмін панелі про яку майже не буде йти річ у моїй дипломній роботі. Лише поверхневий огляд, та невелика частина розповіді буде про архітектуру адмін панелі.

3.1.2 Логічне проектування

На етапі логічного проектування кожну сутність потрібно уявити в вигляді таблиці. Було вирішено, що всі сутності будуть мати локалізовану модель, для нашої БД - це означає створення додаткової lang таблиці. Це потрібно для того, щоб у разі необхідності додавання до сайту нових мов та перекладів наших сутностей це можно було зробити з мінімальними , або взагалі без змін в архітектурі нашої БД.

Реляційна схема бази даних наведено рисунку 3.1.

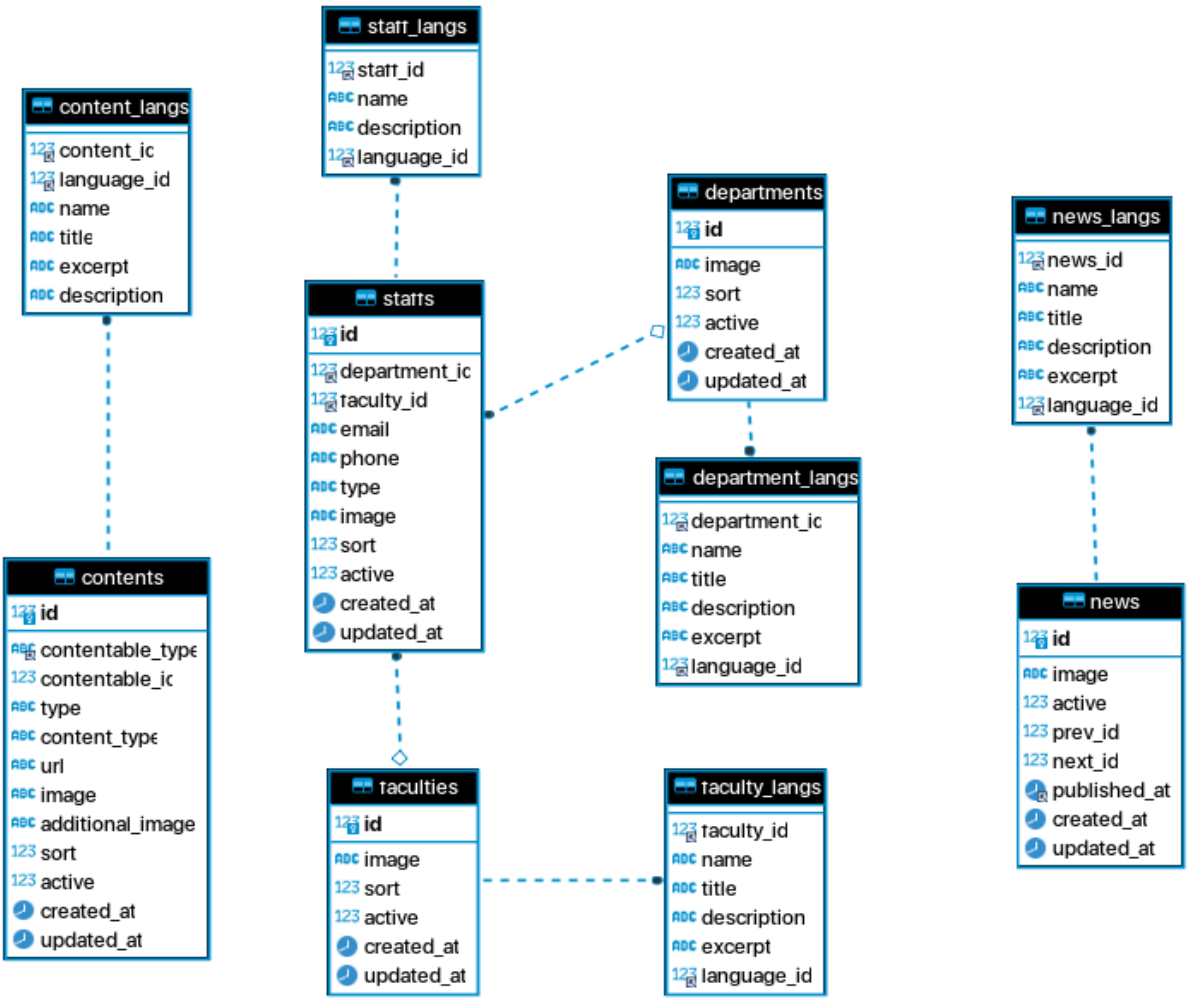

Рисунок 3.1 - Схема реляційної БД

Опис призначення та властивостей полів, а також взаємозв'язків таблиць бази даних наведено далі в таблицях 3.1-3.11.

Таблиця 3.1 - «staffs» - люди

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
id	int	-	-	+	-
department_id	int	-	+	-	+
faculty_id	int	-	+	-	+
email	varchar(255)	-	+	-	-
phone	varchar(255)	-	+	-	-
type	varchar(255)	-	+	-	-
sort	int	0	-	-	-
active	boolean	1	-	-	-
created_at	datetime	-	+	-	-
updated_at	datetime	-	+	-	-

Ця таблиця зберігає в собі людей які працюють в раді. Кожна людина може мати id відділу та/або факультету, поле email, поле phone, поле type (використовується для позначення чи є людина головою відділу), поле sort та active (для сортування). Нижче наведена її локалізована таблиця 3.2.

Таблиця 3.2 - «staff_langs» - локалізована таблиця людей

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
staff_id	int	-	-	-	+
name	varchar(255)	-	+	-	-
description	varchar(255)	-	+	-	-
language_id	int	-	-	-	+

Таблиця 3.3 - «departments» - таблиця відділів

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
id	int	-	-	+	-
image	varchar(255)	-	+	-	-
sort	int	0	-	-	-
active	boolean	1	-	-	-
created_at	datetime	-	+	-	-
updated_at	datetime	-	+	-	-

Ця таблиця зберігає в собі відділи РМВ. Кожен відділ може мати поле image, поле sort та active (для сортування). Нижче наведена її локалізована таблиця 3.4.

Таблиця 3.4 - «department_langs» - локалізована таблиця відділів

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
department_id	int	-	-	-	+
name	varchar(255)	-	+	-	-
description	varchar(255)	-	+	-	-
excerpt	varchar(255)	-	+	-	-
title	varchar(255)	-	+	-	-
language_id	int	-	-	-	+

Таблиця 3.5 - «faculties» - таблиця факультетів

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
id	int	-	-	+	-
image	varchar(255)	-	+	-	-
sort	int	0	-	-	-
active	boolean	1	-	-	-
created_at	datetime	-	+	-	-
updated_at	datetime	-	+	-	-

Ця таблиця зберігає в собі факультети. Кожен факультет може мати поле image, поле sort та active (для сортування).Нижче наведена її локалізована таблиця 3.6.

Таблиця 3.6 - «faculty_langs» - локалізована таблиця факультетів

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
faculty_id	int	-	-	-	+
name	varchar(255)	-	+	-	-
description	varchar(255)	-	+	-	-
excerpt	varchar(255)	-	+	-	-
title	varchar(255)	-	+	-	-
language_id	int	-	-	-	+

Таблиця 3.7 - «news» - таблиця новин

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
id	int	-	-	+	-
image	varchar(255)	-	+	-	-
prev_id	int	-	+	-	-
next_id	int	-	+	-	-
published_at	datetime	-	+	-	-
created_at	datetime	-	+	-	-
updated_at	datetime	-	+	-	-

Ця таблиця зберігає в собі новини РМВ. Кожна новина може мати prev_id та next_id (це ідентифікатори попередньої/наступної новини), поле image, поле published_at (дата публікації новини). Нижче наведена її локалізована таблиця 3.8.

Таблиця 3.8 - «news_langs» - локалізована таблиця новин

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
news_id	int	-	-	-	+
name	varchar(255)	-	+	-	-
description	varchar(255)	-	+	-	-
excerpt	varchar(255)	-	+	-	-
title	varchar(255)	-	+	-	-
language_id	int	-	-	-	+

Таблиця 3.9 - «contents» - таблиця контенту

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
id	int	-	-	+	-
contentable_id	int	-	+	-	+
contentable_type	varchar(255)	-	+	-	-
type	varchar(255)	-	+	-	-
content_type	varchar(255)	-	+	-	-
url	varchar(255)	-	+	-	-
image	varchar(255)	-	+	-	-

Таблиця 3.10 - «content_langs» - локалізована таблиця контенту

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
content_id	int	-	-	-	+
name	varchar(255)	-	+	-	-
description	varchar(255)	-	+	-	-
excerpt	varchar(255)	-	+	-	-
title	varchar(255)	-	+	-	-
language_id	int	-	-	-	+

Таблиця contents була спроектована для легкої можливості створення та редагування сутностей з однаковими полями. Це може знадобитися у майбутньому для розширення сайту.

Таблиця 3.11 - «documents» - таблиця документів

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
id	int	-	-	+	-
name	int	-	+	-	+
document	varchar(255)	-	+	-	-

3.1.3 Фізичне проектування. Міграції у Laravel.

Фреймворк Laravel надає можливість використання міграцій.

Міграції – це своєрідний механізм контролю версій бази даних, він дозволяє команді програмістів легко змінювати та спільно використовувати схему бази даних програми.

За допомогою міграцій здійснюється доступ до бази даних прямо з без використання інтерфейсів СУБД.

Кожна міграція реалізується як клас і містить у собі 2 методи – up() та down(). Метод up() спрацьовує під час запуску міграції, а метод down() при відкаті.

У методі up() ми можемо визначити тип полів таблиці, властивості цих полів та навіть первинні та зовнішні ключі, що пов'язують створену таблицю коїться з іншими.

Для всіх міграцій для зручності написання, я використовую трейт MigrationCreateFieldTypes приклад методів якого можна побачити на рисунку 3.2.

```
trait MigrationCreateFieldTypes
{
    /**
     * @var Blueprint
     */
    protected $tableBlueprint;

    public function setTable(Blueprint $table): self
    {
        $this->tableBlueprint = $table;

        return $this;
    }

    public function createActive(): self
    {
        $this->table()->boolean( column: 'active')->default( value: true);

        return $this;
    }
```

Рисунок 3.2 - трейт MigrationCreateFieldTypes

Наприклад, повний текст міграції для створення таблиці «staffs» наведено на рисунках 3.3 - 3.4. Повний текст усіх міграцій міститься у додатку А.

```
public function up(): void
{
    Schema::create($this->table, function (Blueprint $table) {
        $this->builder->setTable($table);

        $table->id();
        $table->unsignedBigInteger( column: 'department_id')->nullable();
        $table->unsignedBigInteger( column: 'faculty_id')->nullable();
        $this->builder
            ->createNullableChar( column: 'email')
            ->createNullableChar( column: 'phone')
            ->createNullableChar( column: 'type')
            ->createImage()
            ->createSort()
            ->createActive()
        ;
        $table->timestamps();

        $this->builder->addForeign( localOwnerKey: 'department_id', ownerTable: 'departments');
        $this->builder->addForeign( localOwnerKey: 'faculty_id', ownerTable: 'faculties');
    });


    Schema::create($this->tableLang, function (Blueprint $table) {
        $this->builder->setTable($table);
        $table->unsignedBigInteger($this->foreignKey);

        $this->builder
            ->createName()
            ->createDescription()
            ->createLanguageKey()
        ;
        $table->foreign($this->foreignKey)
            ->references( columns: 'id')->on($this->table)
            ->onUpdate( action: 'cascade')->onDelete( action: 'cascade');
    });
}
```

Рисунок 3.3 - up метод міграції для створення таблиці «staffs»

```
public function down(): void
{
    Schema::dropIfExists($this->tableLang);
    Schema::dropIfExists($this->table);
}
```

Рисунок 3.4 - down метод міграції для створення таблиці «staffs»

Для використання переваг механізму міграцій необхідно створення кожній таблиці БД 2-х додаткових полів: «created_at» та «updated_at». Ці поля містять інформацію про те, коли було створено запис таблиці, і коливона була востаннє змінена.

Створення таблиці "migrations" дозволити стежити за міграціями (див. рисунок 3.5 та табл. 3.12)

Рисунок 3.5 - таблиця в середі MySQL «migrations»

Таблиця 3.12 - «migrations» - таблиця міграцій

	Тип даних	Значення за замовчуванням	Може бути NULL	Первинний ключ	Зовнішній ключ
id	int	-	-	+	-
migration	varchar(255)	-	-	-	-
batch	int	-	-	-	-

3.2 Розробка архітектури сайту

У розділі 2.1 було прийнято рішення реалізовувати програму, як трирівневу клієнт-серверну систему на основі шаблону MVC (Model-View-Controller). На рисунку 3.6 зображено діаграму розміщення нашого програми.

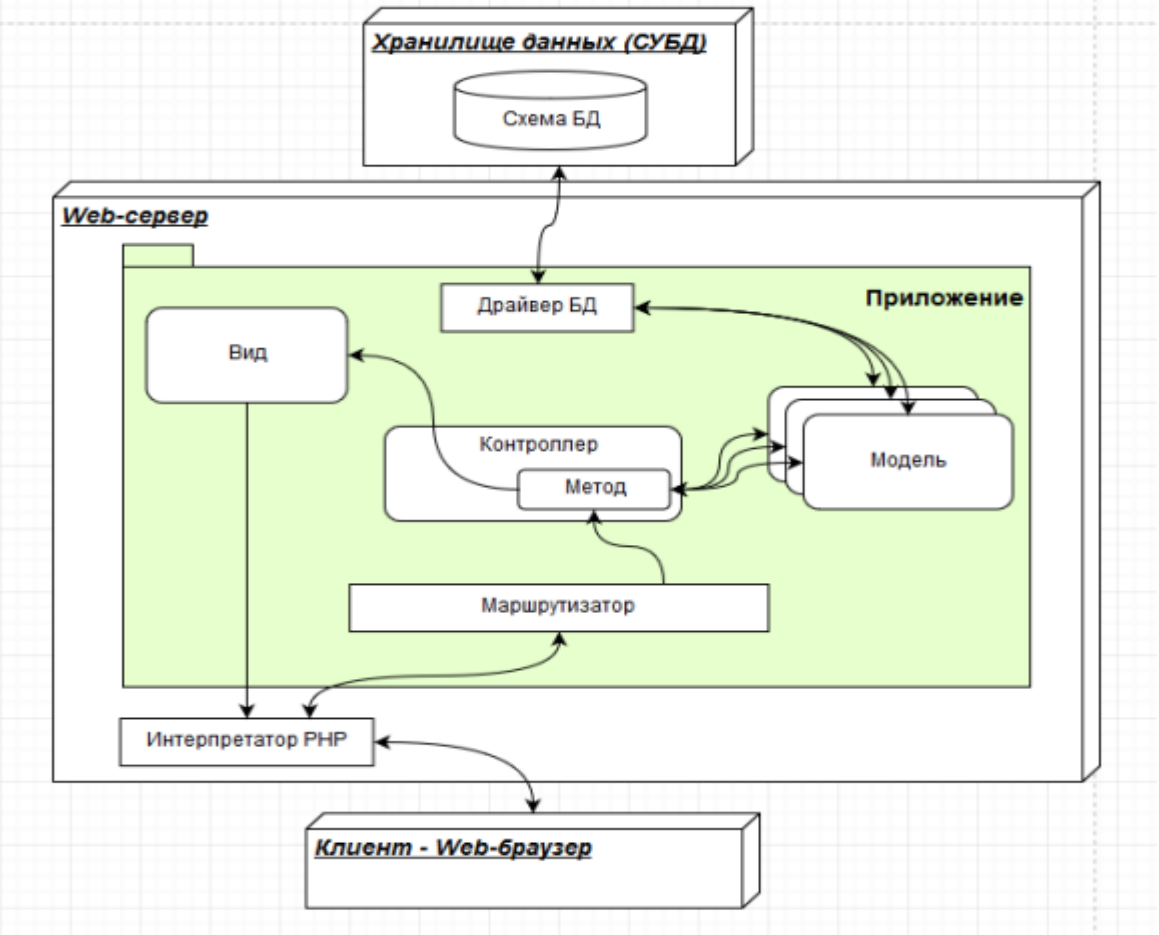

Рисунок 3.6 - Діаграма розміщення (з реалізацією моделі MVC)

На діаграмі позначено 3 рівні клієнт-серверної системи: клієнт (браузер, через який користувач передивляє веб-сайти), сервер (на якому розміщено саму програму) і сервер бази даних, що використовується додатком.

У блоці «Web-сервер» описано механізм взаємодії компонентів програми згідно з архітектурним шаблоном MVC. Розглянемо цей механізм докладніше. Клієнт надсилає запит (наприклад, друкує в адресному рядку якийсь URL-адрес), цей запит передається маршрутизатору. Маршрутизатор (router) це клас, частина коду, яка визначає, куди (якому контролеру чи якому

методу контроллера) передати керування при отриманні того чи іншого запиту. Згідно з прописаними правилами маршрутизації, роутер передає управління методом контролера. Усередині методу описано алгоритм, що

здійснює взаємодія із моделями. Моделі, відповідно до шаблону MVC, відповідають таблиць-сутностей бази даних. На діаграмі зображено взаємодію моделей із базою даних через драйвер. Драйвер БД - частина коду, що реалізує підключення та взаємодію з базою даних. Алгоритм, описаний усередині методу може додавати, змінювати, видаляти моделі, та здійснювати будь-які інші дії, передбачені логікою програми. Зазвичай метод повертає уявлення, яке слід передати користувачеві після завершення алгоритму.

Визначивши, як користувач взаємодіє із системою, можна перейти до більш докладний опис програми. Наприклад, визначити його компоненти. Компонент – фізична частина системи. Компоненти можуть включати реалізацію класу, реалізацію функціональних можливостей системи та ін [16].

Для роботи сайту знадобляться файли моделей, контролерів, роут файли, репозиторії та уявлення.

Моделі (головна та її локалізована модель):

- Staff.php, StaffLang.php
- Department.php, DepartmentLang.php
- Faculty.php, FacultyLang.php
- Staff.php, StaffLang.php
- Document.php

Файл з роутами для маршрутизації публічних сторінок:

- web.php

Файл з роутами для маршрутизації сторінок адмін панелі:

- web_admin.php

Публічні контролери:

- HomeController.php
- NewsController.php
- FacultyController.php
- DepartmentController.php
- DocumentController.php

Адмінські контроллери:

- NewsController.php
- FacultyController.php
- DepartmentController.php
- DocumentController.php

Репозиторії (використовуються для відокремлення методів для роботи з БД від моделей):

- DocumentRepository.php
- DepartmentRepository.php
- FacultyRepository.php
- NewsRepository.php
- StaffRepository.php

Уявлення :

- (про уявлення буде описано у розділі 3.3)

Весь вихідний код цих файлів буде представлений у додатку Б.

3.3 Розробка front-end частини

Інтерфейс публічної частини сайту розроблявся з допомогою Blade шаблонізатора. Blade - це простий, але потужний двигун шаблонів, що входить до складу Laravel. На відміну від деяких шаблонізаторів PHP, Blade не обмежує вас у використанні звичайного "сирого" коду PHP в шаблонах. Для створення оформлення інтерфейсу було використано чистий CSS та JavaScript.

Дякуючи Blade я зробив головний шаблон публічної сторінки app.blade.php (див. рис. 3.7).

```
<!DOCTYPE html>
<html lang="ru">
@include('public.layout.includes.head')
<body>

@includeIf('public.layout.includes.preloader')

@hasSection('content')
    @yield('content')
@else
    {!! $content ?? '' !!}
@endif

@include('public.layout.includes.assets.scripts')
</body>
</html>
```

Рисунок 3.7 - Головний шаблон app.blade.php

Від головного шаблону я успадкував всі наступні шаблони публічних сторінок додатку. В мене вийшла елегантна структура з html шаблонів Blade (рис. 3.8).

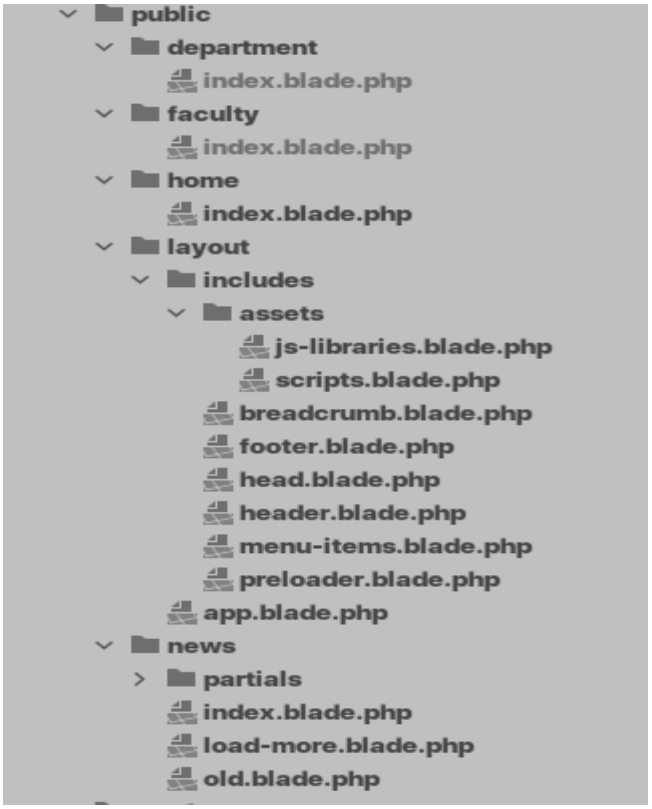

Рисунок 3.8 - Структура з html шаблонів Blade

Додавши в свій html класи css та JavaScript, в мене вийшов такий зовнішній вигляд сайту дивіться на рисунках 3.9-3.12.

Вихідний код css та JavaScript можна побачити у додатку В.

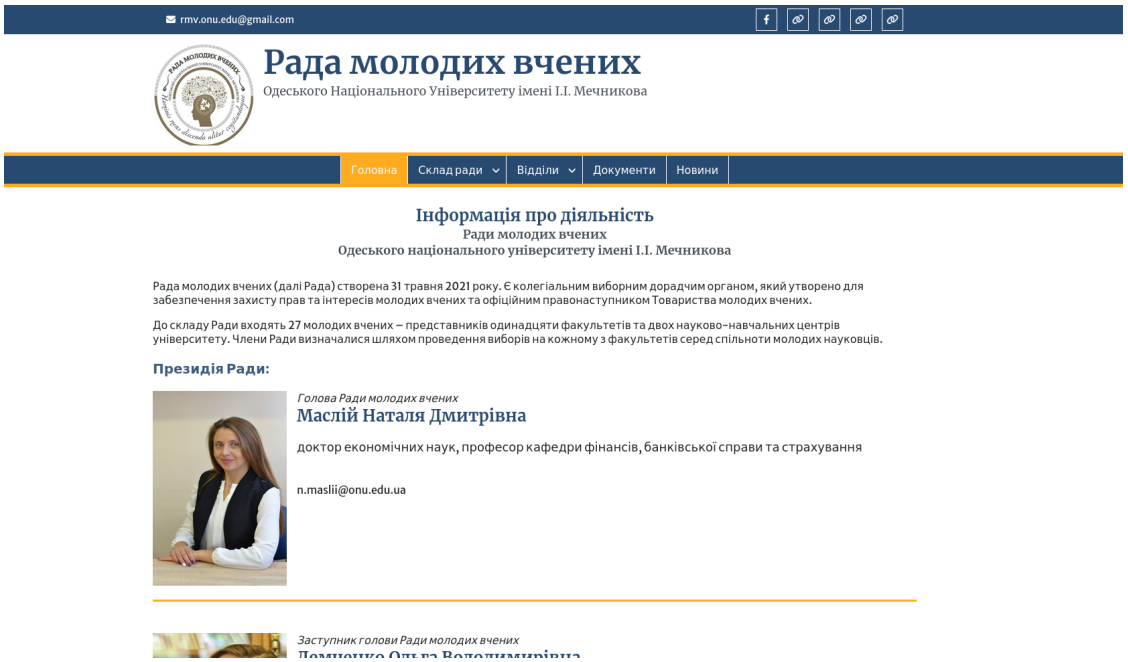

Рисунок 3.9 - Головна сторінка

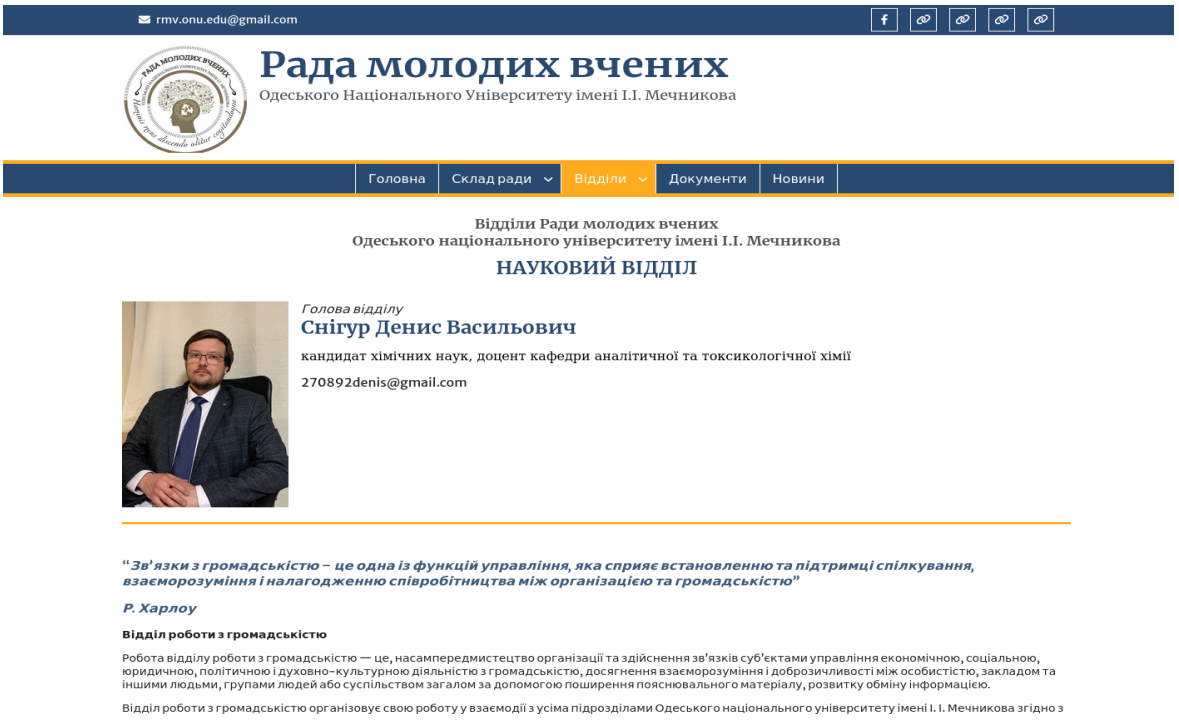

Рисунок 3.10 - Сторінка відділу

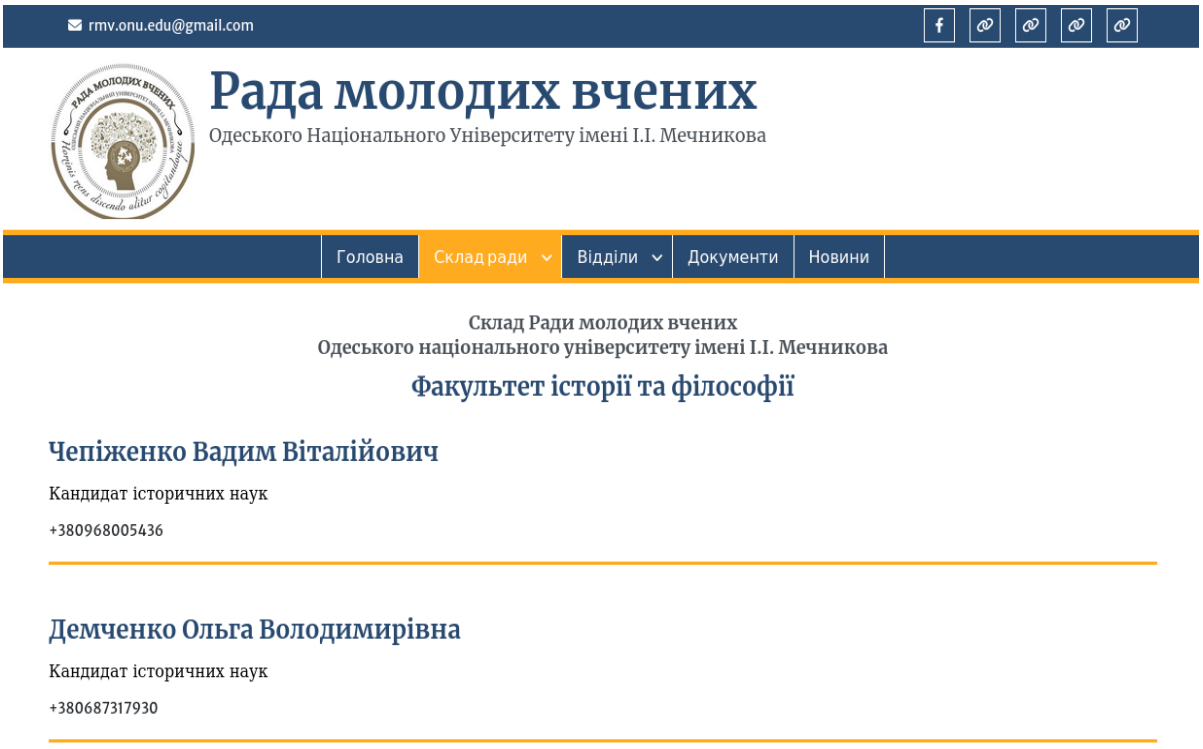

Рисунок 3.11 - Сторінка факультету

Рисунок 3.12 - Сторінка новин

3.4 Використовувані технології

Веб-технології та проекти, які створюються з їх допомогою, розвиваються з величезною швидкістю. Те, що було популярним ще вчора, не обов'язково буде таким завтра. Індустрія веб-розробки відрізняється своєю рухливістю. Одні технології змінюються іншими і навпаки. На сьогодні в бізнес-процеси пропонують рішення практично для будь-яких сфер життя. Сьогодні для того, щоб залишатися конкурентоспроможними, треба постійно «тримати руку на пульсі» і стежити за розвитком цієї індустрії.

Для реалізації front-end (клієнтської) частини програми використовувалися:

– CSS
– JavaScript
– Blade шаблонизатор

А для back-end (серверної) частини програми:

– PHP
– Laravel

Для зберігання даних обрана MySQL.

MySQL - вільна реляційна система управління базами даних [22]. Розробка та підтримка сайта MySQL здійснює корпорація Oracle, яка отримала права на торговельну марку разом з поглиненої Sun Microsystems, яка раніше придбала шведську компанію MySQL AB. MySQL є рішенням для малих і середніх додатків. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється

із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць.

3.5 Висновки розділу

- У третьому розділі описана програмна реалізація веб-додатку.
- У підрозділі 3.1 була розроблена база даних для програми. Описано всі 3 етапи створення реляційної бази даних: концептуальний, логічний та фізичний. Крім того, у розділі присутні описи всіх таблиць, створених на основі із зазначенням їх полів, обмежень та зв'язків.
- У розділі 3.2 уточнено раніше вибрану архітектуру програми. Вона представлена на діаграмі розміщення та описані компоненти програми.
- У розділі 3.3 розроблено основні сторінки сайту.
- У розділі 3.4 описано використовувані технології при розробці сайту.

ВИСНОВОК

У роботі розглянуто структуру інформаційної системи, орієнтована на збирання, обробку та надання даних про звіт роботи РМВ. Проведено аналіз потреб користувачів, розглянуті вже існуючі подібні ресурси, та на основі цього визначено необхідні функції та інші вимоги до системі.

Основним пріоритетом при розробці стала необхідність створити систему максимально доступною та зручною для користувача. Виходячи з цього було прийнято рішення реалізовувати систему як веб-ресурс з використанням PHP фреймворку Laravel.

Була спроектована архітектура програми, заснована на шаблоні MVC, створено базу даних. В результаті було розроблено візуальний макет сайту, а також серверна частина, що відповідає за обробку запитів та реалізацію функціоналу.

Реалізовано базові можливості: CRUD всіх сутностей сайту, можливість в майбутньому легко додавати нові мови до сайту. Весь створений контент зберігається у серверній базі даних MySQL. Використані переваги фреймворку Laravel: механізм міграцій, захист від кросбраузерних атак.

В результаті отриманий додаток вийшов легко масштабованим і відкритим для подальшого розвитку. Ці переваги стали наслідком вибору архітектури MVC та використання фреймворку. Надалі додаток може бути розширено , додано мультиязичність або інші корисні функції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. William, S. Davis The Information System Consultant's Handbook. Systems Analysis and Design. / William S. Davis, David C. Yen – CRC Press, 1998. — 800 с.
2. Класифікація інформаційних систем. — Режим доступа: https://pidru4niki.com/11241013/informatika/klasifikatsiya_informatsiynih_sistem
3. Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер; пер. с англ. – М.: «Вильямс», 2006. — 544 с.
4. Модель-вид-контролер. — Режим доступа: https://uk.wikipedia.org/wiki/Модель-вид-контролер
5. Котеров, Д.В. PHP 7: наиболее полн. рук. / Д. В. Котеров, И. В. Симдянов. – СПб.: БХВ-Петербург, 2016. – 1088 с
6. PHP: Manual. — Режим доступа: http://php.net/manual/en/preface.php (дата обращения - 30.05.2022)
7. The Definitive PHP 7.2, 7.3, 7.4, 8.0, and 8.1 Benchmarks (2022). — Режим доступа: https://kinsta.com/blog/php-benchmarks/
8. Ульман, Л. MySQL / Л. Ульман.– М. : ДМК Пресс, 2008. – 352 с
9. Справочник по HTML. — Режим доступа: htmlbook.ru/html
10. Справочник CSS. — Режим доступа: htmlbook.ru/css
11. Хоган, Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения / Б. Хоган.– СПб.: Питер, 2014. – 320 с.
12. Зервас, Квентин Web 2.0: создание приложений на PHP. / Квентин Зервас; пер. с англ. – М.: ООО «И.Д. Вильямс», 2010. – 544 с.
13. Официальный сайт фреймворка Laravel. — Режим доступа: https://laravel.com
14. Русскоязычное сообщество Laravel. — Режим доступа: https://laravel.su/

15. Scott, W. Ambler Mapping Objects to Relational Databases: O/R Mapping In Detail. — Режим доступа: http://www.agiledata.org/essays/mappingObjects.html

16. Буч, Г. Язык UML. Руководство пользователя. / Г. Буч, Д. Рамбо, И. Якобсон. – М.: ДМК Пресс, 2008. – 496 с.

ДОДАТОК А

Вихідний програмний код back-end

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

namespace App\Http\Controllers;

use App\Models\Department\Department;
use App\Repositories\DepartmentRepository;
use App\Repositories\StaffRepository;
use Illuminate\Http\Request;

class DepartmentController extends Controller
{
    /**
     * @var DepartmentRepository
     */
    private $departmentRepository;
    /**
     * @var StaffRepository
     */
    private $staffRepository;

    public function __construct(DepartmentRepository
$departmentRepository, StaffRepository $staffRepository)
    {
        $this->departmentRepository = $departmentRepository;
        $this->staffRepository = $staffRepository;
    }

    public function show($departmentId)
    {
        $item =
$this->departmentRepository->find($departmentId);
        $head = $this->staffRepository->where('department_id',
$departmentId)->where('type', 'head')->first();
        $with = compact(array_keys(get_defined_vars()));

        return view('public.department.show')->with($with);
    }
}

<?php declare(strict_types=1);

namespace App\Http\Controllers;

use App\Http\Controllers\Admin\AdminController;
use App\Models\Faculty\Faculty;
use App\Repositories\FacultyRepository;
use App\Repositories\StaffRepository;
```

```
use Illuminate\Http\Request;

class FacultyController extends Controller
{
    /**
     * @var FacultyRepository
     */
    private $facultyRepository;
    /**
     * @var StaffRepository
     */
    private $staffRepository;

    public function __construct(FacultyRepository
$facultyRepository, StaffRepository $staffRepository)
    {
        $this->facultyRepository = $facultyRepository;
        $this->staffRepository = $staffRepository;
    }

    public function show($facultyId)
    {
        $faculty = $this->facultyRepository->find($facultyId);
        $staffs =
$this->staffRepository->findAllByFaculty($facultyId);
        $with = compact(array_keys(get_defined_vars()));

        return view('public.faculty.show')->with($with);
    }
}


<?php declare(strict_types=1);

namespace App\Http\Controllers;

use App\DataContainers\News\SearchDataContainer;
use App\Enum\ContentTypeEnum;
use App\Repositories\ContentRepository;
use App\Repositories\NewsRepository;

class HomeController extends SiteController
{
    private $newsRepository;
    /**
     * @var ContentRepository
     */
    private $contentRepository;

    public function __construct(
        NewsRepository $newsRepository,
        ContentRepository $contentRepository
    )
```

```
    {
        parent::__construct();
        $this->newsRepository = $newsRepository;
        $this->contentRepository = $contentRepository;
    }

    public function home(
        SearchDataContainer $newsData
    )
    {
        $news =
$this->newsRepository->getListPublic($newsData);
        $brands =
$this->contentRepository->getListPublicByType(ContentTypeEnum::B
RAND);
        $with = compact(array_keys(get_defined_vars()));

        return view('public.home.index')->with($with);
    }
}

<?php declare(strict_types=1);

namespace App\Http\Controllers;

use App\DataContainers\News\SearchDataContainer;
use App\Models\News\News;
use App\Repositories\NewsRepository;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;

final class NewsController extends SiteController
{
    private $newsRepository;

    private $container;

    public function __construct(
        NewsRepository $newsRepository,
        SearchDataContainer $container,
        Request $request
    )
    {
        parent::__construct();
        $this->newsRepository = $newsRepository;
        $container->fillFromRequest($request);
        $this->container = $container->setOnPage(9);
    }

    public function index()
    {
        $news =
$this->newsRepository->getListPublic($this->container);
```

```
        return
view('public.news.index')->with(compact('news'));
    }

    public function show(string $id)
    {
        $item = $this->newsRepository->find((int)$id);

        return view('public.news.show')->with(compact('item'));
    }

}

<?php declare(strict_types=1);

namespace App\Models\Content;

use App\Contents\ContentFieldsTypeInterface as FI;
use App\Contracts\HasLocalized;
use App\Models\Model;
use App\Traits\Models\ImageAttributeTrait;
use App\Traits\Models\Localization\RedirectLangColumn;
use Illuminate\Database\Eloquent\Relations\MorphTo;

class Content extends Model implements HasLocalized
{
    use RedirectLangColumn;
    use ImageAttributeTrait;

    protected $langColumns = [FI::NAME, FI::TITLE, FI::EXCERPT,
FI::DESCRIPTION, 'language_id'];

    protected $guarded = ['id'];

    protected $hasOneLangArguments = [ContentLang::class];

    public function contentable(): MorphTo
    {
        return $this->morphTo();
    }

    public function getName(): ?string
    {
        return $this->getAttribute(FI::NAME);
    }

    public function getTitle(): ?string
    {
        return $this->getAttribute(FI::TITLE);
    }

    public function getExcerpt(): ?string
```

```
    {
        return $this->getAttribute(FI::EXCERPT);
    }

    public function getDescription(): ?string
    {
        return $this->getAttribute(FI::DESCRIPTION);
    }

}
<?php

namespace App\Models\Content;

use App\Models\ModelLang;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class ContentLang extends ModelLang
{

    protected $fillable = [
        'content_id',
        'language_id',
        'name',
        'title',
        'except',
        'description',
    ];

    protected $primaryKey = ['content_id', 'language_id'];

    public function content(): BelongsTo
    {
        return $this->belongsTo(Content::class);
    }
}

<?php declare(strict_types=1);

namespace App\Models\Department;

use App\Contracts\HasImagesContract;
use App\Contracts\HasLocalized;
use App\Models\Model;
use App\Traits\Models\HasImages;
use App\Traits\Models\ImageAttributeTrait;
use App\Traits\Models\Localization\RedirectLangColumn;

class Department extends Model implements HasLocalized,
HasImagesContract
{
    use HasImages;
    use RedirectLangColumn;
```

```
    use ImageAttributeTrait;

    protected $langColumns = [
        'name', 'title', 'description', 'excerpt',
'language_id',
    ];

    protected $hasOneLangArguments = [DepartmentLang::class];

    protected $casts = [
        'active' => 'bool',
    ];

    protected $guarded = ['id'];

    public static function boot(): void
    {
        parent::boot();
    }

    public function getExcerpt()
    {
        return $this->getAttribute('excerpt');
    }

    public function getDescription(): string
    {
        return (string)$this->getAttribute('description');
    }

    public function getTitle(): string
    {
        return (string)$this->getAttribute('name');
    }

    public function getIsActive(): bool
    {
        return (bool)$this->getAttribute('active');
    }
}

<?php declare(strict_types=1);

namespace App\Models\Faculty;

use App\Contracts\HasImagesContract;
use App\Contracts\HasLocalized;
use App\Models\Model;
use App\Traits\Models\HasImages;
use App\Traits\Models\ImageAttributeTrait;
use App\Traits\Models\Localization\RedirectLangColumn;
```

```
  class Faculty extends Model implements HasLocalized,
HasImagesContract
  {
     use HasImages;
     use RedirectLangColumn;
     use ImageAttributeTrait;

     protected $langColumns = [
         'name', 'title', 'description', 'excerpt',
'language_id',
     ];

     protected $hasOneLangArguments = [FacultyLang::class];

     protected $casts = [
         'active' => 'bool',
     ];

     protected $guarded = ['id'];

     public static function boot(): void
     {
         parent::boot();
     }

     public function getExcerpt()
     {
         return $this->getAttribute('excerpt');
     }

     public function getDescription(): string
     {
         return (string)$this->getAttribute('description');
     }

     public function getTitle(): string
     {
         return (string)$this->getAttribute('name');
     }

     public function getIsActive(): bool
     {
         return (bool)$this->getAttribute('active');
     }

  }

  <?php declare(strict_types=1);

  namespace App\Models\News;

  use App\Contracts\HasLocalized;
  use App\Contracts\Models\HasNextPrevAttributes;
```

```
  use App\Models\Model;
  use App\Traits\Models\HasImages;
  use App\Traits\Models\ImageAttributeTrait;
  use App\Traits\Models\Localization\RedirectLangColumn;
  use App\Traits\Models\PrevNextAttributes;
  use Illuminate\Database\Eloquent\Relations\HasOne;
  use Illuminate\Support\Carbon;

  class News extends Model implements HasNextPrevAttributes,
HasLocalized
  {
     use HasImages;
     use PrevNextAttributes;
     use RedirectLangColumn;
     use ImageAttributeTrait;

     protected $langColumns = [
         'name', 'title', 'description', 'excerpt',
'language_id',
     ];

     protected $hasOneLangArguments = [NewsLang::class,
'news_id'];

     protected $casts = [
         'video' => 'string',
         'is_notified' => 'bool',
         'active' => 'bool',
     ];

     protected $guarded = ['id'];

     public static function getBetweenDates($from, $to)
     {
         return self::whereBetween('published_at', [$from,
$to])->get();
     }

     public static function boot()
     {
         parent::boot();
     }

     public function getSlug(): string
     {
         return (string)$this->getAttribute('url');
     }

     public function prevNew(): HasOne
     {
         return $this->hasOne(__CLASS__,
'prev_id')->with('lang');
     }
```

```
    public function nextNew(): HasOne
    {
        return $this->hasOne(__CLASS__,
'next_id')->with('lang');
    }

    public function getExcerpt()
    {
        return $this->getAttribute('excerpt');
    }

    public function getDescription(): string
    {
        return (string)$this->getAttribute('description');
    }

    public function getName(): string
    {
        return (string)$this->getAttribute('name');
    }

    public function getIsActive(): bool
    {
        return (bool)$this->getAttribute('active');
    }

    public function getNext(): ?self
    {
        return $this->nextNew;
    }

    public function getPublishedAt(): Carbon
    {
        return getDateCarbon($this->published_at);
    }

    public function getPublishedAtFormatted(): string
    {
        return $this->getPublishedAt()->formatLocalized('%e %B,
%R');
    }

}

<?php declare(strict_types=1);

namespace App\Models\Staff;

use App\Contracts\HasImagesContract;
use App\Contracts\HasLocalized;
use App\Models\Model;
use App\Traits\Models\HasImages;
```

```
  use App\Traits\Models\ImageAttributeTrait;
  use App\Traits\Models\Localization\RedirectLangColumn;

  class Staff extends Model implements HasLocalized,
HasImagesContract
  {
     use HasImages;
     use RedirectLangColumn;
     use ImageAttributeTrait;

     protected $table = 'staffs';

     protected $langColumns = [
         'name', 'description', 'language_id',
     ];

     protected $hasOneLangArguments = [StaffLang::class];

     protected $casts = [
         'active' => 'bool',
     ];

     protected $guarded = ['id'];

     public static function boot(): void
     {
         parent::boot();
     }

     public function getDescription(): string
     {
         return (string)$this->getAttribute('description');
     }

     public function getName(): string
     {
         return (string)$this->getAttribute('name');
     }

     public function getEmail(): string
     {
         return (string)$this->getAttribute('email');
     }

     public function getPhone(): string
     {
         return (string)$this->getAttribute('phone');
     }

     public function getIsActive(): bool
     {
         return (bool)$this->getAttribute('active');
     }}
```

ДОДАТОК Б

Вихідний програмний код front-end

Б.1 - Javascript (app.js)

```
'use strict';

let isMobile = {
                    Android:          function()          {return
      navigator.userAgent.match(/Android/i);},
                  BlackBerry:         function()          {return
      navigator.userAgent.match(/BlackBerry/i);},
                     iOS:           function()          {return
      navigator.userAgent.match(/iPhone|iPad|iPod/i);},
    Opera: function() {return navigator.userAgent.match(/Opera
      Mini/i);},
                  Windows:          function()          {return
      navigator.userAgent.match(/IEMobile/i);},
       any:    function()    {return    (isMobile.Android()    ||
      isMobile.BlackBerry()       ||       isMobile.iOS()       ||
      isMobile.Opera() || isMobile.Windows());}
};

const body = document.querySelector('body');
const arrow = document.querySelectorAll('.arrow');

if (isMobile.any()) {
  body.classList.add('touch');

  arrow.forEach(function (arrowElement) {
     let thisArrow = arrowElement;
     let subMenu = arrowElement.nextElementSibling;
     let thisLink = arrowElement.previousElementSibling;

      thisLink.classList.add('parent'); // add rigth margin for
      <a>
     arrowElement.addEventListener('click', function() {
        subMenu.classList.toggle('open'); // display sub-menu
        thisArrow.classList.toggle('active') // rotate arrow
     })
  });

} else {
  body.classList.add('mouse')

  arrow.forEach(function (arrowElement) {
     let thisLink = arrowElement.previousElementSibling;
     thisLink.classList.add('parent'); // add rigth margin
  });

}
```

```
// Burger Menu

let btnBurger = document.querySelector('.menu-burger');
let menu = btnBurger.nextElementSibling;
btnBurger.addEventListener('click', function() {
  btnBurger.classList.toggle('burger-active')
  menu.classList.toggle('menu-active')
})


// Current link
const activePage = window.location.href;
const navbar = document.querySelector('.menu');

const navLinks = document
 .querySelectorAll("a.menu__link, a.sub-menu__link")
 .forEach((link) => {
    if (activePage === link.href) {
     link.parentElement.classList.add('current-li');
                                const      mainMenuItem      =
     link.parentElement.parentElement.parentElement;
                      if     (mainMenuItem     !==     navbar)
     mainMenuItem.classList.add('current-li');
    }
 });
```

Б.2 - CSS (style.css)

```
@import
url(https://fonts.googleapis.com/css2?family=Merriweather+Sans:i
tal,wght@0,400;0,700;1,400;1,700&family=Merriweather:ital,wght@0
,400;0,700;1,400;1,700&display=swap);
.header * {
 margin: 0;
 padding: 0;
 -webkit-box-sizing: border-box;
         box-sizing: border-box;
}

.header a {
 text-decoration: none;
}

.header li {
 list-style: none;
}

.contacts {
 background-color: var(--color-primary);
}
```

```
.contacts__content {
 padding: 5px 20px;
 display: -webkit-box;
 display: -ms-flexbox;
 display: flex;
 -webkit-box-pack: justify;
     -ms-flex-pack: justify;
         justify-content: space-between;
 -webkit-box-align: center;
     -ms-flex-align: center;
         align-items: center;
}

.contacts__ul {
 display: -webkit-box;
 display: -ms-flexbox;
 display: flex;
 -webkit-box-pack: justify;
     -ms-flex-pack: justify;
         justify-content: space-between;
 gap: 15px;
}

.contacts__quick-email,
.contacts__ul a {
 color: #fff;
 -webkit-transition: all 0.3s ease-in-out;
 -o-transition: all 0.3s ease-in-out;
 transition: all 0.3s ease-in-out;
}

.contacts__ul i {
 width: 32px;
 height: 32px;
 padding: 7px;
 border: 1px solid #fff;
 display: -webkit-box;
 display: -ms-flexbox;
 display: flex;
 -webkit-box-pack: center;
     -ms-flex-pack: center;
         justify-content: center;
 -webkit-box-align: center;
     -ms-flex-align: center;
         align-items: center;
 -webkit-transition: all 0.3s ease-in-out;
 -o-transition: all 0.3s ease-in-out;
 transition: all 0.3s ease-in-out;
}

.contacts__quick-email:hover {
 color: var(--color-secondary-light);
```

```
   text-shadow: 0px 2px 3px var(--color-secondary-light);
  }

  .contacts__ul i:hover {
   color: var(--color-secondary-light);
   border-color: var(--color-secondary-light);
   background-color: var(--color-primary-lighter);
   -webkit-box-shadow: 0px 0px 10px 0px
var(--color-secondary-light);
           box-shadow: 0px 0px 10px 0px
var(--color-secondary-light);
  }

  @media (max-width: 550px) {
   .contacts__content {
     -webkit-box-orient: vertical;
     -webkit-box-direction: normal;
         -ms-flex-direction: column;
             flex-direction: column;
     -webkit-box-pack: center;
         -ms-flex-pack: center;
             justify-content: center;
   }

   .contacts__quick-email {
     margin-bottom: 12px;
   }

   .contacts__content a {
     font-size: 0.875rem;
   }

   .contacts__ul i {
     width: 30px;
     height: 30px;
     padding: 5px;
   }
  }
  .logo {
   padding: 10px 20px;
  }

  .logo__figure {
   display: -webkit-box;
   display: -ms-flexbox;
   display: flex;
   gap: 15px;
  }

  .logo__img {
   width: 150px;
   height: 150px;
  }
```

```
.logo__main-header {
 font-family: "Merriweather", serif;
 color: var(--color-primary);
 font-size: 2.75rem;
}

.logo__university {
 font-family: "Merriweather", serif;
 font-weight: normal;
 color: var(--color-text-secondary);
 font-size: 1.25rem;
}

@media (max-width: 768px) {
 .logo__main-header {
   font-size: 2.5rem;
 }

 .logo__university {
   font-size: 1.125rem;
 }
}
@media (max-width: 550px) {
 .logo {
   padding: 5px 5px;
 }

 .logo__figure {
   -webkit-box-orient: vertical;
   -webkit-box-direction: normal;
       -ms-flex-direction: column;
           flex-direction: column;
   -webkit-box-align: center;
       -ms-flex-align: center;
           align-items: center;
   gap: 5px;
 }

 .logo__main-header {
   font-size: 2.25rem;
   text-align: center;
 }

 .logo__university {
   font-size: 1rem;
   text-align: center;
 }
}
.navigation-menu {
 background-color: var(--color-primary);
 border-top: 5px solid var(--color-secondary);
 border-bottom: 5px solid var(--color-secondary);
```

```
}

.menu__list li {
 position: relative;
}

.current-li {
 background-color: var(--color-secondary) !important;
}

.menu a.parent {
 padding-right: 35px;
}

.menu__list {
 display: -webkit-box;
 display: -ms-flexbox;
 display: flex;
 -webkit-box-pack: center;
     -ms-flex-pack: center;
         justify-content: center;
 background-color: var(--color-primary);
}

.menu__list > li {
 border-right: 1px solid #fff;
}

.menu__list > li:first-child {
 border-left: 1px solid #fff;
}

.menu__link {
 display: inline-block;
 padding: 10px 15px;
 font-size: 1rem;
 color: #fff;
}

.menu__link--nonclickable {
 pointer-events: none;
 cursor: default;
}

.sub-menu__list {
 display: none;
 position: absolute;
 top: 40px;
 left: 0;
 min-width: 360px;
 max-height: 231px;
 overflow-y: scroll;
 padding-top: 5px;
```

```
}

.sub-menu__list > li {
 background-color: var(--color-primary);
 border-bottom: 1px solid #fff;
}

.sub-menu__link {
 display: inline-block;
 padding: 10px 15px;
 font-size: 1rem;
 color: #fff;
}

.sub-sub-menu__list {
 display: none;
 position: absolute;
 top: 0;
 left: 100%;
 min-width: 180px;
}

.sub-sub-menu__list > li {
 background-color: var(--color-primary);
 border-left: 1px solid #fff;
 border-bottom: 1px solid #fff;
}

.sub-sub-menu__link {
 display: inline-block;
 padding: 10px 15px;
 color: #fff;
 font-size: 1rem;
}

.arrow {
 position: absolute;
 right: 2px;
 top: 6px;
 padding: 8px;
 color: #fff;
 -webkit-transition: all 0.3s ease-in-out;
 -o-transition: all 0.3s ease-in-out;
 transition: all 0.3s ease-in-out;
}

.sub-menu__arrow {
 right: 12px;
}

.arrow.active {
 -webkit-transform: rotate(-180deg);
     -ms-transform: rotate(-180deg);
```

```
            transform: rotate(-180deg);
    }

    .sub-menu__arrow.active {
     -webkit-transform: rotate(-90deg);
         -ms-transform: rotate(-90deg);
             transform: rotate(-90deg);
    }

    body.mouse .menu__list > li:hover .sub-menu__list {
     display: block;
    }

    body.mouse .sub-menu__list > li:hover .sub-sub-menu__list {
     display: block;
    }

    body.mouse .menu__list li:hover {
     background-color: var(--color-secondary-light);
    }

    body.touch .sub-menu__list.open {
     display: block;
    }

    body.touch .sub-sub-menu__list.open {
     display: block;
    }

    .menu-burger {
     display: none;
     -webkit-box-flex: 0;
         -ms-flex: 0 1 60%;
             flex: 0 1 60%;
     padding: 8px 18px;
     margin: 7px 0;
     font-size: 1.5rem;
     color: #fff;
     background-color: var(--color-primary);
     border: 1px solid #fff;
     -webkit-box-shadow: 0px 0px 7px 4px
var(--color-primary-light);
             box-shadow: 0px 0px 7px 4px
var(--color-primary-light);
     -webkit-transition: all 0.3s ease-in-out;
     -o-transition: all 0.3s ease-in-out;
     transition: all 0.3s ease-in-out;
    }

    .menu-burger__text {
     border-left: 2px solid #fff;
     padding-left: 8px;
    }
```

```
  .fa-xmark {
   display: none;
  }

  .burger-active .fa-bars {
   display: none;
  }

  .burger-active .fa-xmark {
   display: inline-block;
  }

  .burger-active.menu-burger {
   background-color: var(--color-secondary);
   -webkit-box-shadow: 0px 0px 7px 0px
var(--color-secondary-light);
           box-shadow: 0px 0px 7px 0px
var(--color-secondary-light);
  }

  @media (max-width: 768px) {
   .navigation-menu {
     background-color: var(--color-primary);
   }

   .menu {
     display: -webkit-box;
     display: -ms-flexbox;
     display: flex;
     -webkit-box-pack: center;
         -ms-flex-pack: center;
             justify-content: center;
     position: relative;
   }

   .menu-burger {
     display: block;
   }

   .menu__list {
     display: none;
   }

   .menu__list.menu-active {
     display: block;
     position: absolute;
     top: 109%;
     width: 75%;
   }

   .sub-menu__list {
     max-height: 100%;
```

```
  min-width: 220px;
  position: relative;
  left: 0;
  top: 0;
  padding: 0;
  background-color: var(--color-primary-light);
}

.sub-menu__arrow {
  -webkit-transform: rotate(90deg);
      -ms-transform: rotate(90deg);
          transform: rotate(90deg);
}

.sub-sub-menu__list {
  position: relative;
  left: 0;
  top: 0;
  background-color: var(--color-primary-lighter);
}

.menu__list > li:first-child {
  border-left: none;
}

.menu__list > li {
  border-right: none;
  border-bottom: 1px solid #fff;
}

.sub-menu__list > li {
  background-color: var(--color-primary-light);
  border: none;
  border-bottom: 1px solid #fff;
}

.sub-menu__list > li:first-child {
  border-top: 1px solid #fff;
}

.sub-menu__list > li:last-child {
  border: none;
}

.sub-sub-menu__list > li {
  background-color: var(--color-primary-lighter);
  border: none;
  border-bottom: 1px solid #fff;
}

.sub-sub-menu__list > li:first-child {
  border-top: 1px solid #fff;
}
```

```
    .sub-sub-menu__list > li:last-child {
      border: none;
    }

    .sub-menu__link {
      padding-left: 25px;
    }

    .sub-sub-menu__link {
      padding-left: 35px;
    }
  }
  .info__text {
    color: var(--color-text-primary);
    font-size: 1rem;
    padding: 0px 32px;
  }
  @media (max-width: 550px) {
    .info__text {
      padding: 0px 16px;
    }
  }

  .info__title {
    color: var(--color-primary-light);
    font-size: 1.375rem;
    padding: 0px 8px;
    margin-bottom: 16px;
  }

  .rmv-document {
    font-size: 1.125rem;
    padding: 0px 16px;
  }

  .info__img {
    display: block;
    -webkit-box-sizing: border-box;
            box-sizing: border-box;
    border: 5px solid var(--color-primary-light);
    width: 75%;
    height: 75%;
    margin: 0 auto;
    -webkit-box-shadow: 0px 0px 7px 0px
var(--color-primary-light);
            box-shadow: 0px 0px 7px 0px
var(--color-primary-light);
  }
  @media (max-width: 768px) {
    .info__img {
      width: 100%;
      height: 100%;
```

```
 }
}

.info__img--1 {
 width: 90%;
 height: 90%;
}

.info__text--bold {
 font-weight: bold;
}

.info__ul {
 padding: 0px 32px;
}
@media (max-width: 550px) {
 .info__ul {
   padding: 0px 16px;
 }
}

.info__li > .info__text {
 padding: 0px;
}

.news-item {
 padding-bottom: 25px;
 border-bottom: 3px solid var(--color-secondary);
}

.news-item__title {
 color: var(--color-primary-light);
 font-size: 1.25rem;
}

.news-item__full {
 text-decoration: none;
 color: inherit;
}

.news-item__date {
 margin-right: 15px;
 color: var(--color-text-secondary);
}

.fa-calendar-days {
 color: var(--color-secondary);
 margin-right: 5px;
}

.news-item__author {
 color: var(--color-text-secondary);
}
```

```
.fa-user-pen {
 color: var(--color-secondary);
 margin-right: 5px;
}

.news-item__short {
 font-size: 1rem;
}

.news-item__more {
 color: #fff;
 text-decoration: none;
 padding: 8px;
 background-color: var(--color-primary-light);
 border-radius: 7px;
}

.ckeditor-wrapper h1,
.ckeditor-wrapper h2,
.ckeditor-wrapper h3,
.ckeditor-wrapper h4,
.ckeditor-wrapper h5,
.ckeditor-wrapper h6 {
 color: var(--color-primary-light);
}
.ckeditor-wrapper h3 {
 font-size: 1.375rem;
 margin-bottom: 8px;
}
.ckeditor-wrapper p {
 color: var(--color-text-primary);
}
.ckeditor-wrapper li {
 margin-bottom: 7px;
}
.ckeditor-wrapper div {
 color: var(--color-text-primary);
}

.department-chief {
 border-bottom: 3px solid var(--color-secondary);
 padding-bottom: 20px;
 margin-bottom: 45px;
 display: -webkit-box;
 display: -ms-flexbox;
 display: flex;
 gap: 25px;
}
.department-chief img {
 max-width: 200px;
 background-color: #546e8d;
 border: 3px solid var(--color-primary-lighter);
```

```
   -webkit-box-shadow: 0px 0px 7px 0px
var(--color-primary-light);
           box-shadow: 0px 0px 7px 0px
var(--color-primary-light);
  }
  .department-chief .chief__info .chief__position {
   font-size: 1.125rem;
   margin: 0;
  }
  .department-chief .chief__info .chief__name {
   margin-top: 0;
   font-family: "Merriweather", serif;
   font-size: 1.5rem;
   font-weight: bold;
   color: var(--color-primary);
  }
  .department-chief .chief__info .chief__activity {
   font-size: 1.125rem;
   margin-bottom: 40px;
  }
  .department-chief .chief__info .chief__email {
   font-size: 1rem;
   margin-bottom: 5px;
  }
  .department-chief .chief__info .chief__phone {
   font-size: 1rem;
   margin-bottom: 5px;
  }
  @media (max-width: 768px) {
   .department-chief {
     margin-top: 15px;
     -webkit-box-orient: vertical;
     -webkit-box-direction: normal;
         -ms-flex-direction: column;
             flex-direction: column;
     gap: 10px;
   }
   .department-chief img {
     -ms-flex-item-align: center;
         align-self: center;
     margin-top: 10px;
   }
   .department-chief .chief__info .chief__position,
.department-chief .chief__info .chief__name, .department-chief
.chief__info .chief__email, .department-chief .chief__info
.chief__phone {
     text-align: center;
   }
   .department-chief .chief__info .chief__name {
     font-size: 1.25rem;
   }
   .department-chief .chief__info .chief__activity {
     text-align: center;
```

```
    margin-bottom: 10px;
   }
  }

  .member {
   margin-bottom: 15px;
   max-width: -webkit-fit-content;
   max-width: -moz-fit-content;
   max-width: fit-content;
   padding: 7px;
   border: 1px solid var(--color-primary-lighter);
   -webkit-box-shadow: 0px 0px 5px 0px
var(--color-primary-light);
           box-shadow: 0px 0px 5px 0px
var(--color-primary-light);
  }

  .member__name {
   font-size: 1.125rem;
   font-weight: bold;
  }

  .member__phone {
   font-size: 1rem;
   font-weight: normal;
  }

  .member__email {
   font-size: 1rem;
   font-weight: normal;
  }

  .member__faculty {
   font-size: 1rem;
   font-weight: normal;
  }

  .member__activity {
   font-size: 1rem;
   font-weight: normal;
  }

  .home,
  .faculty,
  .department,
  .rmv-news {
   padding: 20px 10px;
  }

  .main__header {
   font-family: "Merriweather", serif;
   text-align: center;
   margin-bottom: 30px;
```

```
}
@media (max-width: 550px) {
 .main__header {
   margin-bottom: 20px;
 }
}
.main__header .header__title--main {
 color: var(--color-primary);
 margin: 0px 0px 5px 0px;
 font-size: 1.75rem;
}
@media (max-width: 768px) {
 .main__header .header__title--main {
   font-size: 1.5rem;
 }
}
.main__header .header__title--pre {
 color: var(--color-text-secondary);
 margin: 0;
 font-size: 1.25rem;
}
@media (max-width: 768px) {
 .main__header .header__title--pre {
   font-size: 1.125rem;
 }
}

.footer {
 border-top: 5px solid var(--color-secondary);
 border-bottom: 5px solid var(--color-secondary);
 padding: 10px 0;
 background-color: var(--color-primary);
}

.footer__content {
 display: -webkit-box;
 display: -ms-flexbox;
 display: flex;
 -webkit-box-pack: center;
     -ms-flex-pack: center;
         justify-content: center;
 -webkit-box-align: center;
     -ms-flex-align: center;
         align-items: center;
}

.footer__text {
 color: #fff;
 font-size: 1rem;
}

html {
 height: 100%;
```

```
}

body {
 margin: 0;
 min-height: 100%;
 display: -webkit-box;
 display: -ms-flexbox;
 display: flex;
}

.index-page {
 display: -webkit-box;
 display: -ms-flexbox;
 display: flex;
 -webkit-box-orient: vertical;
 -webkit-box-direction: normal;
     -ms-flex-direction: column;
         flex-direction: column;
 -webkit-box-flex: 1;
     -ms-flex-positive: 1;
         flex-grow: 1;
 min-width: 300px;
}

.main {
 -webkit-box-flex: 1;
     -ms-flex-positive: 1;
         flex-grow: 1;
}

:root {
 --color-primary: #294a70;
 --color-primary-light: #3e5c7e;
 --color-primary-lighter: #546e8d;
 --color-secondary: #ffab1f;
 --color-secondary-light: #ffbc4c;
 --color-text-primary: #212529;
 --color-text-secondary: #495057;
}

body {
 margin: 0;
 font-family: "Merriweather Sans", sans-serif;
}

.container {
 max-width: 1140px;
 margin: 0 auto;
}
```

Б.3 - Blade шаблон (app.blade.php)

```
<!DOCTYPE html>
<html lang="ru">
@include('public.layout.includes.head')
<body>

@includeIf('public.layout.includes.preloader')

@hasSection('content')
   @yield('content')
@else
   {!! $content ?? '' !!}
@endif

@include('public.layout.includes.assets.scripts')
</body>
</html>
```