Пояснювальна записка
до дипломного проекту
Прилуцького Павла Валерійовича
на тему: «Автоматизована система управління сайтом університету»

Київ – 2019 рік

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

MVC – Model-View-Controller;

HTML – HyperText Markup Language;

UI – User Interface;

UX – User eXperience;

ORM – Object-Relational Mapping;

Url – Uniform Resource Locator;

Json - JavaScript Object Notation;

SMTP – Simple Mail Transfer Protocol;

AJAX – Asynchronous Javascript And Xml.

ВСТУП

Університет завжди був важливим етапом у житті майже кожної людини. Його вплив змінювався, навчальна програма, кількість студентів та викладачів. Але суть не змінювалась – кожен мав отримати знання для своєї майбутньої професії, а університет завдяки своїм досягненням та вдосконалень повинен був викликати повагу.

Зараз відкривається велика кількість як державних, так і приватних університетів. Є навіть такі, в яких можна навчатися вдома і лише приїжджати на складання екзаменів. Але таке різноманіття у виборі не завжди є корисним, оскільки абітурієнту дуже важко обрати саме той напрямок освіти, у якому він хоче розвиватися. Проблема навіть не в тому, що надто багато кандидатів, а в тому, що не завжди можна найти повну та чітку інформацію про університет. Якщо раніше кожна людина повинна була їхати до самого університету та дивитися на велику дошку, яка описувала його діяльність, то зараз всі звикли до того, що у кожного, навіть найменшого підприємства або державної установи повинен бути сайт, на якому можна знайти всю необхідну інформацію. Особливо замислитися над цією проблемою повинні технічні університети, бо майбутні студенти, саме ознайомившись з сайтом, можуть зрозуміти, наскільки він розвивається у технологічному плані та чи варто до нього йти.

До всього, що було описано вище, вже прийшли майже всі університети, але цього замало. Сайт повинен бути зрозумілим, інформативним та сучасним, адже розвиток технологій робить користувача вибагливим і, якщо сайт його не задовольняє, то він може просто піти з нього не ознайомившись з інформацією. Також потрібно чітко розуміти, чим сайт навчального закладу повинен відрізнятися від звичайного сайту. Наприклад, застосунок не повинен бути звичайним односторінковим сайтом, на ньому повинні бути новини,

інформація про факультети, де розміщена інформація про кафедри які на ньому присутні та викладачів, які працюють на них.

Основною метою дипломного проекту було розробити незалежну систему, яка поєднає у собі зрозумілий та сучасний сайт та багатофункціональну адміністративну панель.

Задачею проекту було на основі огляду існуючих рішень розробити архітектуру додатку, запрограмувати її та протестувати.

Практичним використання проекту є те, що його можна застосувати до будь-якого університету баз особливих модифікацій.

У розділі 1 описується постановка задачі.

У розділі 2 досліджуються існуючі рішення.

У розділі 3 обґрунтовується вибір засобів реалізації.

У розділі 4 міститься реалізація системи.

У розділі 5 проведено тестування системи.

У розділі 6 представлено інструкцію користувача.

У додатку А знаходиться структура бази даних системи.

У додатку Б зображено діаграму активності.

У додатку В описується блок схема автентифікація.

У додатку Г наведена діаграма послідовної роботи системи запланованих публікацій.

У додатку Д знаходиться лістинг програмного коду.

1. ПОСТАНОВКА ЗАДАЧІ

Метою дипломного проекту була розробка застосунка, який поєднує у собі як сучасний та зрозумілий інтерфейс для користувача, так і місце, де університет може тримати свою інформацію про факультети, викладачів, кафедри та середовище для створення новин, сторінок та користувачів застосунка. Система повинна бути незалежною від впливу сторонніх факторів.

Одним з завдань було продумати UI/UX сайту. Цей крок є необхідним, оскільки в наш час для людей є важливим, наскільки сайт добре виглядає та наскільки легко ним користуватися. Найчастіше, якщо користувач не може розібратися в додатку за першу хвилину або дві, то він його залишає в пошуках кращого, більш зрозумілішого. Застосунок повинен бути адаптований під всі браузери та платформи. Також повинна бути адаптація під всі розміри екранів, адже, за статистикою, кількість користувачів мобільних телефонів досягла числа тих, хто використовую комп'ютери. Зараз з'являються різноманітні пристрої з різними розмірами екранів, то ж адаптація повинна бути під кожний з них. Присутність посилань на соціальні мережі необхідні для забезпечення інформації про життя університету. На головній сторінці повинна бути лише найнеобхідніша інформація для ознайомлення. Якщо вона зацікавить користувача, то всю іншу інформацію він зможе знайти вже на інших сторінках. У всі поля, де користувач вводить щось самостійно, потрібно додати екранування символів, щоб ті, хто хоче нашкодити системі або дістати інформацію, не змогли записати в ці поля різні скрипти. Кнопку для входу в систему було вирішено не додавати на сайт, адже це може збити студента з пантелику, бо існують системи, де кожен користувач може зареєструватися, а в нашому випадку університет сам надає доступ до свого сайту. Необхідно додати можливість перемикання мов та пошуку по всім сторінкам. У кожної новини повинні бути один або декілька тегів, які є ключовими словами для цієї новини і після кліку по тегу користувач повинен переходити на сторінку

пошуку де вже показані всі новини з цим тегом. На головній сторінці розмістити слайдер з основними картинками університету та формою зворотного зв’язку. Сайт повинен мати сучасний дизайн для того, щоб зацікавити абітурієнта ще на етапі ознайомлення.

Другим кроком є розробка адміністративної частини системи, яка буде доступна тільки працівникам університету для супроводження та підтримки системи. Вона і є основним керуючим механізмом системи. Одними з головних завдань є:

- додати можливість створювати, редагувати та видаляти новини про життя університету з автоматичною публікацією;
- розділити новини на дві категорії – новини та події;
- розробити повноцінну університетську структуру до якої належать факультети, кафедри та викладачі. Всі сутності повинні бути пов’язані між собою;
- можливість самостійно редагувати верхнє та нижнє меню, додавати варіант з одним рівнем вкладеності, можливість самостійно визначати порядок пунктів меню;
- зміна посилань на соціальні мережі;
- проробити також UI/UX системи, бо чим краще працівники будуть орієнтуватися в застосунку, то тим швидше та якісніше вони виконуватимуть свою роботу;
- система повинна мати декілька мов для того щоб залучити студентів з інших країн;
- додати функціонал для експортування всіх підписників сайту до файлів ексель;
- розробити форму для автентифікації користувачів;
- можливість зміни всіх надписів через адміністративну панель, щоб не залучати кожного разу розробників;

- система повинна мати медіабазу з категоріями в які можна додавати картинки;
- можливість виходу з застосунку;
- підписка користувачів;
- додати екранування символів, щоб не можна було записати скрипти у блок з текстом;
- створення та деактивування користувачів;
- зворотній зв’язок.

2. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1 Сайт та адміністративна панель Новосибірського університету

Звісно, перед тим як приступити до розробки власної системи потрібно подивитися, які схожі варіанти вже представлені на ринку. З одного боку, у вільному доступі є сотні різних варіантів сайтів навчальних закладів, а з іншого, немає доступу до їх адміністративних панелей, оскільки треба бути їх співробітниками або студентами. Але «Новосибирский государственный университет Настоящая наука»[1] цілком наглядно описав структуру її адміністративної панелі.

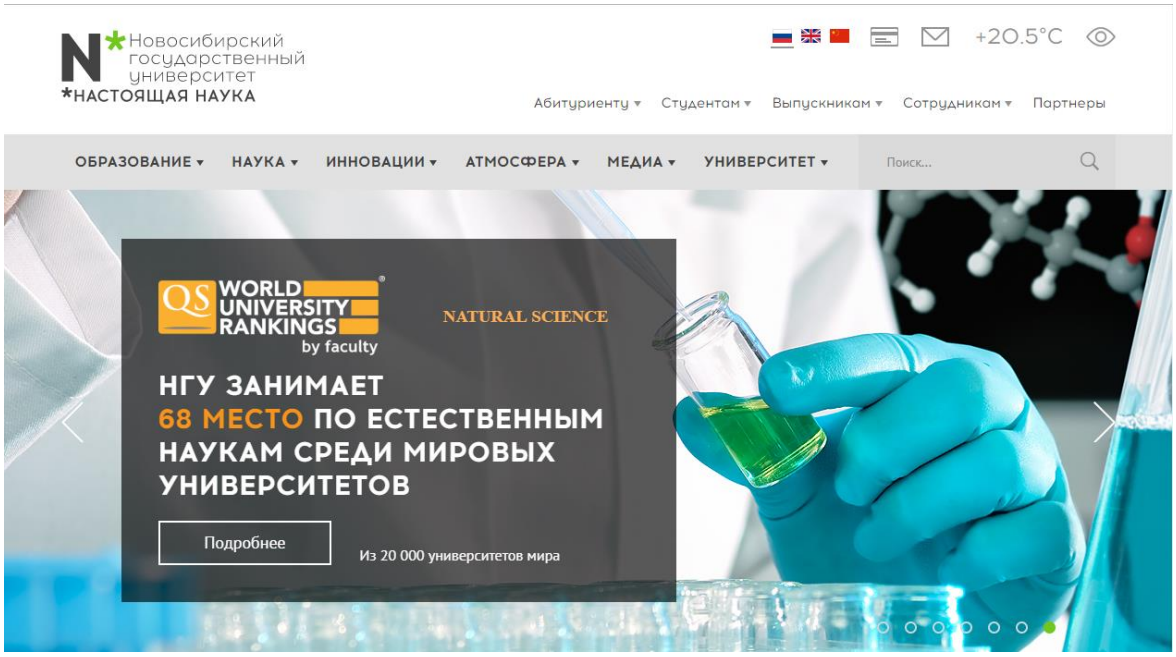

Рисунок 2.1 – Новосибірський університет

Переваги сайту університету (рисунок 2.1):

- сучасний дизайн;
- багатомовність;
- адаптивність;

– присутній режим для людей з поганим зором, де користувач може підібрати для себе розмір шрифту та контрастність тексту та фону сторінки (рисунок 2.2).

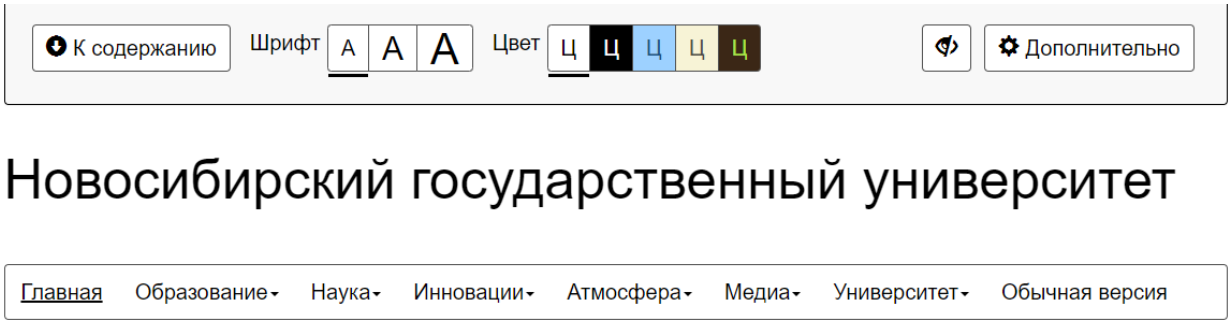

Рисунок 2.2 – Режим для людей з поганим зором

Недоліки сайту:

– погано продуманий UX (наприклад, слайдер настільки швидкий, що не встигаєш прочитати текст);
– на головній сторінці розташовані одразу всі новини, що призводить до розсіюванню уваги користувача;
– через те, що вантажаться багато новин с картинками, перші декілька секунд він гальмує;
– у нижній частині погано підібрані кольори тексту і фону, через що погана читабельність.

Переваги адміністративної панелі:

– застосунок зроблений схожим на комп’ютерну файлову систему, тож користувачу легко призвичаїтись до роботи;
– можливість додавати різні шаблони на сторінки (наприклад слайдер);
– можливість додати контент з іншого сайту;
– широкий набір для форматування новин.

Недоліки адміністративної панелі:

- розробка велась без допомоги візуальних ефектів, а з використанням звичайного HTML;
- жорстка та не зрозуміла система меню, в якої важко буде розібратися початківцям;
- немає інформації про факультети та кафедри;
- сайт розроблений за допомогою сторонньої системи, що ставить під сумніви безпеку сайту, адже приватна компанія не може дати ніяких гарантій про надійне зберігання даних користувачів.

Оскільки більше не вдалось знайти у відкритому доступі адміністративних панелей інших університетів, то надалі буде оцінюватися тільки переваги та недоліки саме сайтів.

2.2 Сайт Київського національного університету

Наступним для порівняння я обрав сайт Київського національного університету імені Тараса Шевченка[2].

Переваги сайту університету (рисунок 2.3):

- швидко вантажиться;
- багатомовність;
- майже все оформлено у вигляді посилань.

Недоліки:

- дизайн не змінювався декілька років;
- не адаптивний;
- деяких елементів, такі як зміна мови, майже не видно;
- меню та назва закладу зображена важким для читання білим та жовтим кольорами.

Рисунок 2.3 – Сайт КНУ

2.3 Сайт Харківського національного університету

І останній на черзі це «Харківський національний університет імені В. Н. Каразіна»[3].

Переваги сайту університету (рисунок 2.4):

– велика кількість матеріалу;
– онлайн бібліотека на сайті;
– багатомовність;
– посилання на соціальні мережі.

Недоліки:

– дрібний текст;
– немає інформації про факультети;
– нагромадження великої кількості інформації на головній сторінці;
– співвідношення картинок до тексту надзвичайно мале, що не дає повної картини новин;
– не адаптивний;

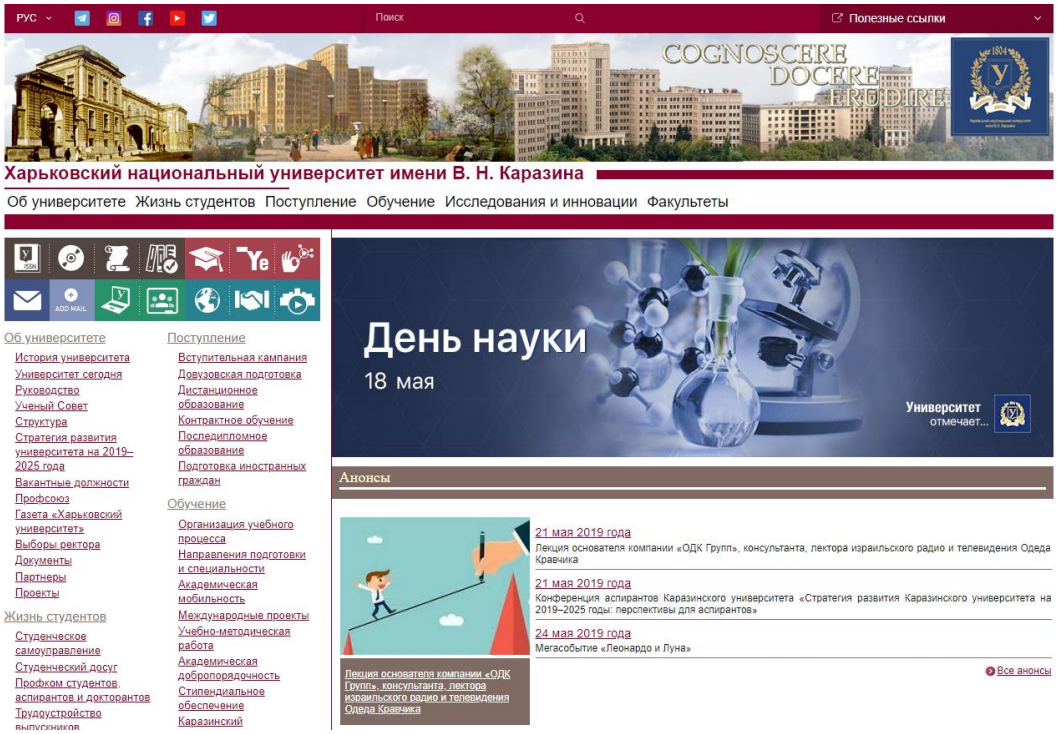

Рисунок 2.4 – Сайт харківського національного університету

Висновки до розділу 2

Переглянувши всі існуючі рішення, було виділено найчастіші помилки при створенні сайтів: адаптивність, використання картинок, швидкість завантаження. Що стосується адміністративної панелі, то зрозуміло, що основну увагу потрібно приділити створенню більш широкої функціональності, яка забезпечить усі потреби користувачів, але необхідно і не забути про зовнішній вигляд, для роботи працівників.

3. ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

3.1 Обґрунтування вибору мови програмування

Вибір мови програмування є одним з найважливіших етапів під час створення додатку. Потрібно чітко розуміти, що саме повинен робити застосунок та які проблеми він повинен вирішувати. По-перше застосунок є веб орієнтованим, що суттєво зменшує коло вибору. Також не слід забувати, що гарний зв'язок з базою є необхідним оскільки велика кількість інформації повинна бути доступна за лічені секунди. І наостанок ця мова повинна інтегруватися з мовою розмітки.

Тож зважаючи на все вищесказане для виконання дипломного проекту було обрану мову PHP через ряд деяких переваг:

- найбільш вагомий фактор це те, що цю мову я знаю найкраще і можу наперед визначити з якими труднощами можна зіштовхнутися і як з ними боротися;
- PHP з’явився давно, тож в Інтернеті можна знайти достатньо матеріалів;
- легко працювати з HTML;
- легко інтегрувати з будь-якою базою даних. Навіть більшість книжок називаються типу «PHP та MySql», бо вони завжди йдуть в парі;
- мова PHP може використовуватись на різних платформах змінивши лише пару конфігураційних файлів;
- безперечно, найпопулярніша мова в світі для програмування веб застосунків.

Але і варто згадати про декілька недоліків цієї мови програмування:

- необхідно бути обережним з обчисленнями, оскільки PHP не закінчує роботу програми, якщо, наприклад, поділити на нуль, а просто присвоїть йому значення false, що може трактуватися як 0;

– буває важко дотримуватися стандартів написання коду, що в майбутньому призведе до поганої читабельності коду;
– недосконалий набір роботи з виключеннями.

Також потрібно розуміти, що важко говорити про PHP як про одно мову, адже версії 5.6 та 7.2 мають велику кількість відмінностей як і зі сторони програміста, так і швидкодію для кількості пам’яті для клієнта. Наприклад, в версії 5.6 глобальні змінні можна було визначати за межами коду на сторінці. У версії 6.0 це виправили.

3.2 Безпека та швидкість

Проблеми з безпекою існують в файлі php.ini – головному конфігураційному файлі PHP. Там зберігаються параметри для всіх платформ і там їх сотні та не всі з них добре описані.

Слід зазначити, що найбільшою проблемою в безпеці є самі розробники, адже, як сказано вище, тільки в файлі php.ini є сотні варіантів для зміни конфігурації і одна зміна має вплив на десятки інших параметрів. Наприклад, якщо встановити log_errors = off чи error_reporting = 0 то всі помилки не будуть відображатися та логуватися, і це призведе до того, що розробник буде думати, що програма працює правильно, а якщо все таки зрозуміє, що щось не так, то цю помилку буде дуже складно знайти. Тож в деяких випадках ця мова з однієї з найпростіших та зрозуміліших перетворюється в складну та важко конфігуровану. І не варто забувати, що ретельний підхід до написання коду зменшує ймовірність помилок.

Мова має обширну документацію та велику кількість книжок в яких можна знайти статті з описами того, як зробити свій застосунок безпечнішим.

Часто можна почути, що мову PHP вважають повільною[4]. Це дійсно так, якщо невміло нею користуватися, а якщо, наприклад, використовувати для неї додатки написані на більш низкорівневих мовах програмування або

використовувати систему кешування, то вона може бути однією із найшвидших.

PHP забезпечує розробника багатьма функціями для взаємодії з файловою системою: file_exists(), system() та інші. Але не слід забувати, що використовувати такі функції зручно і на малих застосунках можна практично нічого не втратите у швидкості, але на великих продуктах кожна така функція може призвести до великих втрат у часу. Слід провести дослідження в інтернеті або самостійно, як така чи інша функція впливає на швидкість програми.

Одним із важливих пунктів при виборі мови програмування є обмеження на час виконання та обсяг використовуваної пам’яті. На відміну від такої мови як Perl, PHP має таке обмеження, яке можна задати в php.ini. Але важко однозначно сказати що це недолік або перевага. Якщо скрипт використовує більше ніж дозволено пам’яті, або виконується довше дозволеного часу, його робота припиняється, що може призвести до втрати даних.

3.3 Фреймворк Zend Framework 3

У дипломному проекті було використано останню версію Zend Framework. Фреймворк – бібліотека, що надає розробнику базу і стандартні шляхи для створення застосунків. Zend це безкоштовний фреймворк з відкритим вихідним кодом, перша версія якого була випущена в 2007 році, що робить його найстаршим серед популярних нині PHP фреймворків[5] та свідчить про те, що йому вдається кожного разу відстоювати свої лідерські позиції у цій галузі. Фреймворк – це шматок програмного забезпечення, який можна використовувати для свого додатку, бо він має шляхи вирішення ряду повсякденних задач з якими стикається програміст. Було вирішено використовувати саме його, тому що в нього вже є реалізовані автентифікація,

авторизація, взаємодія з базою і там є багато інших корисних речей на які може покластися розробник.

Також перевагами є:

- зручне масштабування додатку;
- шаблон MVC;
- використання ORM – Doctrine.

Doctrine – найпопулярніша ORM та найпопулярніший спосіб з'єднання з базою даних у мові PHP. На рисунку 3.1 зображено принцип дії ORM. Вона як прошарок між кодом та базою даних.

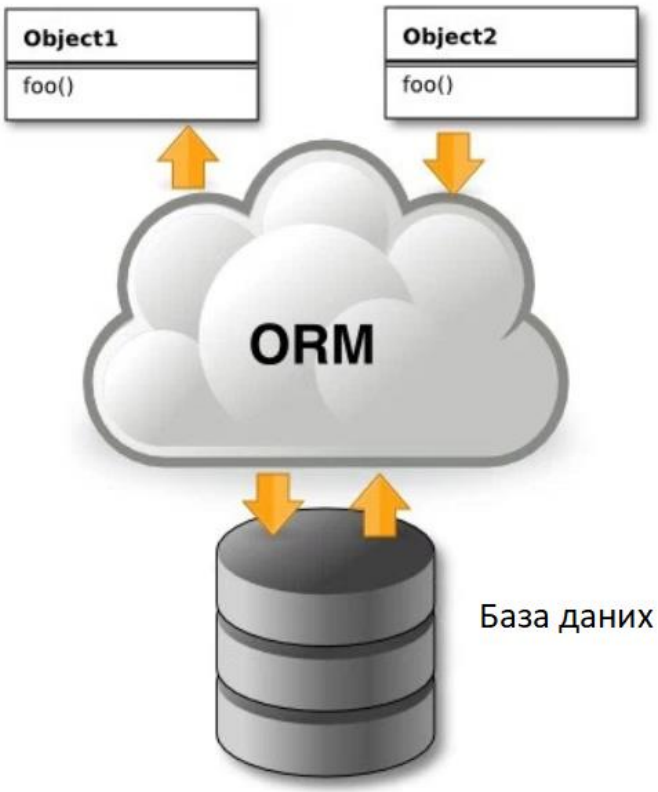

Рисунок 3.1 – Приклад роботи ORM[6]

Ключовою характеристико є те, що вона використовує власний об'єктно-орієнтований діалект – dql. За допомогою її можна використовувати базу даних, як об'єкт.

Висновки до розділу 3

Звернувши увагу на все вище сказане, обраний стек технологій повністю відповідає всім вимогам, адже безпека, швидкість та масштабування є основними перевагами кожного застосунка. Також великий плюс до вибору цього стеку технологій є велика база інформації мови PHP та інших технологій, що будуть використані у додатку. Завдяки обмеженням Zend та Doctrine буде мінімізована можливість виникнення помилок на етапі архітектури.

4. РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

4.1 База даних

Під час розробки застосунка, треба звернути велику увагу на вибір бази даних. База даних – сховище інформації з різними типами даних. У проекті було використано реляційну базу даних, так як, в цьому випадку використання залежностей та зав'язків є необхідним. Для того, щоб взаємодіяти з базами необхідно використовувати систему управління базами даних (СУБД). В кожної СУБД є свої типи даних, але основні з них є у кожної.

Для проектування був вибір між трьома найпопулярнішими СУБД:

- MySQL;
- SQLite;
- PostgreSQL.

Після порівняння усіх факторів довго не вдавалося визначитися, тому що вони схожі за своїм функціоналом, але детальніше проаналізувавши всі обрані варіанти було обрано MySQL і ость чому я зробив такий вибір:

Ця СУБД є найпоширенішою серед усіх інших, а це свідчить, що вона є затребувано у світі;

- більшість функцій SQL присутні в ній;
- MySQL легко піддається масштабування, тож розвитку проекту це ніяк не зашкодить;
- швидкість на високому рівні;
- багато функцій для гарантування безпеки додатку.

Також хочу перерахувати, декілька найбільш вагомих причин чому я не обрав інші СУБД.

SQLite:

- невисока продуктивність роботи;
- погане масштабування для середніх за розміром проектів.

PostgreSql:

– найгірша з усіх швидкість на читання даних, а оскільки велика частина додатку саме інформативна, то цей параметр є критичним;
– важка підтримка, необхідний досвідчений спеціаліст.

Кількість таблицю у веб додатку складає 19, кожна з яких містить різноманітну інформацію про стан системи. Структура бази даних знаходиться в додатку А.

Department – містить інформацію про кафедри, структуру таблиці представлено в таблиці 4.1.

Таблиця 4.1 – Таблиця department

Назва колонки	Тип даних	Призначення
id	int(11)	Унікальний ідентифікатор , який надається для кожної кафедри
title_UA	varchar(150)	Заголовок
title_EN	varchar(150)	Заголовок
content_UA	text	Опис кафедри
content_EN	text	Опис кафедри
seo_title_EN	varchar(150)	Заголовок, що буде використовуватися для просування сайту в інтернеті
seo_title_UA	varchar(150)	Заголовок, що буде використовуватися для просування сайту в інтернеті
seo_description_UA	varchar(300)	Опис, що буде використовуватися для просування сайту в інтернеті

Продовження таблиці 4.1

Назва колонки	Тип даних	Призначення
seo_description_EN	varchar(300)	Опис, що буде використовуватися для просування сайту в інтернеті
seo_keywords_RU	varchar(300)	Ключові слова, що буде використовуватися для просування сайту в інтернеті
seo_keywords_EN	varchar(300)	Ключові слова, що буде використовуватися для просування сайту в інтернеті
thumbnail	varchar(255)	Шлях до картинки
status	varchar(20)	Статус кафедри(активна чи ні)
create_at	datetime	Коли була створена запис
autror_id	int(11)	Хто створив запис
route	varchar(255)	Шлях, який буде відображатися у стрічці пошуку

Таблиця faculties за змістом дуже схожа на попередню, але також має свої відмінності, саме ці відмінності були винесені до таблиці 4.2.

Таблиця 4.2 – Таблиця faculties

Назва колонки	Тип даних	Призначення
programme_RU	text	Освітня програма
programme_EN	text	Освітня програма
pdf_EN	varchar(255)	Шлях до прикріпленого файлу
pdf_RU	varchar(255)	Шлях до прикріпленого файлу

Таблиця teachers на відміну від інших має поле position на двох мовах, де міститься інформація про посаду, яку займає викладач.

Коли обирався спосіб, як поєднати факультети, кафедри та викладачів вибір зупинився на зв'язку багато до багатьох, тому що цей принцип для університету є найбільш вдалим. Було створено дві таблиці faculties_rel_department та faculties_rel_teachers, де розмістив унікальні ідентифікатори кожної з систем, як зображено на рисунку 4.1. Також ще присутні таблиці де використовується зв'язок багато до багатьох з такою структурою: media_rel_categories, news_rel_tags.

	id	faculty_id	department_id
1	19	2	2
2	70	4	1
3	71	4	2
4	80	5	1
5	81	5	2
6	82	5	3
7	83	3	1
8	84	3	2

Рисунок 4.1 – Таблиця faculties_rel_department

Таблиця migrations – зберігає номери усіх міграцій, що вже були використані в системі для того, щоб, коли переносиш зміни на сервер вони не застосувалися знову, адже це призведе до помилки. Міграції видаляти не можна, бо якщо не знаходиться міграція, номер якої є в базі, система віддає помилку, що не може знайти такий файл.

У таблиці feedback міститься інформацію про відгук, що залишають користувачі з усією іншою інформаціє про себе як з форми, яка знаходиться

на головній сторінці, так і форми, яка знаходиться на сторінці контактної інформації.

Таблиця 4.3 – Таблиця feedback

Назва колонки	Тип даних	Призначення
id	int(11)	Унікальний ідентифікатор кожного відгука
content	text	Місце де зберігається інформація про країну та факультет до якого планує вступати користувач
date	datetime	Дата, коли залишили відгук
ip	varchar(80)	Ip користувача
name	varchar(150)	Ім’я користувача
email	varchar(50)	E-mail користувача

Таблиця menu відповідає за верхнє та нижнє меню сайту, де зберігається їх структура, тож вона має лише два рядки з перекладами та системною назвою.

Таблиця 4.4 – Таблиця menu

Назва колонки	Тип даних	Призначення
id	int(11)	Унікальний ідентифікатор кожного відгука
name_RU	varchar(255)	Назва на російській мові
name_EN	varchar(255)	Назва на англійській мові
code	text	Структура меню, де зберігається рівень вкладеності
system_name	varchar(150)	Верхнє чи нижнє меню

Таблиця users відповідає за користувачів, які є зареєстровані у системі. Там зберігається їх ім’я, фото, телефон, пароль та інша особиста або службова інформація.

Таблиця 4.5 – Таблиця users

Назва колонки	Тип даних	Призначення
id	int(11)	Унікальний ідентифікатор кожного користувача
email	varchar(128)	E-mail
firstname	varchar(300)	Ім’я
lastname	varchar(300)	Прізвище
status	int(11)	Статус, активний чи не активний користувач
pwd_reset_token	varchar(32)	Токен, для зміни пароля
data_created	datetime	Дата регістрації користувача в системі
photo	varchar(255)	Шлях до фото
phone	varchar(255)	Номер телефону
password	varchar(150)	Пароль у вигляді хешу

На рисунку 4.2 зображено інші таблиці (news, pages, settings), але їх не було розглянуто через те, що вони містять всі або схожі елементи до попередніх таблиць.

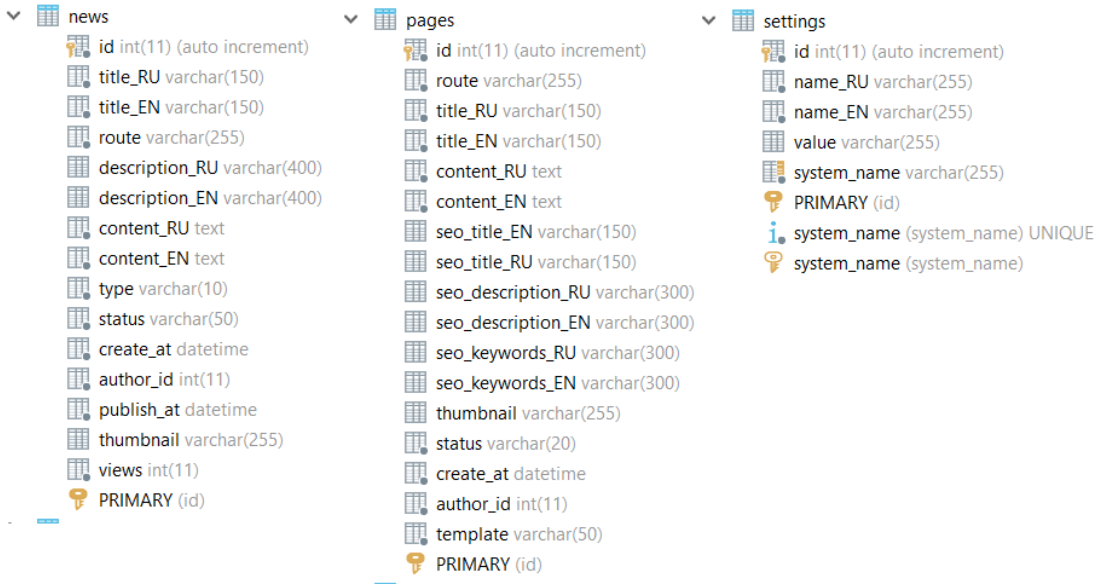

Рисунок 4.2 – Таблиці news, pages, settings

4.2 Структура коду та архітектурний шаблон

Однією з найбільших переваг використання фреймворку, в нашому випадку Zend Framework 3, це те, що він дає нам вже готову структуру файлів, яка перевірена багатьма користувачами, що постійно дають відгуки, як зробити її кращою, а розробники проекту вже багато років підтримують його та випускають нові версії. Готова структура сприяє тому, що розробник намагається дотримуватися правил написання коду або як ще їх називають – best practices. Також не слід забувати, що єдина структура краще дозволяє взаємодіяти з іншими розробниками.

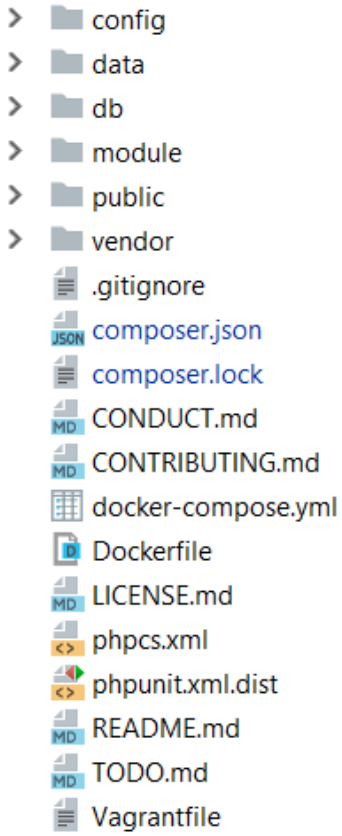

Рисунок 4.3 – Файлова структура проекту

Zend має доволі просту і зрозумілу структуру (рисунок 4.3). У папці config зберігаються основні конфігурації системи. Там можна налаштувати підключення до бази даних, драйвер для анотацій або SMTP для конфігурації відправки листів. У папці data зберігається кеш додатку, міграції до бази даних та шрифти, якщо не задовольняє стандартний. У файлі з міграцією містяться методи up і down. В up вказується те, що хочемо зробити з базою, а down пишемо зворотній запит на той випадок, якщо міграція виявиться невдалою і потрібно її відкотити.

Серед файлів найбільше необхіднішими є composer.json та composer.lock. composer.json – це файл конфігурації Composer, де вказується, які пакети потрібні в додатку. А composer.lock містить інформацію про пакети, які встановлені через Composer. Інші файли відносяться до стандартних файлів фреймворку, але в додатку використовуватися не будуть, тому не будемо зупинятися на них більш детальніше. Всі залежності, що встановлені,

зберігаються у папці vendor, тобто це папка для сторонніх бібліотек. Файли в папці vendor не додаються до репозиторію проекту, а коли інший розробник встановлює собі застосунок, то він просто скачує ті версії файлів, які вказані в composer.json та composer.lock До речі, всі файли даного фреймверку також зберігаються в цій папці. Папка public містить у собі медіа-, css-, html- та js-файли.

У папці db міститься всі сутності та репозиторії до них. Сутністю є клас в якому є всі властивості з геттерами и сеттерами для взаємодії з ними. У кожної властивості є анотація, де вказано як називається колонка в базі даних, тип даних, довжина та чи може колонка бути пустою. Можна це все побачити на рисунку 4.4.

```
/**
 * @var string
 *
 * @ORM\Column(name="status", type="string", length=20, nullable=false)
 */
private $status;
```

Рисунок 4.4 – Приклад властивості сутності

Папка module – це місце де зосереджено 90% всього коду програми, адже налаштування проекту робиться один раз, міграції пишуться рідко і нові сутності майже не створюються, бо їх зазвичай закладають на етапі розробки архітектури застосунка. Для зручності було розділено сайт від адміністративної панелі, розмістивши їх в папках App та Manage відповідно.

Було обрано архітектурний шаблон MVC, бо він є одним з найпопулярніших серед веб застосунків та має зрозумілу структуру. З назви зрозуміло, що головними елементами в ньому є модель, представлення та контролер – основні складові керуючої логіки. Всі частини є незалежними одна від одної, що дозволяє легко модифікувати одну компоненту не

зачіпаючи іншу. Оскільки цей шаблон вже багато років є одним з найпопулярніших, то зупинилися на ньому. Головними елементами в ньому є:

– модель – взаємодіє з базою даних та змінює поведінку в залежності від запиту контролера;
– контролер – в залежності від дій користувача сповіщає модель про зміну його стану;
– представлення – відповідає за відображення інформації.

На рисунку 4.5 можна чітко побачити структуру виконання дій. Пунктирними лініями зображена керуюча інформація (id новини чи користувача).

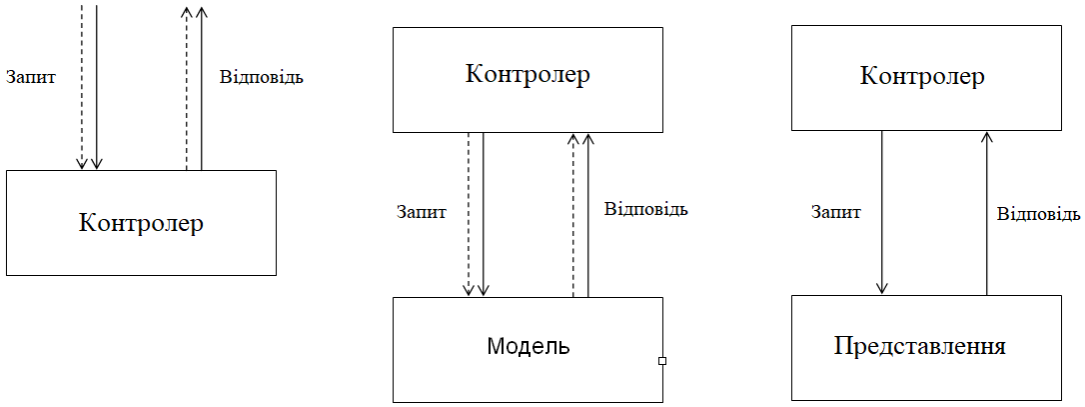

Рисунок 4.5 – Модель MVC

Для наочності використано приклад:

1) спочатку користувач, який переходить на сторінку новин, провокує виклик контролера для цієї сторінки. Контролер знає яку саме інформацію необхідно віддати;
2) далі контролер передає інформацію до моделі;
3) модель дістає з бази всі необхідні дані в тому вигляді, в якому запросив контролер, та віддає назад;

4) контролер відправляє запит до представлення з необхідними даними для відображення;
5) отримавши дані від представлення контролер відображає дані клієнту.

4.3 Розробка системи

Перед тим, як перейти до написання коду потрібно правильно сконфігурувати проект, щоб всі компоненти правильно взаємодіяли між собою. Перш за все, потрібно приєднатися до бази даних, що не вимагає вагомих зусиль. Як показано на фрагменті коду нижче, для зв'язку використано Doctrine, одну з найпопулярніших ORM. Потім вказано адресу хоста, на якому розташована база даних, в даному випадку це localhost, юзер, пароль, назва бази даних та її кодування. Використовується кодування utf8 тому що воно є одним з найбільш відомих та стандартизованих типів кодування тексту та дозволяє зберігати символи Unicode. Нижче знаходится фрагмет налаштування, де відбувається підключення до бази даних.

```
    'orm_default' => [
        'driverClass' => PDOMySqlDriver::class,
        'params' => [
            'host'     => '127.0.0.1',
            'user'     => 'root',
            'password' => '',
            'dbname'   => 'database',
            'charset'  => 'utf8',
        ]
    ],
```

Приклад міграції зображено нижче, вона містить лише запит до бази даних (addSql) та вираз, що містить інформацію при якій умові не потрібно її проводити (abortIf):

```
public function up(Schema $schema)
{
    $this->abortIf(…);
```

```
    $this->addSql(…);
}
```

В кожної сутності було використано анотації, для того щоб поєднати її з полями, які є в базі даних, наприклад, анотації для класу, що пов’язує її з таблицею та репозиторієм до цією сутності:

```
/**
 * Department
 *
 * @ORM\Table(name="department")
 * @ORM\Entity(repositoryClass="Model\Repository\Department")
 */
```

Більшість репозиторіїв мають схожі методи для збереження, видалення, редагування та пошуку. Одною з перевагою Doctrine є те, що можна звертатися до бази даних за допомогою методів. Також це робить запит більш стандартизованим, що позитивно впливає на процес роботи, якщо на проекті працюють більше ніж одна людина. Приклад запиту на пошук по полю зображено нижче.

```
$results = $this->getEM()->createQueryBuilder()
        ->select("p.".$field['title'].' as
title',"p.".$field['content'].' as content','p.route')
        ->from("Model\Entity\\{$class}","p")
        ->where("p.".$field['title']." LIKE :str")
        ->orWhere("p.".$field['content']." LIKE :str")
        ->andWhere("p.status = :status")
        ->setParameter("status", "publish")
        ->setParameter("str", "%".$string."%")
        ->getQuery()
        ->getArrayResult();
```

Як видно з коду більшість методів схожі на звичайний sql (select, from, where), потім в кінці метод getQuery() формує запит, а getArrayResult() віддає результат у вигляді масиву.

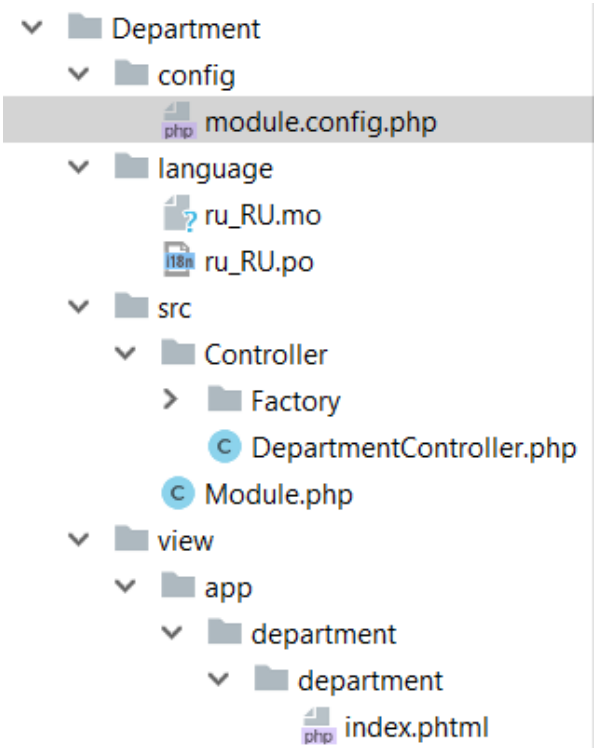

Рисунок 4.6 – Приклад змісту папки

Кожну сторінку сайту або сутність було покладено у різні папки для того, щоб можна було легше орієнтуватися у додатку. На рисунку 4.6 зображено структуру кожної з папок на прикладі кафедри.

У кожній директорії є папка з конфігурацією, мовами, на які треба перекласти систему, є папка для технічної частини, де відбуваються усі основній дії та .phtml файли, які використовуються для відображення усієї інформації в системі. Також є файл Module.php де розміщений шлях до файлу з конфігурацією. Мову можна змінювати у файлах, але для більшої комфортності можна використовувати сторонню програму. При розробці було використано програму Poedit (рисунок 4.7) і у неї був увесь необхідний функціонал. В програмному коді записуються усі слова на англійській мові, потім програма їх усіх вибирає та виводить у колонку зліва, а справа вже необхідно записати відповідник на тій мові, на яку треба перекласти. Кількість мов, на яку можна перекласти застосунок, необмежений.

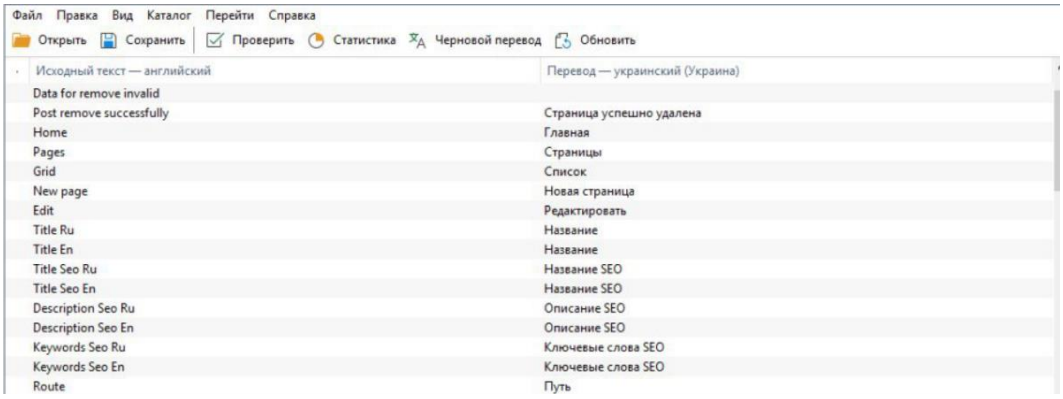

Рисунок 4.7 – Програма Poedit

У конфігурації кожної частини вказуються шляхи до контролерів, фабрики, конфігурації перекладача та представлення. Для більшого розуміння у наступних поясненнях, як приклад взято модуль меню сайту з адміністративної панелі. У шляхах вказується:

- назва шляху (manage/menu);
- типи взаємодії з шляхом (/manage/menu[/:action][/:id][/]), наприклад, якщо є задача, подивитися тільки один пункт меню, то передається id, а якщо потрібно зберегти (ajax-save) або відредагувати (edit), то відповідний метод передається в поле action;
- контролер (Controller\MenuController::class);
- метод, який стоїть за замовчуванням (index).

Також необхідно вказати, до якого контролера, яка фабрика відноситься (Controller\MenuController::class=>Controller\Factory\MenuControllerFactory::class).

Zend як і інші популярні фреймворки дотримується сервісної архітектури. Наприклад менеджер сутностей (EntityManager), який відповідає за доступ до бази даних можна просто додати до фабрики і потім викликати в контролері як сервіс. Була можливість додати усі сервіси до головного конфігураційного файлу додатку, але варіант з доданням сервісу в кожному модулі більш кращий, бо всі сервіси в одному файлі можуть призвести до плутанини. Тож було створено фабрику під кожний контролер.

```php
class MenuControllerFactory implements FactoryInterface
{

    public function __invoke(ContainerInterface $container, $requestedName, array $options = null)
    {
        $entityManager = $container->get('doctrine.entitymanager.orm_default');
        $translator = $container->get('translator');

        return new MenuController($entityManager, $translator);
    }
}
```

Рисунок 4.8 – Фабрика MenuControllerFactory

На рисунку 4.8 предсталено приклад фабрики, для того щоб правильно її сконфігурувати, потрібно імплементувати интерфейс FactoryInterface. Клас містить лише один метод __invoke(), який відноситься до спеціальних методів PHP. Метод __invoke() спрацьовує, коли звертатися до об’єкта як до функції. В середині методу дістаємо менеджер сутностей та перекладач. Після чого створюємо екземпляр класу контролера. Однією з переваг Zend Framework є те, що в нього є свій зручний перекладач. Якщо було встановлений перекладач, але не додали слово у словник, то він просто видасть повідомлення без перекладу.

```php
public function indexAction()
{
    $menus = $this->getEM()->getRepository(Menu::class)->findAll();

    return (new ViewModel())->setVariables([
        'breadcumps'=>$this->getBreadcumps(),
        'menus'=>$menus,
    ]);
}
```

Рисунок 4.9 – Метод вибірки всіх елементів меню

На рисунку 4.9, для того, щоб вибрати всі дані з меню, в змінну $menu записуються всі данні з репозиторіями та в новий екземпляр класу ViewModel

передаємо її та пункти навігаційного меню. Пункти меню формуються у кожного контролера окремо в методі getBreadcumps().

```
<?php if (!empty($modules['department'])): ?>
    <select id="module_entit" name="module_entity_department" class="form-control" style="...">
        <option value="0"><?=$this->translate("- Pick -")?></option>
        <?php foreach ($modules['department'] as $item): ?>
            <option data-title-ru="<?=$item['title_en']?>" value="<?=$item['route']?>"><?=$item['title_ru']?></option>
        <?php endforeach; ?>
    </select>
<?php endif; ?>
```

Рисунок 4.10 – Приклад коду представлення

На рисунку 4.10 зображено, що представлення – це поєднання HTML та PHP. Всі дані, які було передано з контролера до ViewModel, можна дістати в цих .phtml файлах.

Після того, як необхідні пункти меню були додані, всі дані зберігаються та відправляємо їх за допомогою AJAX. Ця технологія дозволяє робити запит без перевантаження сторінки. Для відправки запиту на сервер необхідно вказати:

- шляхи, на якій нам потрібно зробити запит (/manage/menu/ajax-save);
- метод запиту (POST);
- всі дані, які зібрано і які хочемо зберегти;
- метод, в якому можна прописати поведінку, що треба робити перед відправкою даних. В нашому випадку кнопка відключається, щоб користувач не зміг відправити ті ж самі дані знову.
- після відправки повідомлення запит може як буди відправленим так і повернути помилку. В будь-якому випадку кнопка відправки знов робиться активною та на три секунди з’являється стрічка зі статусом, яка повідомляє успішно чи ні відбулася операція.

Після цього сервер перенаправляє всі дані до метода ajaxSaveAction() і якщо запит проходить перевірку на метод та присутністьX-Requested-With зі

значенням XMLHttpRequest, то він зберігає всі передані дані до таблиці. А якщо не проходить перевірку, то віддає помилку, що така дія не знайдена.

Одним з найважливіших кроків в розробці додатку було придумати як легко та коректно будувати меню, керуючи його порядком у відображанні та рівнем вкладеності, та як зберігати його в базі даних. Було вирішено, що в таблиці menu буде повністю зберігатися структура меню. Цей варіант здався найкращим через ряд деяких причин:

– після кожного збереження, дані формуються заново з тим порядком, який вказав користувач, тож нам не потрібно піклуватися про зміну порядкового номеру;
– пункти меню в яких є рівень вкладеності містять у собі поле children в якому зберігаються підпункти;
– можна зберігати шляхи до них і не треба робити окремий запит, для їх знаходження.

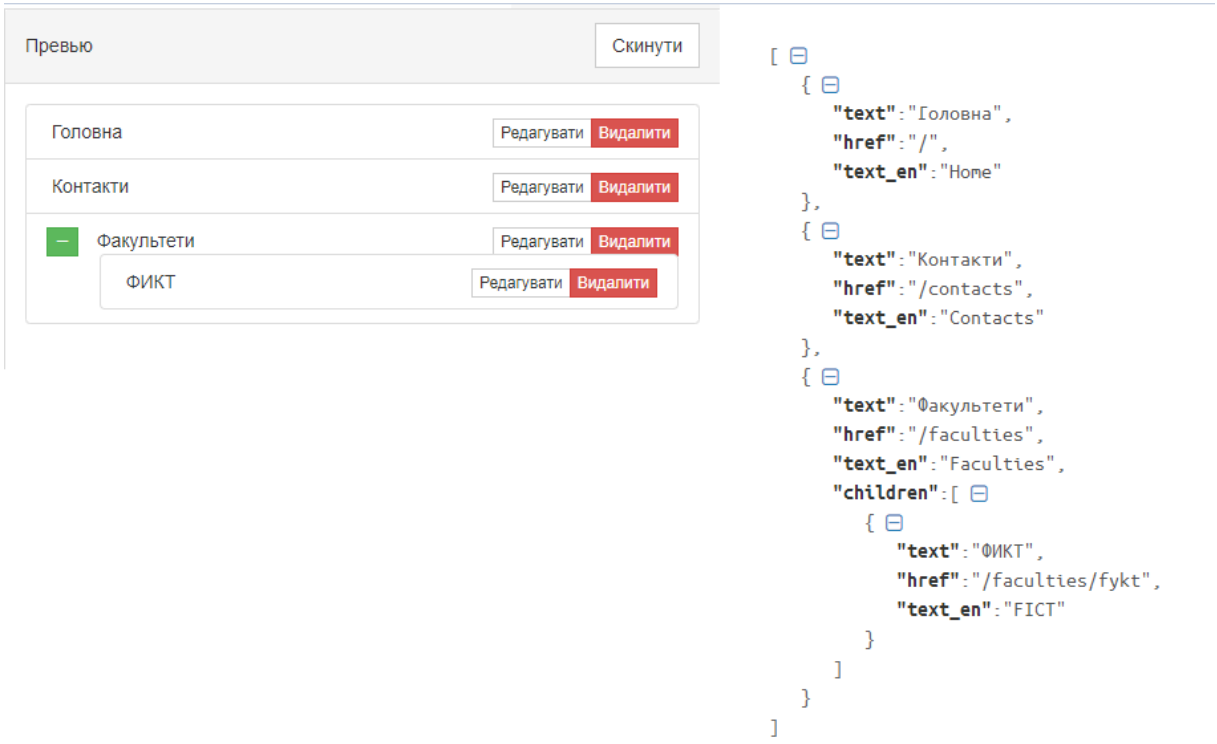

Рисунок 4.11 – Приклад структури меню

На рисунку 4.11 зображено приклад меню (зліва) та відповідний йому json об’єкт, який зберігається в базі даних (праворуч), як бачимо кожний пункт меню зберігає інформацію двома мовами про його назву, посилання на нього та вкладені списки, якщо такі є.

Наступним кроком була розробка таблиць для відображення такої інформації, як перелік новин, користувачів та інших. Рішенням цього було додавання до кожної сутності, де це вимагається, сервісу с формуванням таблиці. Оскільки в пункті меню немає потреби у таблиці, то для прикладу будемє використано підписників (subscribers). Сервіс має назву SubscribersGrid, він також має свою фабрику де використовує менеджер сутностей та перекладач.

Для того, щоб цей сервіс можна було використати було потрібно в налаштуваннях модуля додати аліас, тобто дати назву сервісу:

```
'aliases' => [
    'gridSubscribers' => Service\SubscribersGrid::class,
]
```

Оскільки нам потрібно використати його в контролері SubscribersController, тому у фабрику до цього контролера додаємо його:

```
$grid = $container->get('gridSubscribers');
```

Коли йде перехід на сторінку с підписниками то відправляється запит на сервер, де вказується сторінка, в цьому випадку перша та кількість записів на одній сторінці (50). При переході на іншу сторінку відправляється номер сторінки куди перейде користувач. Серед інших варіантів параметрів, які можна передати, це статус, з якої та по яку дату вибрати підпісників та застосувати пошук.

Після цього в SubscribersController створюємо метод ajaxGridAction(), де, після перевірки на валідність даних, вже викликаємо метод сервісу SubscribersGrid – getIndex(). В цьому методі створюється запит до бази даних у декілька етапів з різними варіантами:

– у змінну $limit записується кількість рядків на сторінці;

- потім визначається, з якого підписника треба починати відлік. Для цього потрібно помножити кількість рядків на номер сторінки та відняти кількість рядків, це і буде номер першого підписника;
- створюється порожня змінна $where, куди буде записуватися увесь запит;
- перевірка чи передали статус підписника, якщо передали, то додаємо це в запит – $where .= "s.isActive = ".$filter;
- перевірки на дати та пошук відбуваються тим же способом, що і перевірка на статус;
- для того, щоб знайти кількість всіх записів, робиться count(id) по таблиці Subscribers;
- якщо було обраний якийсь з фільтрів, то вибираємо кількість з їх урахуванням;
- всіх підписників отримуємо за тим же принципом, що й кількість, але вибираємо емейл, ip, дату створення та чи активний підписник;
- встановлюємо перший елемент (setFirstResult($offset)) та кількість записів (setMaxResults($limit));
- формуємо дані та віддаємо на представлення;

В таблиці також є дії над записом. В даному випадку, це активування та видалення. На ці запити просто відправляється POST запит з унікальним ідентифікатором запису, після чого, для активування, просто змінюється статус або видаляється сутність якщо необхідне видалення.

Для форм було використано стандартні рішення від Zend. Щоб створити таку форму, необхідно створити клас, який успадкується від \Zend\Form\Form. Далі, в конструкторі передаємо назву форми до конструктора батьківського класу та всі необхідні поля форми записати в $this->add() (рисунок 5.10). Їх всього лише чотири:

- name відповідає за назву поля;

– type вказує до якого типу належить поле в формі (text, password, number, select);
– attributes визначає список атрибутів, які будуть розміщені в полі;
– options – масив опцій для елемента, оскільки в нашому випадку тип поля є випадаючий список, то можна вказати, які пункти будуть у цьому списку.

```
$this->add([
    'name' => 'grid-filter-status',
    'type' => 'select',
    'attributes' => [
        'class'                    => 'form-control input-sm',
        'data-grid-filter'         => "status",
        'data-grid-filter-name' => "managePostsIndex",
    ],
    'options' => [
        'value_options' => [
            ""          => _(" - Filter: Status - "),
            "1"    => _("Active"),
            "-1"       => _("Not active"),
        ],
    ],
]);
```

Рисунок 4.12 – Приклад поля вибору статусу підписника

Для того, щоб використати цю форму у представленні, потрібно звернутися до неї наступним чином: $this->formElement($this->form->get('name')), де name – це назва поля (grid-filter-status).

Наступною задачею було розробити експорт підписників до excel файлу. Цей процес можна умовно розділити на два кроки: вибрати усію необхідну інформацію та розмістити всі дані в excel файлі. Після того, як користувач ввів дати в якому проміжку часу необхідно шукати данні, будується запит до бази даних, де вказуються ці межі. Після отримання всіх даних йде передача їх на розміщення до файлів.

Для роботи з excel було використано бібліотеку phpexcel через її достатній функціонал та популярність. Першим кроком беремо шаблон файлу,

який лежить в папці public. Активним листком відзначаємо перший з них. На другій стрічці розміщується інформація про період репорту:

```
$sheet->setCellValue("C2","За період:".$data['date']);
```

Метод setCellValue відповідає за вставку стрічки в excel файл. Він приймає два параметри, з яких перший – це колонка та номер рядку(C2), а другий – це дані, які необхідно туди вставити. Після цього встановлюється позиція на четвертий рядок та, за допомогою циклу і методу setCellValue, вставляємо дані з нашого масиву підписників до файлу, кожного разу збільшуючи номер рядку на один. Далі зберігаємо інформацію про перший листок та переходимо до етапу збереження файлу.

Для того, щоб зберегти файл, спочатку формуємо його ім’я за допомогою дати створення та дати в форматі timestamp на момент створення. Потім йде перевірка, чи існує такий файл, і якщо ні, то створюємо його і записуємо туди все, що було згенеровано для файлу.

Користувачу віддається посилання через яке він може завантажити цей файл собі на пристрій.

Для завантаження файлів на сервер було обрано бібліотеку Dropzone.JS. Це відкрита бібліотека, яка надає інтерфейс для того, щоб перетягувати файли для завантаження. Цей спосіб є одним з найпопулярніших, а бібліотека є легка у використанні, тому не дивно, що було обрано її.

На сторінці використовуємо форму, завдяки Dropzone звертаємося до цієї форми:

```
$("form#my-awesome-dropzone").dropzone()
```

В метод dropzone необхідно передати шлях, на який необхідно передати файл та що необхідно зробити після цього, в нашому випадку це перезавантажити місця для відображення. Після перевірки на POST запит та його заголовки перевіряється зображення:

– перевірка, щоб висота та ширина зображення були менші за 3080 і 4920 відповідно, оскільки файли більшого розміру будуть навантажувати систему;
– перевірка на кількість байт, щоб зменшити ризики, що в картинці будуть небажані дані;
– перевірка на MIME тип файлу, який вказано в масиві, які є найбільш поширені (jpeg, png, ico).

Після того, як пройдено всі перевірки, файл зберігається до папки uploads. В базі зберігається назва файлу, шлях до файлу, mime тип та дату, коли зображення було додано. Якщо одна з перевірок не була пройдена, то користувачу буде видано повідомлення про те, що формат картинки не задовольняє умови.

Окремим модулем в додатку є Mail, через який можна відправляти листи. Коли користувач відправляє форму зворотного зв'язку то на email адміністратора відправляється лист з усією інформацією. Перед тим, як приступати до розробки, спочатку було сконфігуровано smtp (рисунок 5.13).

```
'smtp' => [
    'name'               => 'yandex',
    'host'               => 'smtp.yandex.ru',
    'port'               => '587',
    'connection_class'   => 'login',
    'connection_config' => [
            'username'   => 'info@gmaol.com',
            'password'   => '14asdedeQq$',
            'ssl' => 'ssl',
    ],
],
```

Рисунок 4.13 – Конфігурація smtp

Перелік налаштуваннь:

– name – назва smtp хоста;

- host – хост или ip адреса;
- port – порт, який слухає хост;
- connection class – назва класу чи аліас для SmtpLoader;
- connection_config – масив параметрів, де вказується логін, пароль та тим з'єднання.

Для того, щоб надіслати лист, необхідно передати на сервіс Mail email адміністратора, шаблон для листа (лист до адміністратора або лист з новиною), дату, яку необхідно передати, та заголовок листа. Остаточне формування та відправка листа відбувається у файлі Sender.php. У ньому вказується, з якої адреси та на яку відбудеться доставка, тема листа, кодування тексту, основна частина разом з шаблоном та заголовки, в нашому випадку, це multipart/alternative. Далі створюється екземпляр класу Smtp, де додається до нього опції і відправляємо лист.

Якщо для адміністративної панелі використовується сітка, то на сайті такий вибір не є задовільним через дизайн. Тому для цієї задачі було обрано використати стандартне рішення від Zend.

Для того, щоб використати Zend Paginator потрібно використати адаптер, тому що пагінатор від Zend нічого не знає про модель даних. Для цього було обрано використати DoctrinePaginator, що буде як міст між ORM та Zend Paginator.

Тож, коли відбувається перехід на сторінку новин спрацьовує метод mainAction(), в якому виконуються наступні кроки:

- якщо зі сторінки передається номер сторінки, то береться він, а якщо ні, то береться одиниця;
- виконується попередній крок, але для тегів;
- якщо теги є, то вибирається запит на всі новини з цим тегом, а якщо ні, то запит на усі новини;
- цей запит передається на DoctrinePaginator, створивши його екземпляр;

– завдяки цьому адаптеру створюється Zend Paginator;
– для пагінатора встановлюються налаштування, такі як кількість сторінок на сторінці (6) та номер активної сторінки.

Всі отримані данні передаються у представлення, де формується список новин та кнопки переходу на інші сторінки.

Наступним етапом у розробці було створення автентифікації користувача, блок схему до якої знаходиться в додатку В. Цьому було приділено багато уваги, оскільки через неї можна отримати дані університету. Для форми було обрано додаткові параметри, – такі як:

– StringTrim – фільтр, який прибирає пробіли з початку та кінця тексту, оскільки їх зазвичай ставлять ненароком;
– StringLength – валідатор в якому можна задати мінімальну та максимальну довжину сторінки;
– EmailAddress – валідатор, який перевіряє, що стрічка є емейлом;
– UserExistsValidator – валідатор, який перевіряє, чи існує такий користувач.

Останній з валідаторів є самописний, коли на нього передається значення, то спочатку він перевіряє, чи ця величина є скалярним типом. Якщо це так, то він намагається знайти такого користувача за полем email у репозиторії і якщо знаходить, то перевірка пройдена, а якщо ні, то видає помилку, що такого користувача не існує.

Якщо користувач, наприклад, був на сторінці створення новини (/manage/news/create) або йому надали таке посилання, то його перенаправлять на сторінку логіну, але щоб не втратити цю url, вона додається у GET параметр (риссунок 4.14). Звісно, є перевірка, що довжина такого посилання була менша за 2048 символів.

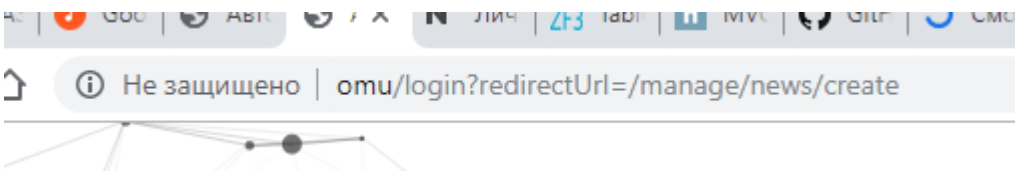

Рисунок 4.14 – Строка пошуку

Першим кроком є створення екземпляру класу LoginForm, якому одразу записуємо значення посилання для редіректу. Після цього йде перевірка чи був це POST запит і якщо це так, то записуємо всі данні з форми властивості класу Form - $data. Слід зазначити, що LoginForm наслідується від Form. Коли отримано всі данні, то йде перевірка за допомого фільтрів та валідаторів, що є у форми. Якщо всі перевірки пройшли вдало, то відправляємо емейл, логін та позначку запам’ятати себе до сервісу AuthManager, який за допомого адаптера перевіряє, чи є співпадіння з таким емейлом та паролем в базі. Після чого, якщо такий користувач знайдений, то він його логує, а якщо клієнт хотів, щоб його запам’ятали то створюється сессія на один місяць. Єдина помилка, яка могла трапитися на цьому етапі то це той випадок, якщо користувач зміг потрапити на цю форму, коли він вже був у системі, але такий варіант оброблюється. І останній крок це те, що, якщо такого користувача не знайдено, то спливе повідомлення про це, а якщо є, – то його буде направлено на посилання з якого його перекинуло на форму автентифікації. У випадку якщо він одразу зайшов на форму, то його буде перенаправлено на головну сторінку адміністративної панелі (/manage).

Однією з задач було передбачити випадок, коли у співробітників немає змоги, певний проміжок часу, публікувати новини. На допомогу приходить система виконання періодичних команд або, як її ще називають – Cron.

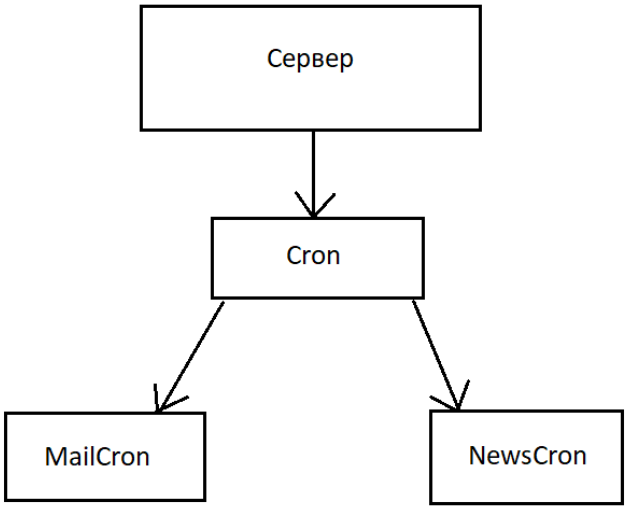

Рисунок 4.15 – Схема роботи Cron

На рисунку 4.15 зображено, як ця система працює. Сервер періодично посилає команди на головний Cron, а він вже вирішує, до якого модуля відноситься цей запит. Cron є окремим модулем в системі і виступає як точка входу усіх консольних команд, а вже в середині делегує дані на різні модулі системи. Діаграма послідовності Crone зображена в додатку Г.

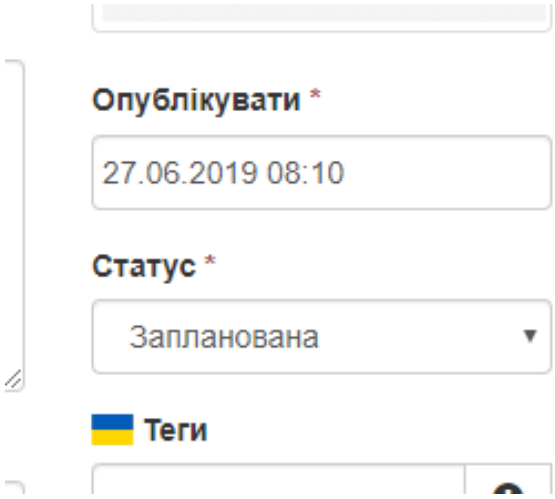

Рисунок 4.16 – Вибір дати публікації

На рисунку 4.16 представлено, як при виборі користувача з’являється новий елемент, у якому можна вказати дату та час публікації новини. На

сервері створюється консольна команда, яка кожних 10 хвилин направляє запит на систему. Коли застосунок отримає сигнал команди, то він починає шукати всі новини зі статусом “planning”. Потім, якщо дата, коли новина повинна бути опублікована, менша за теперішню дату, то застосунок перевіряє наступну новину, а якщо все-таки більша, то новина переходить у статус “publish” після чого вона публікується на сайті.

Схожий принцип дії і у автоматичної відправки новин, наприклад, раз на день запускається метод sendMainNews, у якому виконується наступне:

- спочатку вибираються усі активні підписники;
- вибираються новини за останню добу, які вже були опубліковані;
- корегування розмірів картинок новин під стандарти листа;
- відправлення новини підписникам;

Висноки до розділу 4

У даному розділі було описано процес розробки інформаційної системи сайту університету. Було описано всі етапи від розробки бази даних до реалізації, в кожному з яких розглядаються основні проблеми та шляхи їх вирішення.

Результатом розробки стала успішно запрограмована система, яка відповідає усім вимогам описаним у завданні. Використання основних та загальноприйнятих підходів до розробки та слідування принципів фреймворку Zend, забезпечило розробку з мінімальною кількість проблем. Усі елементи стеку технологій, що було обрано для використання справилися з завданнями, які були на них покладені.

5. ТЕСТУВАННЯ

Кожний програмний продукт, перед тим як вважається завершеним, повинен бути протестований. Тестування системи – це процес оцінки, головним призначенням якого є виявлення помилок та дефектів у роботі системи[7]. Але не потрібно фокусуватися саме на помилках системи, потрібно також провести перевірку на відповідність до завдання, яке стояло перед розробником.

5.1 Функціональне тестування

Основною метою функціонального тестування є перевірка, чи дотримано усіх вимог до продукту, які закладалися на етапі проектування.

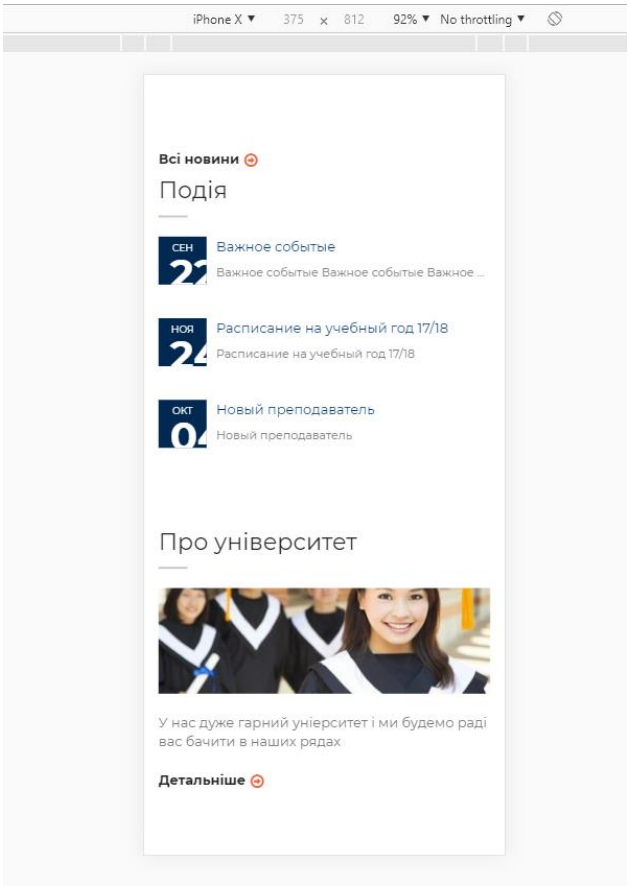

Рисунок 5.1 – Адаптивність сайту

На рисунку 5.1 зображено, як сайт веде себе при зміні розміру екрану. Для перевірки було використано стандартну можливість адаптації екрану браузера Google Chrome. Сайт перевірений у всіх найпоширеніших браузерах, такі як Firefox, Opera та Chrome. Час переходу на іншу сторінку складає менше двох секунд, що цілком задовольняє умовам.

Рисунок 5.2 – Запис в строку пошуку скрипта на мові JavaScript

Однією з найголовніших перевірок є перевірки безпеки. На рисунку 5.2 наводиться приклад, як в строку пошуку по новинам було записано скрипт - <script>alert(1)</script>, при умові, що застосунок не є безпечним, то з’явилося б модальне вікно з написом «1», а оскільки цього не трапилося, а просто відбувся запит на пошук, то можна вважати, що перевірка пройдена. Цей тест був пройдений на всіх елементах сайту та адміністративної панелі, де користувач сам вводить дані.

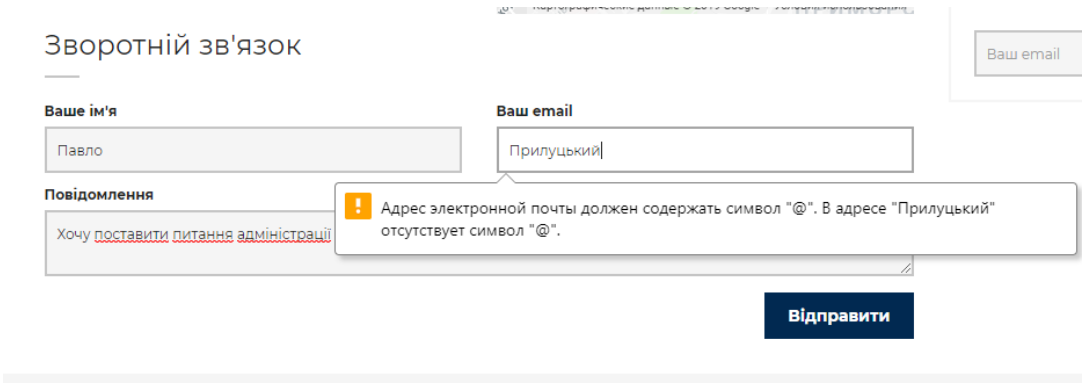

Рисунок 5.3 – Форма зворотного зв’язку

Попередня перевірка була спрямована на визначення того, як застосунок поведе себе у разі, коли процесу його роботи чи безпеці зберігання даних загрожуватиме зловмисник. Але не варто забувати, що користувач може просто допустити помилку, наприклад, при вводі свого email. Тож на рисунку 5.3 зображено, як веде себе сайт, якщо користувач ввів неправильний email, схоже повідомлення також спливає, якщо користувач не заповнив одне з полів.

5.2 Деструктивне тестування

Цей вид тестування має за мету перевірити, чи система може вийти з ладу при неочікуваних факторах.

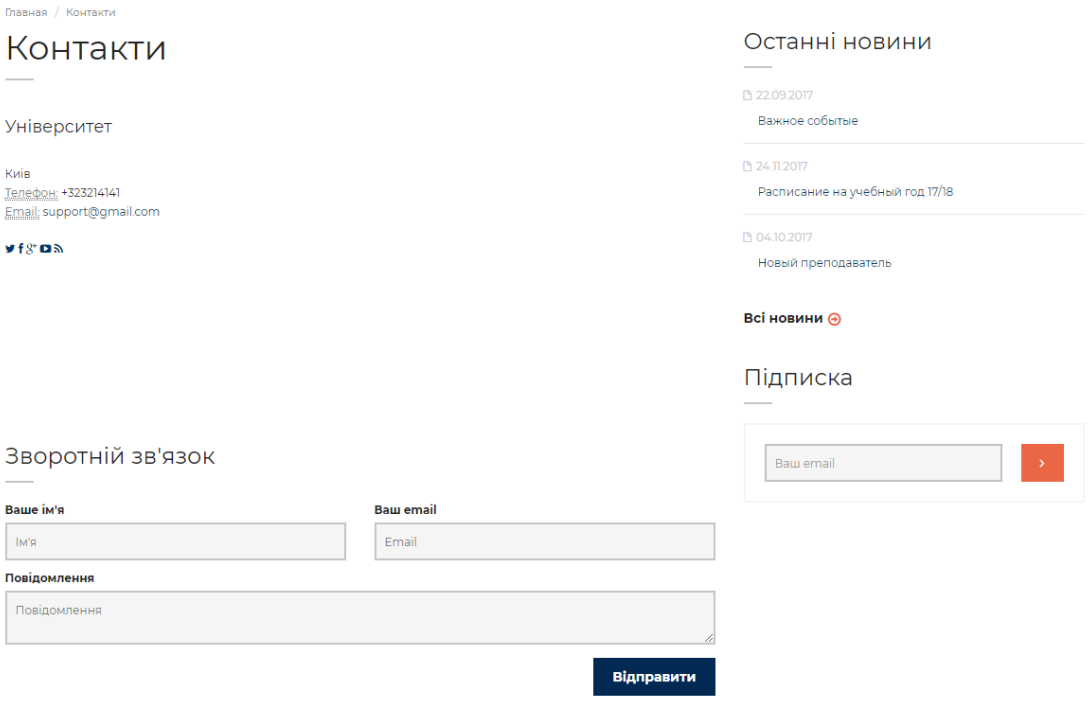

Рисунок 5.4 – Сторінка контактів із заблокованою можливість використання JavaScript

В наш час майже всі браузери не заблоковують JavaScript у своїх рішеннях, хоч і залишають таку можливість користувачу. Користувач може

його вимкнути через такі причини, як безпека даних та при наявності поганого підключення до інтернету, щоб не витрачати трафік на непотрібні візуальні ефекти. На рисунку 5.4 представлено, як веде себе сторінка контактів при заблокованому JavaScript. Особливо добре веде себе у цій ситуації форма логіна, оскільки в інших формах використовується AJAX, який працює за допомогою JavaScript, а форма логіна використовує звичайну форму, тому користувач може без труднощів зайти до системи.

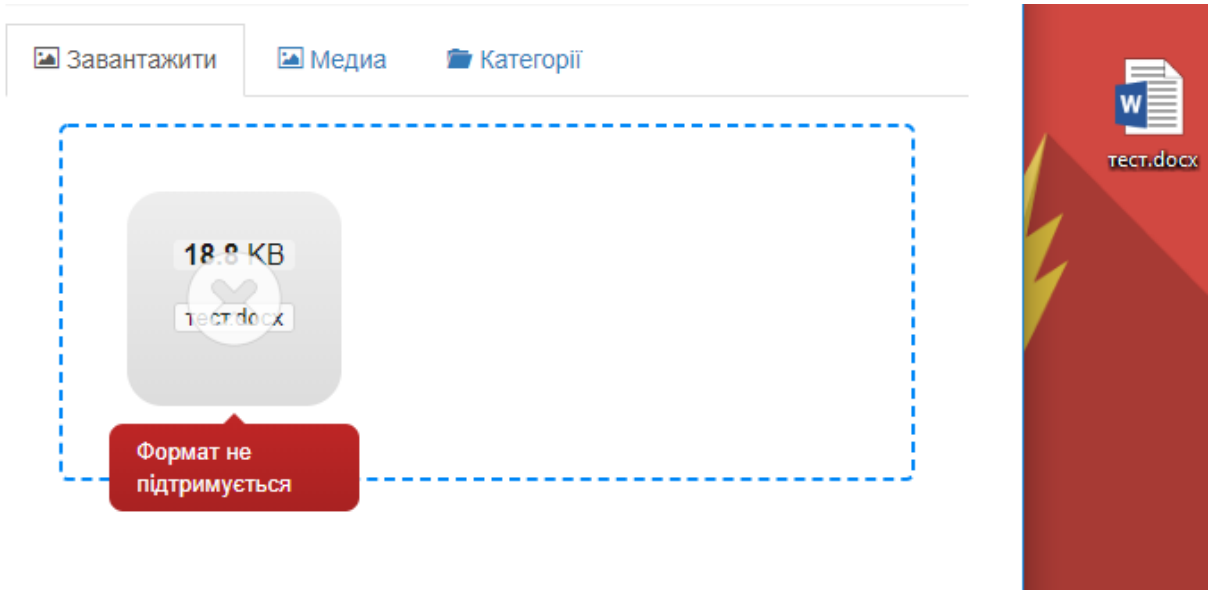

Рисунок 5.5 – Спроба завантаження файлу неправильного формату

На рисунку 5.5 зображено приклад того, як повела себе система при спробі завантажити файл недозволеного формату. Це є одна з найважливіших перевірок, адже неправильний формат може призвести до некоректного відображення інформації.

5.3 Тестування зручності

Оскільки однієї з головних вимог при розробці системи було те, що вона повинна бути продумана с точки зору UX, то не можна було не згадати цей вид тестування. Для цього було залучено сторонніх тестувальників, оскільки

розробник не може адекватно оцінити цей показник. Тестувальники були обрані серед людей різного віку та різних професій для максимально точної оцінки системи. Результати тестів були позитивні, але серед людей, чия професія не пов'язана з використанням комп'ютера, певний час були труднощі, шо і не є дивним.

5.4 Модульне тестування

Виходячи з назви цього тестування, можна зрозуміти, що воно служить для перевірки окремих модулів системи, а саме програмного коду. Основна мета цього тестування – перевірити, що модуль поведе себе саме так, як і розраховує розробник. Цей метод є одним з найпопулярніших через такі переваги:

- перевірка, що окремі частини проекту працюють правильно і не треба шукати помилку по всьому проекту;
- не слід турбуватися про рефакторинг, оскільки після нього можна запустити тести і перевірити їх працездатність;
- тести можуть пояснити, як має працювати код, тобто можна вважати їх своєрідною документацією.

Для цього тестування було обрано PHPUnit.

```
public function testIndexActionCanBeAccessed()
{
    $this->dispatch( url: '/manage', method: 'GET');
    $this->assertResponseStatusCode( code: 200);
    $this->assertControllerName( controller: UserController::class);
    $this->assertControllerClass( controller: 'UserController');
    $this->assertMatchedRouteName( route: 'login');
}
```

Рисунок 5.6 – PHPUnit тест

На рисунку 5.6 представлено приклад PHPUnit тесту. На ньому імітується випадок, коли користувач переходить на шлях “/manage”. Після цього, йде перевірка, що йому прийшла відповідь з 200-м код, що означає що запит пройшов вдало. Наступні дві перевірки відповідають за те, що при цій дій спрацював правильний контролер. І остання перевірка стежить чи спрацював коректний шлях застосунка.

Висновки до розділу 5

Система була протестована усіма необхідними та найпопулярнішими методами для того, щоб стверджувати, що вона є працездатною. Було дотримано усіх вимог, які були висунуті при постановці задачі. Система є зрозумілою для більшості користувачів різного віку та професій і правильно веде себе у нештатних випадках, таких як відключення JavaScript та спроби додати файл з некоректним форматом.

6. ІНСТРУКЦІЯ КОРИСТУВАЧА

6.1 Клієнтська частина

Спочатку розглядається саме інструкція користувача з боку клієнта, а в даному випадку, абітурієнта який хоче ознайомитися з інформацією на сайті.

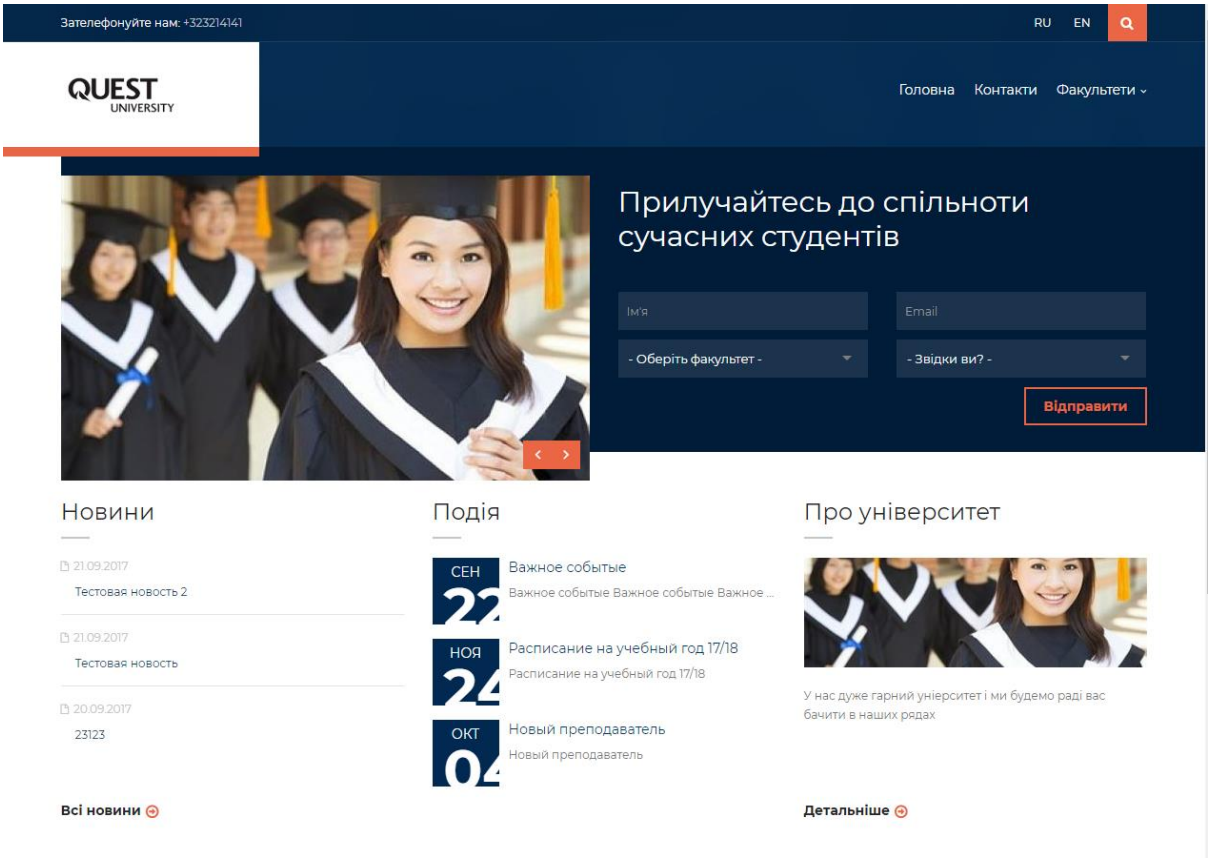

Рисунок 6.1 – Головна сторінка сайту

Головна сторінка (рисунок 6.1) сайту зустрічає нас мінімальною інформацією, що сприяє покращенню взаємодії з користувачем. Головними взаємодіючими компонентами є:

- слайдер зображень;
- пошук по сайту;
- вибір мови контенту;
- меню сайту;

– форма зворотного зв’язку;
– перелік новин, подій, що пов’язані с життям університету, та інформація про сам навчальний заклад.

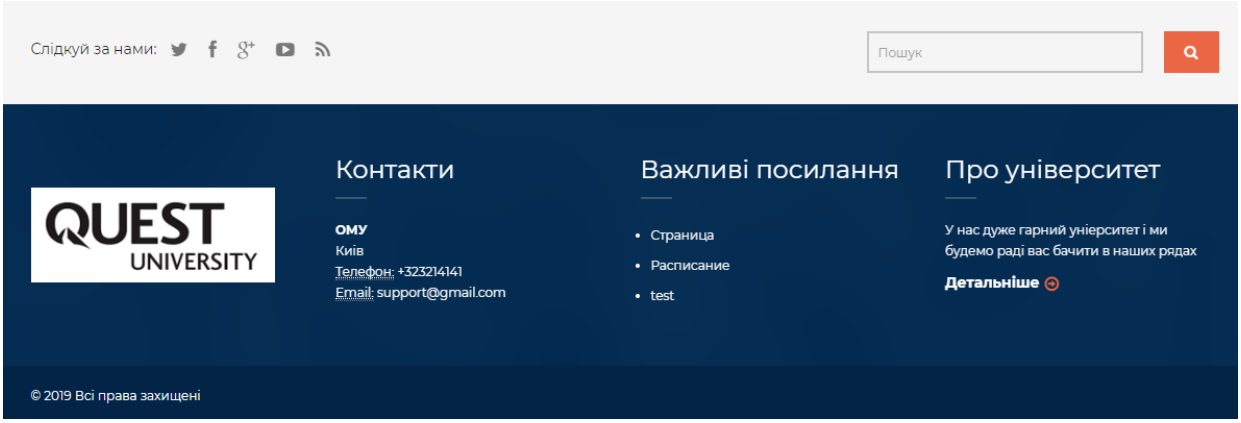

Рисунок 6.2 – Нижня частина сайту

У нижній частині сайту (рисунок 6.2) винесено якомога менше елементів, щоб не загромаджувати його:

– інформація про університет та контактні дані;
– нижнє меню;
– пошук по сайту;
– посилання на соціальні мережі.

Також серед основних особливостей сайту університету є структура його факультетів, адже факультет завжди пов’язаний з кафедрами. Перейшовши на сторінку факультету, можна ознайомитися з усією інформацією про факультет, включаючи його освітню програму, переглянути викладачів та кафедри факультету.

Також в тексті кожної новини є можливість додавати теги. Тег – це ключове слово новини, по якому можна знайти схожі статті.

Найважливішою сторінкою, після головної, є сторінка контактів (рисунок 6.3). На ній розміщена інформація для того, щоб зв’язатися з адміністрацією університету. Більшість з надписів та посилань на соціальні мережі можна

змінити в адміністративній панелі. Також користувач може знайти університет на карті, є можливість розгорнути карту на весь екран. Внизу сторінки є форма зворотного зв'язку, як на головній сторінці, але на відмінну від неї, замість країни та факультету, до якого хоче вступити абітурієнт, є можливість ввести повідомлення до адміністрації університету. Якщо користувач її використає, то після того як адміністратор це побачить, він зв'яжеться з клієнтом. Праворуч зображені останні новини університету та вікно з можливістю ввести свій email, після чого він буде збережений у базі підписників для отримання новин сайту.

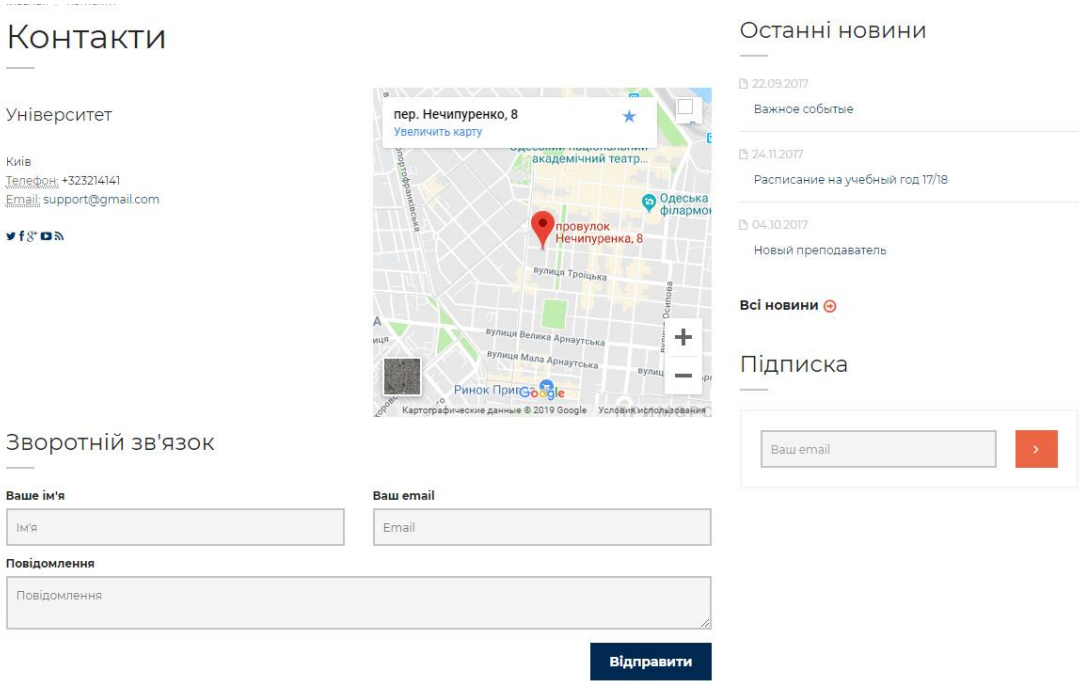

Рисунок 6.3 – Сторінка контактів

6.2 Адміністративна панель

Для того, щоб увійти до панелі необхідно спочатку авторизуватися. Реєстрація відбувається наступним чином:

– вам необхідно зв'язатися з адміністратором університету та передати йому ваш email, щоб він створив ваш обліковий запис;

– потім він повинен створити ваш профіль в системі;
– після цього він наддасть вам пароль, за допомогою якого можна увійти до застосунку.

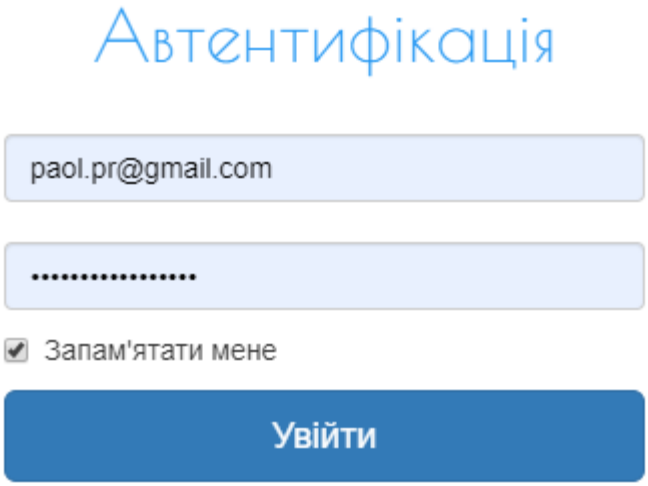

Рисунок 6.4 – Форма входу

На рисунку 6.4 представлена форма входу до адміністративної панелі, де, окрім полів email та пароля, можна поставити галочку на полі «Запам’ятати мене». Якщо ви обираєте це поле, то система створить сессію, термін дії якої – один місяць.

Після того, як ви увійшли в систему, то перед вами відкривається сторінка (рисунок 6.5), на якій в правому верхньому куті є кнопка для виходу з системи та меню ліворуч, де кожна вкладка відповідає за певний функціонал на сайті:

1) меню – у цьому пункті можна повністю змінювати верхнє та нижнє меню, як показано на рисунку 6.5, є можливість робити вкладені списки та змінювати порядок вкладок. Можна переглянути url для обраного варіанту, обрати спосіб відкриття сторінки (наприклад, можна відкрити сторінку у новому вікні) та є можливість обрати текст,

який буде показано, коли користувач наведе курсором на назву пункту меню;

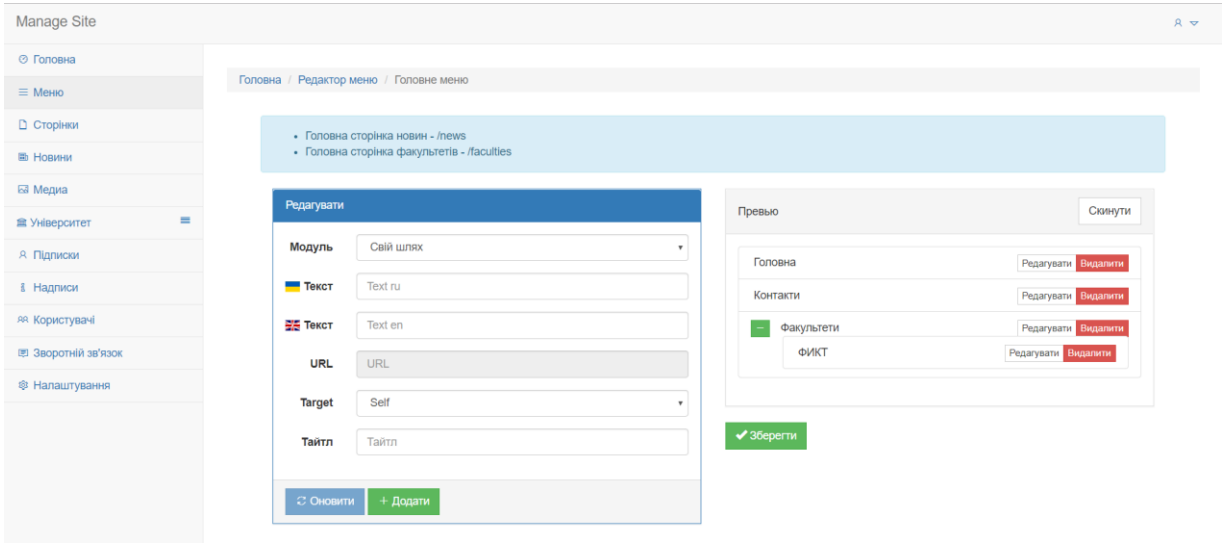

Рисунок 6.5 – Адміністративна панель

2) сторінки – містить інформацію та всі сторінки сайту. Є можливості їх переглядати, редагувати, створювати та видаляти. Якщо новин буде багато, то є фільтр по автору, даті та статусу;
3) новини – є схожими за функціоналом на сторінки, але мають деякі відмінності, такі як тип (новини чи подія) та відмінності при створенні новин;
4) медіа – тут зберігаються всі медіа файли з сайту для того, щоб була можливість використовувати їх повторно для декількох новин, адже якщо кожного разу додавати один і той самий файл то система буде забиватися непотрібними файлами, що буде збільшувати навантаження на сервер на якому розташований застосунок і в майбутньому це буде негативно впливати на швидкість та працездатність роботи сайту. Також є базові категорій і можливість створити нові. Вони відповідають за медіа файли в різних частинах сайту, наприклад, слайдер;

5) університет – містить у собі інформацію про факультети, кафедри та викладачів;
6) підписки – містять email-и користувачів, які вирішили підписатися на новини сайту;

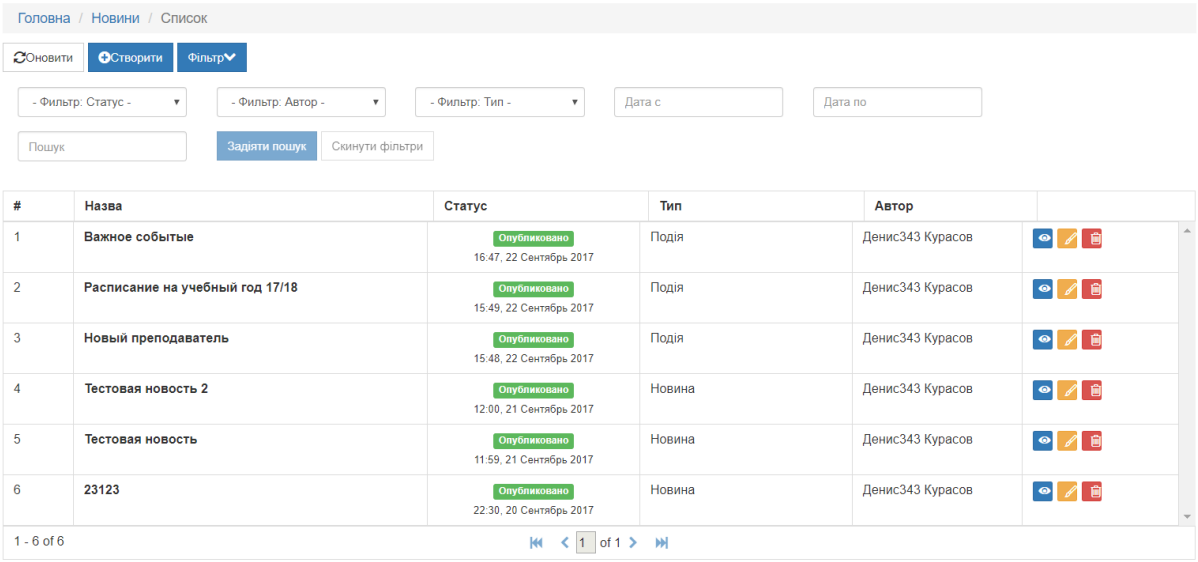

Рисунок 6.6 – Приклад списку

7) надписи – місце, де адміністратор може змінити ключові написи на сайті, такі як контактний телефон, інформацію про університет, електронну скриньку для зв’язку та багато іншого. Всі надписи дублюються обома мовами;
8) користувачі – члени адміністрації університету. Тут можна створити нового або відключити старого користувача;
9) зворотній зв'язок – зберігає інформацію про користувачів, які скористалися формою зворотного зв’язку на сайті;
10) налаштування – місце, де можна вказувати посилання на соціальні мережі.

На рисунку 6.6 зображено один із прикладів списку вкладок. Окрім перегляду списку, можна переглянути новину окремо, відредагувати,

видалити чи створити її. Присутній фільтр, який був згаданий вище, та оновлення сторінки.

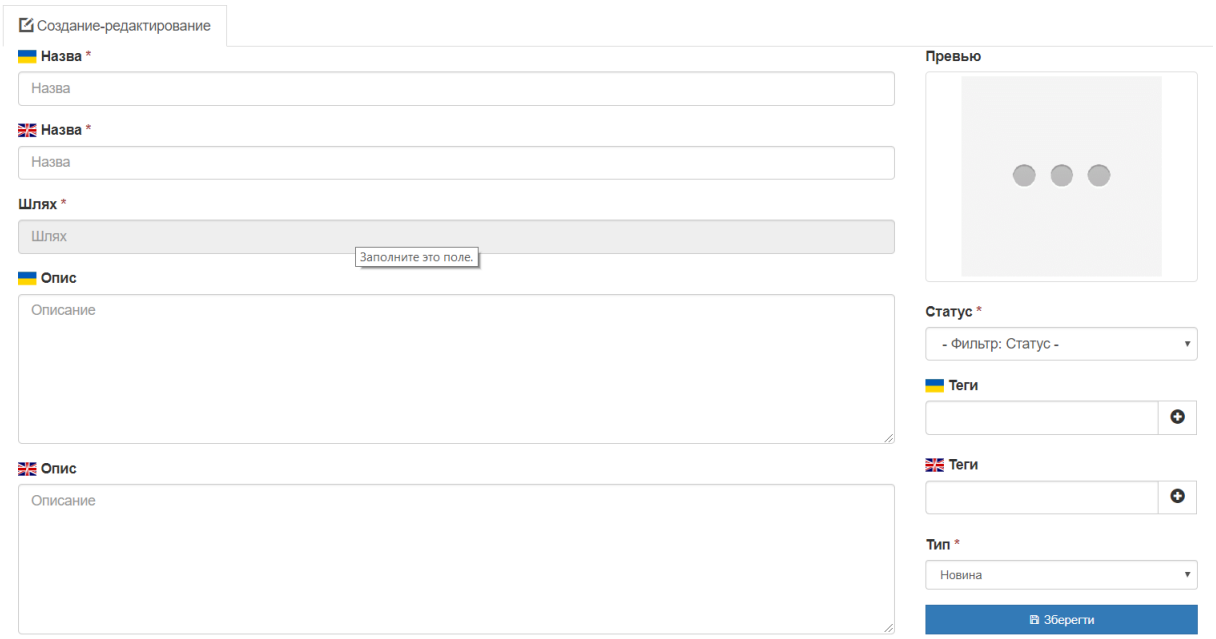

Рисунок 6.7 – Приклад редагування та створення новини

На рисунку 6.7 представлено, як створювати чи редагувати вже створені новини. По-перше, такі елементи як назва, опис, зміст та теги повинні бути написані одразу на двох мовах, адже якщо використовувати сторонній застосунок для перекладу, то він приведе до помилок, що погано відобразиться на відвідування сайту з боку іноземців. Після того, як всі поля заповнені, то з назви побудується шлях, який буде використовуватися у url сайту. Можна встановити картинку для новини та обрати тип, в нашому випадку це новина чи подія. Під основним текстом статті є підрахунок кількості символів з пробілами та без них. Також необхідно обрати, який статус повинен бути у новини. Всього є три варіанти: чернетка, опубліковано та заплановано. Якщо обрати чернетку, то новина просто збережеться на тому етапі, якому ви її залишили і потім можете повернутися знову та закінчити її. Статус опубліковано свідчить про те, що новина чи подія одразу з’являться на сайті. І коли користувач вибирає статус заплановано, то після цього

з’являється вікно де він може обрати дату та час, коли новина повинна бути опублікована, але слід врахувати, що вона буде опублікована протягом 10 хв від часу, який було встановлено.

Висновки до розділу 6

Інструкція є закінченою, бо у неї розглянуто більшість аспектів та принципів роботи системи. Всі етапи, що були розглянуті, є послідовними кроками користувача, а не окремі частини інформації, де користувач може заплутатися. Вона містить увесь необхідний графічний та текстовий матеріал, який можна дати на ознайомлення користувачам перед тим, як надавати доступ до системи.

ВИСНОВОК

У дипломному проекті була розроблена автоматизована система управління сайтом університету. У ході роботи був здійснений аналіз існуючих рішень, що дало змогу з’ясувати недоліки схожих систем та уникнути їх. На основі завдання було обрано технології для його реалізації.

Розроблено веб застосунок який відповідає всім вимогам у заданні і включає в себе сайт та адміністративну. Сайт розроблений у сучасному дизайні з усією необхідною інформацією. Застосунок отримав зрозумілий інтерфейс, як зі сторони сайту так і адміністративної панелі. Остання в свою чергу дозволяє змінювати контент на сайті не залучаючи технічних спеціалістів та наділена широким функціоналом для комфортної роботи. Було впроваджено усі можливі заходи безпеки як сайту так і адміністративної панелі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Новосибірський університет [Електронний ресурс]. – Режим доступу до ресурсу: https://www.nsu.ru/n/.
2. Київський національний університет імені Тараса Шевченка [Електронний ресурс]. – Режим доступу до ресурсу: http://www.univ.kiev.ua.
3. Харківський національний університет імені В. Н. Каразіна [Електронний ресурс]. – Режим доступу до ресурсу http://www.univer.kharkov.ua.
4. Авторський сайт IT-консультанта [Електронний ресурс] – Режим доступу до ресурсу: https://ivan-shamaev.ru/slow-php-on-windows-server-iis-fastcgi/
5. Using Zend Framework 3 [Електронний ресурс]. – Режим доступу до ресурсу: https://olegkrivtsov.github.io/using-zend-framework-3-book/html/index.html.
6. Stfalcon.com [Електронний ресурс]. – Режим доступу до ресурсу: https://stfalcon.com/ru/blog/post/doctrine2-ORM-architecture.
7. Wikipedia [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення

ДОДАТОК А

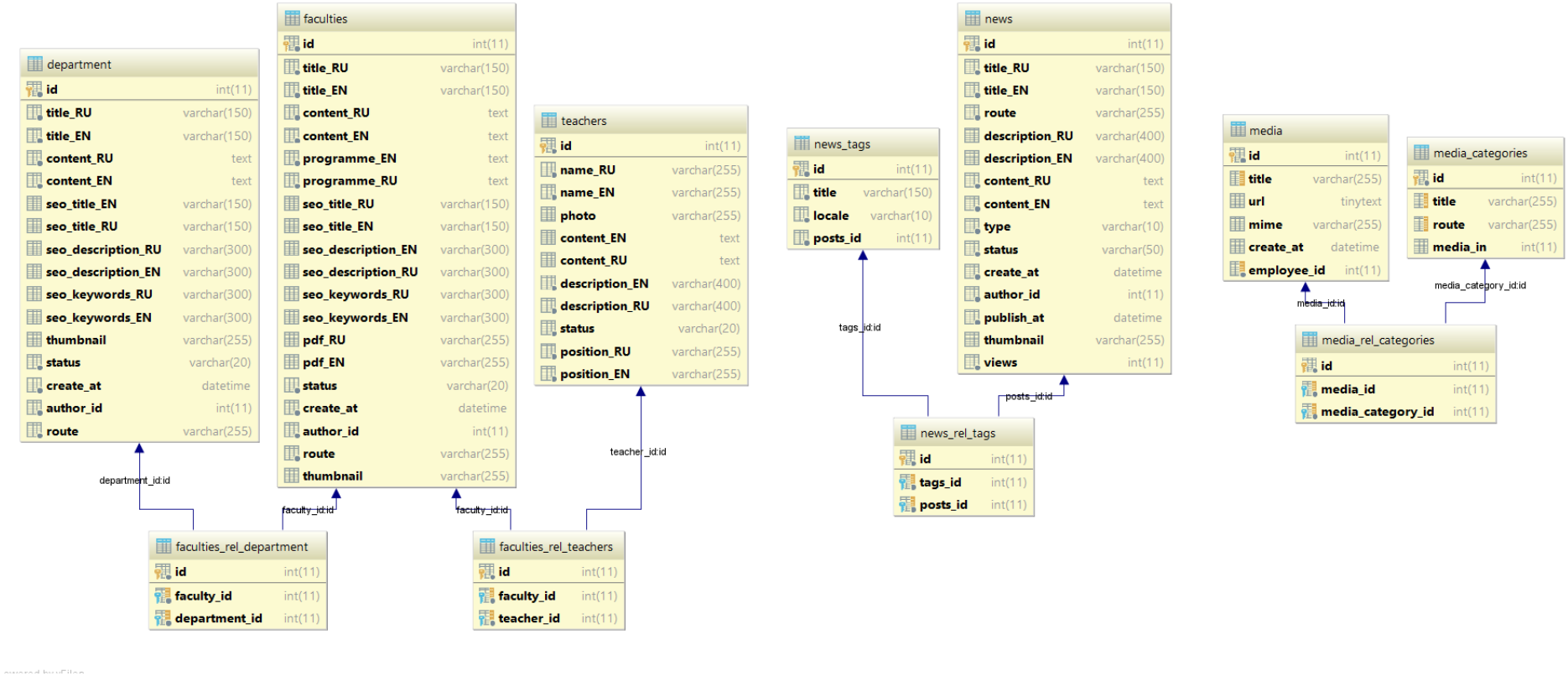

Рисунок А.1 — Структура бази даних системи

ДОДАТОК Б

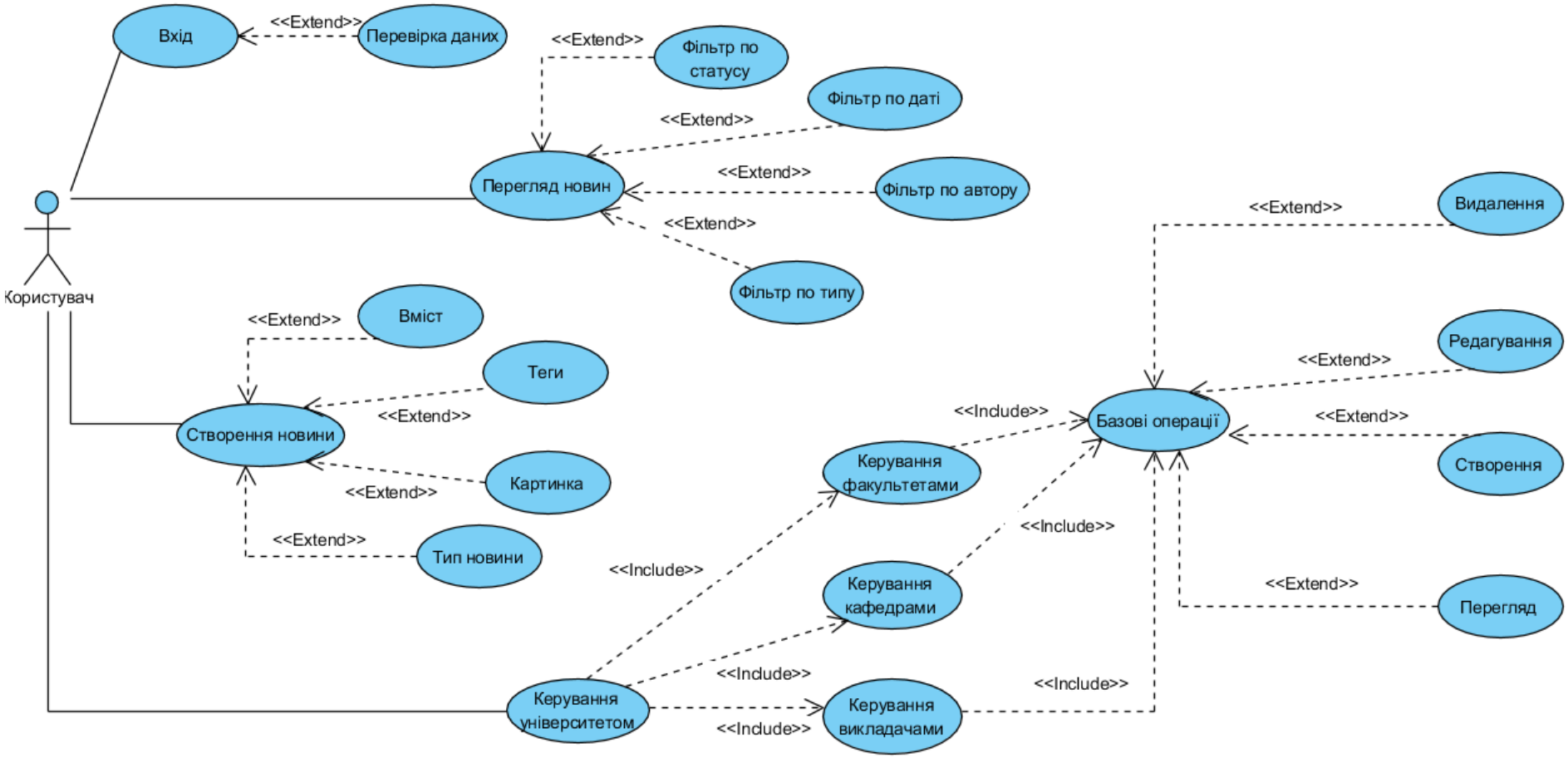

Рисунок Б.1 — Діаграма активності

ДОДАТОК В

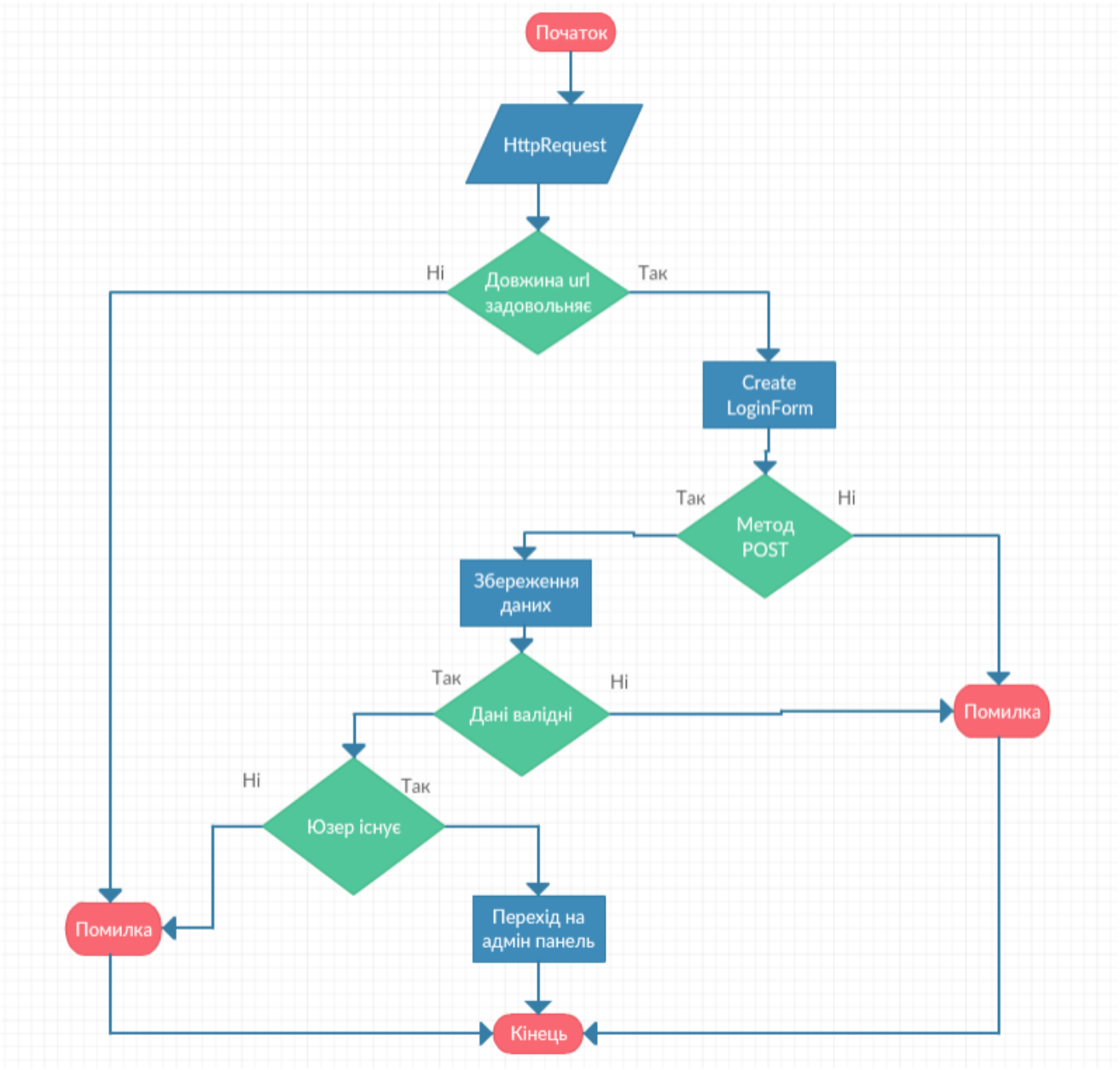

Рисунок В.1 — Блок схема автентифікації

ДОДАТОК Г

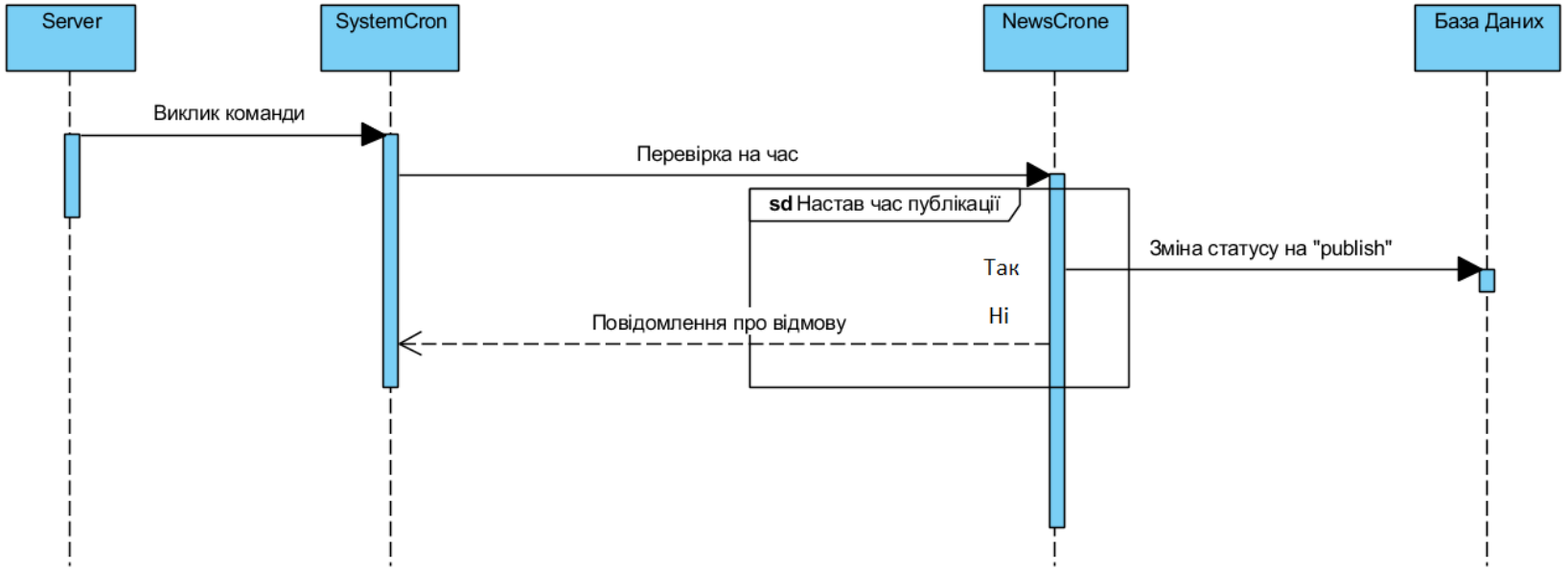

Рисунок Г.1 — UML діаграма послідовності роботи Crone

ДОДАТОК Д

Вихідний код програми

```php
<?php

namespace Model\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Department
 *
 * @ORM\Table(name="department")
 *
@ORM\Entity(repositoryClass="Model\Repository\Department")
 */
class Department
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     *
@ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="title_RU", type="string", length=150, nullable=false)
     */
    private $titleRu;

    /**
     * @var string
     *
     * @ORM\Column(name="title_EN", type="string", length=150, nullable=false)
     */
    private $titleEn;

    /**
     * @var string
     *
     *
@ORM\Column(name="content_RU", type="text", length=65535, nullable=false)
     */
    private $contentRu;

    /**
     * @var string
     *
     *
@ORM\Column(name="content_EN", type="text", length=65535, nullable=false)
     */
    private $contentEn;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_title_EN", type="string", length=150, nullable=true)
     */
    private $seoTitleEn;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_title_RU", type="string", length=150, nullable=true)
     */
    private $seoTitleRu;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_description_RU", type="string", length=300, nullable=true)
     */
    private $seoDescriptionRu;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_description_EN", type="string", length=300, nullable=true)
     */
    private $seoDescriptionEn;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_keywords_RU", type="string", length=300, nullable=true)
     */
    private $seoKeywordsRu;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_keywords_EN", type="string", length=300, nullable=true)
     */
    private $seoKeywordsEn;

    /**
     * @var string
```

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

    /**
     * @var string
     *
     * @ORM\Column(name="status",
type="string", length=20,
nullable=false)
     */
    private $status;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="create_at",
type="datetime", nullable=false)
     */
    private $createAt;

    /**
     * @var integer
     *
     * @ORM\Column(name="author_id",
type="integer", nullable=false)
     */
    private $authorId;

    /**
     * @var string
     *
     * @ORM\Column(name="route",
type="string", length=255,
nullable=false)
     */
    private $route;



    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set titleRu
     *
     * @param string $titleRu
     *
     * @return Department
     */
    public function
setTitleRu($titleRu)
    {
        $this->titleRu = $titleRu;

        return $this;
    }

    /**
     * Get titleRu
     *
     * @return string
     */
    public function getTitleRu()
    {
        return $this->titleRu;
    }

    /**
     * Set titleEn
     *
     * @param string $titleEn
     *
     * @return Department
     */
    public function
setTitleEn($titleEn)
    {
        $this->titleEn = $titleEn;

        return $this;
    }

    /**
     * Get titleEn
     *
     * @return string
     */
    public function getTitleEn()
    {
        return $this->titleEn;
    }

    /**
     * Set contentRu
     *
     * @param string $contentRu
     *
     * @return Department
     */
    public function
setContentRu($contentRu)
    {
        $this->contentRu =
$contentRu;

        return $this;
    }

    /**
     * Get contentRu
     *
     * @return string
     */
    public function getContentRu()
    {
        return $this->contentRu;
    }

    /**
     * Set contentEn
     *
     * @param string $contentEn
     *
     * @return Department
     */
    public function
setContentEn($contentEn)
```

```
    {
        $this->contentEn =
$contentEn;

        return $this;
    }

    /**
     * Get contentEn
     *
     * @return string
     */
    public function getContentEn()
    {
        return $this->contentEn;
    }

    /**
     * Set seoTitleEn
     *
     * @param string $seoTitleEn
     *
     * @return Department
     */
    public function
setSeoTitleEn($seoTitleEn)
    {
        $this->seoTitleEn =
$seoTitleEn;

        return $this;
    }

    /**
     * Get seoTitleEn
     *
     * @return string
     */
    public function getSeoTitleEn()
    {
        return $this->seoTitleEn;
    }

    /**
     * Set seoTitleRu
     *
     * @param string $seoTitleRu
     *
     * @return Department
     */
    public function
setSeoTitleRu($seoTitleRu)
    {
        $this->seoTitleRu =
$seoTitleRu;

        return $this;
    }

    /**
     * Get seoTitleRu
     *
     * @return string
     */
    public function getSeoTitleRu()
    {
        return $this->seoTitleRu;
    }

    /**
     * Set seoDescriptionRu
     *
     * @param string
$seoDescriptionRu
     *
     * @return Department
     */
    public function
setSeoDescriptionRu($seoDescriptionR
u)
    {
        $this->seoDescriptionRu =
$seoDescriptionRu;

        return $this;
    }

    /**
     * Get seoDescriptionRu
     *
     * @return string
     */
    public function
getSeoDescriptionRu()
    {
        return $this-
>seoDescriptionRu;
    }

    /**
     * Set seoDescriptionEn
     *
     * @param string
$seoDescriptionEn
     *
     * @return Department
     */
    public function
setSeoDescriptionEn($seoDescriptionE
n)
    {
        $this->seoDescriptionEn =
$seoDescriptionEn;

        return $this;
    }

    /**
     * Get seoDescriptionEn
     *
     * @return string
     */
    public function
getSeoDescriptionEn()
    {
        return $this-
>seoDescriptionEn;
    }

    /**
     * Set seoKeywordsRu
     *
     * @param string $seoKeywordsRu
     *
     * @return Department
     */
    public function
setSeoKeywordsRu($seoKeywordsRu)
```

```
    {
        $this->seoKeywordsRu =
$seoKeywordsRu;

        return $this;
    }

    /**
     * Get seoKeywordsRu
     *
     * @return string
     */
    public function
getSeoKeywordsRu()
    {
        return $this->seoKeywordsRu;
    }

    /**
     * Set seoKeywordsEn
     *
     * @param string $seoKeywordsEn
     *
     * @return Department
     */
    public function
setSeoKeywordsEn($seoKeywordsEn)
    {
        $this->seoKeywordsEn =
$seoKeywordsEn;

        return $this;
    }

    /**
     * Get seoKeywordsEn
     *
     * @return string
     */
    public function
getSeoKeywordsEn()
    {
        return $this->seoKeywordsEn;
    }

    /**
     * Set thumbnail
     *
     * @param string $thumbnail
     *
     * @return Department
     */
    public function
setThumbnail($thumbnail)
    {
        $this->thumbnail =
$thumbnail;

        return $this;
    }

    /**
     * Get thumbnail
     *
     * @return string
     */
    public function getThumbnail()
    {
        return $this->thumbnail;
    }

    /**
     * Set status
     *
     * @param string $status
     *
     * @return Department
     */
    public function
setStatus($status)
    {
        $this->status = $status;

        return $this;
    }

    /**
     * Get status
     *
     * @return string
     */
    public function getStatus()
    {
        return $this->status;
    }

    /**
     * Set createAt
     *
     * @param \DateTime $createAt
     *
     * @return Department
     */
    public function
setCreateAt($createAt)
    {
        $this->createAt = $createAt;

        return $this;
    }

    /**
     * Get createAt
     *
     * @return \DateTime
     */
    public function getCreateAt()
    {
        return $this->createAt;
    }

    /**
     * Set authorId
     *
     * @param integer $authorId
     *
     * @return Department
     */
    public function
setAuthorId($authorId)
    {
        $this->authorId = $authorId;

        return $this;
    }

    /**
     * Get authorId
```

```
     *
     * @return integer
     */
    public function getAuthorId()
    {
        return $this->authorId;
    }

    /**
     * Set route
     *
     * @param string $route
     *
     * @return Pages
     */
    public function setRoute($route)
    {
        $this->route = $route;

        return $this;
    }

    /**
     * Get route
     *
     * @return string
     */
    public function getRoute()
    {
        return $this->route;
    }

}
<?php

namespace Model\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Faculties
 *
 * @ORM\Table(name="faculties")
 *
@ORM\Entity(repositoryClass="Model\R
epository\Faculties")
 */
class Faculties
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id",
type="integer", nullable=false)
     * @ORM\Id
     *
@ORM\GeneratedValue(strategy="IDENTI
TY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="title_RU",
type="string", length=150,
nullable=false)
     */
    private $titleRu;

    /**
     * @var string
     *
     * @ORM\Column(name="title_EN",
type="string", length=150,
nullable=false)
     */
    private $titleEn;

    /**
     * @var string
     *
     *
@ORM\Column(name="content_RU",
type="text", length=65535,
nullable=false)
     */
    private $contentRu;

    /**
     * @var string
     *
     *
@ORM\Column(name="content_EN",
type="text", length=65535,
nullable=false)
     */
    private $contentEn;

    /**
     * @var string
     *
     *
@ORM\Column(name="programme_EN",
type="text", length=65535,
nullable=false)
     */
    private $programmeEn;

    /**
     * @var string
     *
     *
@ORM\Column(name="programme_RU",
type="text", length=65535,
nullable=false)
     */
    private $programmeRu;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_title_RU",
type="string", length=150,
nullable=true)
     */
    private $seoTitleRu;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_title_EN",
type="string", length=150,
nullable=true)
     */
    private $seoTitleEn;
```

```
    /**
     * @var string
     *
     *
@ORM\Column(name="seo_description_EN
", type="string", length=300,
nullable=true)
     */
    private $seoDescriptionEn;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_description_RU
", type="string", length=300,
nullable=true)
     */
    private $seoDescriptionRu;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_keywords_RU",
type="string", length=300,
nullable=true)
     */
    private $seoKeywordsRu;

    /**
     * @var string
     *
     *
@ORM\Column(name="seo_keywords_EN",
type="string", length=300,
nullable=true)
     */
    private $seoKeywordsEn;

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

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

    /**
     * @var string
     *
     * @ORM\Column(name="status",
type="string", length=20,
nullable=false)
     */
    private $status;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="create_at",
type="datetime", nullable=false)
     */
    private $createAt;

    /**
     * @var integer
     *
     * @ORM\Column(name="author_id",
type="integer", nullable=false)
     */
    private $authorId;

    /**
     * @var string
     *
     * @ORM\Column(name="route",
type="string", length=255,
nullable=false)
     */
    private $route;

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


    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set titleRu
     *
     * @param string $titleRu
     *
     * @return Faculties
     */
    public function
setTitleRu($titleRu)
    {
        $this->titleRu = $titleRu;

        return $this;
    }

    /**
     * Get titleRu
     *
     * @return string
     */
    public function getTitleRu()
    {
        return $this->titleRu;
    }
```

```
    /**
     * Set titleEn
     *
     * @param string $titleEn
     *
     * @return Faculties
     */
    public function
setTitleEn($titleEn)
    {
        $this->titleEn = $titleEn;

        return $this;
    }

    /**
     * Get titleEn
     *
     * @return string
     */
    public function getTitleEn()
    {
        return $this->titleEn;
    }

    /**
     * Set contentRu
     *
     * @param string $contentRu
     *
     * @return Faculties
     */
    public function
setContentRu($contentRu)
    {
        $this->contentRu =
$contentRu;

        return $this;
    }

    /**
     * Get contentRu
     *
     * @return string
     */
    public function getContentRu()
    {
        return $this->contentRu;
    }

    /**
     * Set contentEn
     *
     * @param string $contentEn
     *
     * @return Faculties
     */
    public function
setContentEn($contentEn)
    {
        $this->contentEn =
$contentEn;

        return $this;
    }

    /**
     * Get contentEn
     *
     * @return string
     */
    public function getContentEn()
    {
        return $this->contentEn;
    }

    /**
     * Set programmeEn
     *
     * @param string $programmeEn
     *
     * @return Faculties
     */
    public function
setProgrammeEn($programmeEn)
    {
        $this->programmeEn =
$programmeEn;

        return $this;
    }

    /**
     * Get programmeEn
     *
     * @return string
     */
    public function getProgrammeEn()
    {
        return $this->programmeEn;
    }

    /**
     * Set programmeRu
     *
     * @param string $programmeRu
     *
     * @return Faculties
     */
    public function
setProgrammeRu($programmeRu)
    {
        $this->programmeRu =
$programmeRu;

        return $this;
    }

    /**
     * Get programmeRu
     *
     * @return string
     */
    public function getProgrammeRu()
    {
        return $this->programmeRu;
    }

    /**
     * Set seoTitleRu
     *
     * @param string $seoTitleRu
     *
     * @return Faculties
     */
    public function
setSeoTitleRu($seoTitleRu)
```

```
    {
        $this->seoTitleRu =
$seoTitleRu;

        return $this;
    }

    /**
     * Get seoTitleRu
     *
     * @return string
     */
    public function getSeoTitleRu()
    {
        return $this->seoTitleRu;
    }

    /**
     * Set seoTitleEn
     *
     * @param string $seoTitleEn
     *
     * @return Faculties
     */
    public function
setSeoTitleEn($seoTitleEn)
    {
        $this->seoTitleEn =
$seoTitleEn;

        return $this;
    }

    /**
     * Get seoTitleEn
     *
     * @return string
     */
    public function getSeoTitleEn()
    {
        return $this->seoTitleEn;
    }

    /**
     * Set seoDescriptionEn
     *
     * @param string
$seoDescriptionEn
     *
     * @return Faculties
     */
    public function
setSeoDescriptionEn($seoDescriptionE
n)
    {
        $this->seoDescriptionEn =
$seoDescriptionEn;

        return $this;
    }

    /**
     * Get seoDescriptionEn
     *
     * @return string
     */
    public function
getSeoDescriptionEn()
    {
        return $this-
>seoDescriptionEn;
    }

    /**
     * Set seoDescriptionRu
     *
     * @param string
$seoDescriptionRu
     *
     * @return Faculties
     */
    public function
setSeoDescriptionRu($seoDescriptionR
u)
    {
        $this->seoDescriptionRu =
$seoDescriptionRu;

        return $this;
    }

    /**
     * Get seoDescriptionRu
     *
     * @return string
     */
    public function
getSeoDescriptionRu()
    {
        return $this-
>seoDescriptionRu;
    }

    /**
     * Set seoKeywordsRu
     *
     * @param string $seoKeywordsRu
     *
     * @return Faculties
     */
    public function
setSeoKeywordsRu($seoKeywordsRu)
    {
        $this->seoKeywordsRu =
$seoKeywordsRu;

        return $this;
    }

    /**
     * Get seoKeywordsRu
     *
     * @return string
     */
    public function
getSeoKeywordsRu()
    {
        return $this->seoKeywordsRu;
    }

    /**
     * Set seoKeywordsEn
     *
     * @param string $seoKeywordsEn
     *
     * @return Faculties
     */
    public function
```

```
setSeoKeywordsEn($seoKeywordsEn)
    {
        $this->seoKeywordsEn =
$seoKeywordsEn;

        return $this;
    }

    /**
     * Get seoKeywordsEn
     *
     * @return string
     */
    public function
getSeoKeywordsEn()
    {
        return $this->seoKeywordsEn;
    }

    /**
     * Set pdfRu
     *
     * @param string $pdfRu
     *
     * @return Faculties
     */
    public function setPdfRu($pdfRu)
    {
        $this->pdfRu = $pdfRu;

        return $this;
    }

    /**
     * Get pdfRu
     *
     * @return string
     */
    public function getPdfRu()
    {
        return $this->pdfRu;
    }

    /**
     * Set pdfEn
     *
     * @param string $pdfEn
     *
     * @return Faculties
     */
    public function setPdfEn($pdfEn)
    {
        $this->pdfEn = $pdfEn;

        return $this;
    }

    /**
     * Get pdfEn
     *
     * @return string
     */
    public function getPdfEn()
    {
        return $this->pdfEn;
    }

    /**
     * Set status
     *
     * @param string $status
     *
     * @return Faculties
     */
    public function
setStatus($status)
    {
        $this->status = $status;

        return $this;
    }

    /**
     * Get status
     *
     * @return string
     */
    public function getStatus()
    {
        return $this->status;
    }

    /**
     * Set createAt
     *
     * @param \DateTime $createAt
     *
     * @return Faculties
     */
    public function
setCreateAt($createAt)
    {
        $this->createAt = $createAt;

        return $this;
    }

    /**
     * Get createAt
     *
     * @return \DateTime
     */
    public function getCreateAt()
    {
        return $this->createAt;
    }

    /**
     * Set authorId
     *
     * @param integer $authorId
     *
     * @return Faculties
     */
    public function
setAuthorId($authorId)
    {
        $this->authorId = $authorId;

        return $this;
    }

    /**
     * Get authorId
     *
     * @return integer
     */
    public function getAuthorId()
```

```
    {
        return $this->authorId;
    }

    /**
     * Set route
     *
     * @param string $route
     *
     * @return Faculties
     */
    public function setRoute($route)
    {
        $this->route = $route;

        return $this;
    }

    /**
     * Get route
     *
     * @return string
     */
    public function getRoute()
    {
        return $this->route;
    }


    /**
     * Set thumbnail
     *
     * @param string $thumbnail
     *
     * @return Pages
     */
    public function
setThumbnail($thumbnail)
    {
        $this->thumbnail =
$thumbnail;

        return $this;
    }

    /**
     * Get thumbnail
     *
     * @return string
     */
    public function getThumbnail()
    {
        return $this->thumbnail;
    }
}
<?php

namespace Model\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Media
 *
 * @ORM\Table(name="media",
indexes={@ORM\Index(name="media_empl
oyee_id", columns={"employee_id"}),
@ORM\Index(name="media_title",
columns={"title"})})
 *
@ORM\Entity(repositoryClass="Model\R
epository\Media")
 */
class Media
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id",
type="integer", nullable=false)
     * @ORM\Id
     *
@ORM\GeneratedValue(strategy="IDENTI
TY")
     */
    private $id;

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

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

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

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="create_at",
type="datetime", nullable=true)
     */
    private $createAt;

    /**
     * @var integer
     *
     *
@ORM\Column(name="employee_id",
type="integer", nullable=false)
     */
    private $employeeId;


    /**
     * Get id
     *
     * @return integer
     */
```

```
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set title
     *
     * @param string $title
     *
     * @return Media
     */
    public function setTitle($title)
    {
        $this->title = $title;

        return $this;
    }

    /**
     * Get title
     *
     * @return string
     */
    public function getTitle()
    {
        return $this->title;
    }

    /**
     * Set url
     *
     * @param string $url
     *
     * @return Media
     */
    public function setUrl($url)
    {
        $this->url = $url;

        return $this;
    }

    /**
     * Get url
     *
     * @return string
     */
    public function getUrl()
    {
        return $this->url;
    }

    /**
     * Set mime
     *
     * @param string $mime
     *
     * @return Media
     */
    public function setMime($mime)
    {
        $this->mime = $mime;

        return $this;
    }

    /**
     * Get mime
     *
     * @return string
     */
    public function getMime()
    {
        return $this->mime;
    }

    /**
     * Set createAt
     *
     * @param \DateTime $createAt
     *
     * @return Media
     */
    public function setCreateAt($createAt)
    {
        $this->createAt = $createAt;

        return $this;
    }

    /**
     * Get createAt
     *
     * @return \DateTime
     */
    public function getCreateAt()
    {
        return $this->createAt;
    }

    /**
     * Set employeeId
     *
     * @param integer $employeeId
     *
     * @return Media
     */
    public function setEmployeeId($employeeId)
    {
        $this->employeeId = $employeeId;

        return $this;
    }

    /**
     * Get employeeId
     *
     * @return integer
     */
    public function getEmployeeId()
    {
        return $this->employeeId;
    }
}
<?php

namespace Model\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Menu
 *
 * @ORM\Table(name="menu")
```

```
 *
@ORM\Entity(repositoryClass="Model\R
epository\Menu")
 */
class Menu
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id",
type="integer", nullable=false)
     * @ORM\Id
     *
@ORM\GeneratedValue(strategy="IDENTI
TY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name_RU",
type="string", length=255,
nullable=false)
     */
    private $nameRu;

    /**
     * @var string
     *
     * @ORM\Column(name="name_EN",
type="string", length=255,
nullable=false)
     */
    private $nameEn;

    /**
     * @var string
     *
     * @ORM\Column(name="code",
type="text", nullable=true)
     */
    private $code;

    /**
     * @var string
     *
     *
@ORM\Column(name="system_name",
type="string",length=150,
nullable=false)
     */
    private $systemName;



    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set nameRu
     *
     * @param string $nameRu
     *
     * @return Menu
     */
    public function
setNameRu($nameRu)
    {
        $this->nameRu = $nameRu;

        return $this;
    }

    /**
     * Get nameRu
     *
     * @return string
     */
    public function getNameRu()
    {
        return $this->nameRu;
    }

    /**
     * Set nameEn
     *
     * @param string $nameEn
     *
     * @return Menu
     */
    public function
setNameEn($nameEn)
    {
        $this->nameEn = $nameEn;

        return $this;
    }

    /**
     * Get nameEn
     *
     * @return string
     */
    public function getNameEn()
    {
        return $this->nameEn;
    }

    /**
     * Set code
     *
     * @param string $code
     *
     * @return Menu
     */
    public function setCode($code)
    {
        $this->code = $code;

        return $this;
    }

    /**
     * Get code
     *
     * @return string
     */
    public function getCode()
    {
        return $this->code;
```

```
    }

    /**
     * Set systemName
     *
     * @param string $systemName
     *
     * @return Menu
     */
    public function
setSystemName($systemName)
    {
        $this->systemName =
$systemName;

        return $this;
    }

    /**
     * Get systemName
     *
     * @return string
     */
    public function getSystemName()
    {
        return $this->systemName;
    }
}
<?php

namespace Model\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Users
 *
 * @ORM\Table(name="users",
uniqueConstraints={@ORM\UniqueConstr
aint(name="dont_care",
columns={"email"})})
 *
@ORM\Entity(repositoryClass="Model\R
epository\Users")
 */
class Users
{

    const STATUS_ACTIVE       = 1;
// Active user.
    const STATUS_RETIRED      = 2;
// Inactive user.

    /**
     * @var integer
     *
     * @ORM\Column(name="id",
type="integer", nullable=false)
     * @ORM\Id
     *
@ORM\GeneratedValue(strategy="IDENTI
TY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="email",
type="string", length=128,
nullable=false)
     */
    private $email;

    /**
     * @var string
     *
     * @ORM\Column(name="firstname",
type="string", length=300,
nullable=false)
     */
    private $firstname;

    /**
     * @var string
     *
     * @ORM\Column(name="lastname",
type="string", length=300,
nullable=false)
     */
    private $lastname;

    /**
     * @var string
     *
     *
@ORM\Column(name="firstname_RU",
type="string", length=300,
nullable=false)
     */
    private $firstnameRu;

    /**
     * @var string
     *
     *
@ORM\Column(name="lastname_RU",
type="string", length=300,
nullable=false)
     */
    private $lastnameRu;

    /**
     * @var integer
     *
     * @ORM\Column(name="status",
type="integer", nullable=false)
     */
    private $status;

    /**
     * @var string
     *
     *
@ORM\Column(name="pwd_reset_token",
type="string", length=32,
nullable=true)
     */
    private $passwordResetToken;

    /**
     * @var \DateTime
     *
     *
@ORM\Column(name="pwd_reset_token_cr
eation_date", type="date",
nullable=true)
     */
```

```
    private $passwordResetTokenCreationDate;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="date_created", type="date", nullable=false)
     */
    private $dateCreated;

    /**
     * @var integer
     *
     * @ORM\Column(name="phone", type="string", nullable=true)
     */
    private $phone;

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

    /**
     * @var string
     *
     * @ORM\Column(name="password", type="string", length=255, nullable=false)
     */
    private $password;



    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set email
     *
     * @param string $email
     *
     * @return Users
     */
    public function setEmail($email)
    {
        $this->email = $email;

        return $this;
    }

    /**
     * Get email
     *
     * @return string
     */
    public function getEmail()
    {
        return $this->email;
    }

    /**
     * Set firstname
     *
     * @param string $firstname
     *
     * @return Users
     */
    public function setFirstname($firstname)
    {
        $this->firstname = $firstname;

        return $this;
    }

    /**
     * Get firstname
     *
     * @return string
     */
    public function getFirstname()
    {
        return $this->firstname;
    }

    /**
     * Set lastname
     *
     * @param string $lastname
     *
     * @return Users
     */
    public function setLastname($lastname)
    {
        $this->lastname = $lastname;

        return $this;
    }

    /**
     * Get lastname
     *
     * @return string
     */
    public function getLastname()
    {
        return $this->lastname;
    }

    /**
     * Set firstnameRu
     *
     * @param string $firstnameRu
     *
     * @return Users
     */
    public function setFirstnameRu($firstnameRu)
    {
        $this->firstnameRu = $firstnameRu;
```

```
        return $this;
    }

    /**
     * Get firstnameRu
     *
     * @return string
     */
    public function getFirstnameRu()
    {
        return $this->firstnameRu;
    }

    /**
     * Set lastnameRu
     *
     * @param string $lastnameRu
     *
     * @return Users
     */
    public function
setLastnameRu($lastnameRu)
    {
        $this->lastnameRu =
$lastnameRu;

        return $this;
    }

    /**
     * Get lastnameRu
     *
     * @return string
     */
    public function getLastnameRu()
    {
        return $this->lastnameRu;
    }

    /**
     * Set status
     *
     * @param boolean $status
     *
     * @return Users
     */
    public function
setStatus($status)
    {
        $this->status = $status;

        return $this;
    }

    /**
     * Get status
     *
     * @return boolean
     */
    public function getStatus()
    {
        return $this->status;
    }

    /**
     * Set pwdResetToken
     *
     * @param string $pwdResetToken
     *
     * @return Users
     */
    public function
setPasswordResetToken($passwordReset
Token)
    {
        $this->passwordResetToken =
$passwordResetToken;

        return $this;
    }

    /**
     * Get pwdResetToken
     *
     * @return string
     */
    public function
getPasswordResetToken()
    {
        return $this-
>passwordResetToken;
    }

    /**
     * Set pwdResetTokenCreationDate
     *
     * @param \DateTime
$pwdResetTokenCreationDate
     *
     * @return Users
     */
    public function
setPasswordResetTokenCreationDate($p
asswordResetTokenCreationDate)
    {
        $this-
>passwordResetTokenCreationDate =
$passwordResetTokenCreationDate;

        return $this;
    }

    /**
     * Get pwdResetTokenCreationDate
     *
     * @return \DateTime
     */
    public function
getPasswordResetTokenCreationDate()
    {
        return $this-
>passwordResetTokenCreationDate;
    }

    /**
     * Set dateCreated
     *
     * @param \DateTime $dateCreated
     *
     * @return Users
     */
    public function
setDateCreated($dateCreated)
    {
        $this->dateCreated =
$dateCreated;

        return $this;
```

```
    }

    /**
     * Get dateCreated
     *
     * @return \DateTime
     */
    public function getDateCreated()
    {
        return $this->dateCreated;
    }

    /**
     * Set phone
     *
     * @param integer $phone
     *
     * @return Users
     */
    public function setPhone($phone)
    {
        $this->phone = $phone;

        return $this;
    }

    /**
     * Get phone
     *
     * @return integer
     */
    public function getPhone()
    {
        return $this->phone;
    }

    /**
     * Set photo
     *
     * @param string $photo
     *
     * @return Users
     */
    public function setPhoto($photo)
    {
        $this->photo = $photo;

        return $this;
    }

    /**
     * Get photo
     *
     * @return string
     */
    public function getPhoto()
    {
        return $this->photo;
    }

    /**
     * Set password
     *
     * @param string $password
     *
     * @return Users
     */
    public function setPassword($password)
    {
        $this->password = $password;

        return $this;
    }

    /**
     * Get password
     *
     * @return string
     */
    public function getPassword()
    {
        return $this->password;
    }
}
<?php

namespace Model\Repository;

use Doctrine\ORM\EntityRepository;

class Department extends
EntityRepository
{

    const STATUS_PUBLISH =
'publish';

    public function
save($postParams)
    {
        $entity = null;

        if(empty($postParams["id"]))
{
            $entity = new
\Model\Entity\Department;

            $entity->setCreateAt(new
\DateTime());
            $entity-
>setAuthorId($postParams['author_id'
]);

        } else {
            $entity = $this->getEM()
                -
>find('\Model\Entity\Department',
$postParams["id"]);
        }

        $entity-
>setAuthorId($postParams['author_id'
]);

        $entity-
>setSeoKeywordsRu($postParams['seo_k
eywords_ru']);
        $entity-
>setSeoKeywordsEn($postParams['seo_k
eywords_en']);

        $entity-
>setSeoDescriptionRu($postParams['se
o_description_ru']);
        $entity-
>setSeoDescriptionEn($postParams['se
o_description_en']);
```

```
        $entity->setSeoTitleRu($postParams['seo_title_ru']);
        $entity->setSeoTitleEn($postParams['seo_title_en']);

        $entity->setTitleRu($postParams["title_ru"]);
        $entity->setTitleEn($postParams["title_en"]);

        $entity->setThumbnail($postParams["thumbnail"]);
        $entity->setRoute($postParams["route"]);

        $entity->setContentEn($postParams["content_en"]);
        $entity->setContentRu($postParams["content_ru"]);

        $entity->setStatus($postParams["status"]);

        $this->getEM()->persist($entity);
        $this->getEM()->flush();

        return $entity;
    }

    public function trash($postParams)
    {
        $entity = $this->getEM()
            ->find('Model\Entity\Department', $postParams['id']);

        if(!empty($entity)) {
            $entity->setStatus('trash');
            $this->getEM()->flush($entity);
            $this->getEM()->flush();

            return true;
        }

        return false;
    }


    public function remove($postParams)
    {
        $entity = $this->getEM()
            ->find('Model\Entity\Department', $postParams['id']);

        if(!empty($entity)) {
            $this->getEM()->remove($entity);
            $this->getEM()->flush();

            return true;
        }

        return false;
    }

    public function search($string, $lang)
    {
        $results = [];
        $fields = [
            'ru_RU'=>
                [
                    'title'=>'titleRu',
                    'content'=>'contentRu',
                ],
            "en_GB"=>
                [
                    'title'=>'titleEn',
                    'content'=>'contentEn',
                ]
        ];
        $field = $fields[$lang];
        $class = (new \ReflectionClass($this))->getShortName();

        $results = $this->getEM()->createQueryBuilder()
            ->select("p.".$field['title'].' as title',"p.".$field['content'].' as content','p.route')
            ->from("Model\Entity\\{$class}","p")
            ->where("p.".$field['title']." LIKE :str")
            ->orWhere("p.".$field['content']." LIKE :str")
            ->andWhere("p.status = :status")
            ->setParameter("status", "publish")
            ->setParameter("str", "%".$string."%")
            ->getQuery()
            ->getArrayResult();
        return $results;
    }

    public function getEM()
    {
        return $this->_em;
    }
}
<?php
```

```php
namespace Model\Repository;

use Doctrine\ORM\EntityRepository;

class Faculties extends
EntityRepository
{

    const STATUS_PUBLISH =
'publish';

    public function
save($postParams)
    {
        $entity = null;

        if(empty($postParams["id"]))
{
            $entity = new
\Model\Entity\Faculties;

            $entity->setCreateAt(new
\DateTime());
            $entity-
>setAuthorId($postParams['author_id'
]);

        } else {
            $entity = $this->getEM()
                -
>find('\Model\Entity\Faculties',
$postParams["id"]);
        }

        $entity-
>setAuthorId($postParams['author_id'
]);

        $entity-
>setSeoKeywordsRu($postParams['seo_k
eywords_ru']);
        $entity-
>setSeoKeywordsEn($postParams['seo_k
eywords_en']);

        $entity-
>setSeoDescriptionRu($postParams['se
o_description_ru']);
        $entity-
>setSeoDescriptionEn($postParams['se
o_description_en']);

        $entity-
>setSeoTitleRu($postParams['seo_titl
e_ru']);
        $entity-
>setSeoTitleEn($postParams['seo_titl
e_en']);

        $entity-
>setTitleRu($postParams["title_ru"])
;
        $entity-
>setTitleEn($postParams["title_en"])
;

        $entity-
>setThumbnail($postParams["thumbnail
"]);

        $entity-
>setRoute($postParams["route"]);

        $entity-
>setContentEn($postParams["content_e
n"]);
        $entity-
>setContentRu($postParams["content_r
u"]);

        $entity-
>setProgrammeEn($postParams["program
me_en"]);
        $entity-
>setProgrammeRu($postParams["program
me_ru"]);

        $entity-
>setStatus($postParams["status"]);

        if ($postParams['pdf_ru']) {
            $entity-
>setPdfRu($postParams['pdf_ru']);
        }

        if ($postParams['pdf_en']) {
            $entity-
>setPdfEn($postParams['pdf_en']);
        }

        try {
            $this->getEM()-
>persist($entity);
            $this->getEM()->flush();
        } catch (\Exception $e) {
            throw new
\Exception("Something went wrong");
        }
        // save relationships
        $this->getEM()
                -
>getRepository("Model\Entity\Faculti
esRelDepartment")
                -
>update($postParams, $entity-
>getId());
        $this->getEM()
                -
>getRepository("Model\Entity\Faculti
esRelTeachers")
                -
>update($postParams, $entity-
>getId());

        return $entity;
    }

    public function
findPublishedPosts()
    {
        $entityManager = $this-
>getEM();

        $queryBuilder =
$entityManager-
>createQueryBuilder();

        $queryBuilder->select('n')
            -
```

```
>from("Model\Entity\Faculties", 'n')
            ->where('n.status =
:status')
            ->orderBy('n.createAt',
'DESC')
            ->setParameter("status",
"publish");

        return $queryBuilder-
>getQuery();
    }

    public function
trash($postParams)
    {
        $entity = $this->getEM()
            -
>find('Model\Entity\Faculties',
$postParams['id']);

        if(!empty($entity)) {
            $entity-
>setStatus('trash');
            $this->getEM()-
>flush($entity);
            $this->getEM()->flush();

            return true;
        }

        return false;
    }


     public function
removeFile($postParams)
    {
        $entity = $this->getEM()
            -
>find('Model\Entity\Faculties',
$postParams['id']);

        if (!empty($entity)) {

            if
($postParams['local']=="pdf_ru") {
                $entity-
>setPdfRu("");
            }
            if
($postParams['local']=="pdf_en") {
                $entity-
>setPdfEn("");
            }

            $this->getEM()->flush();

            return true;
        }

        return false;
    }


    public function
remove($postParams)
    {
        $entity = $this->getEM()
            -
>find('Model\Entity\Faculties',
$postParams['id']);

        if(!empty($entity)) {

            $this->getEM()-
>remove($entity);
            $this->getEM()->flush();

            return true;
        }

        return false;
    }

    public function
getDepartments($id)
    {
        $result = [];
        $models = $this->getEM()
            -
>getRepository("Model\Entity\Faculti
esRelDepartment")
            ->findByFacultyId($id);
        foreach ($models as $item):
            $result[] = $item-
>getDepartmentId();
        endforeach;

        return $result;
    }

    public function getTeachers($id)
    {
        $result = [];
        $models = $this->getEM()
            -
>getRepository("Model\Entity\Faculti
esRelTeachers")
            ->findByFacultyId($id);
        foreach ($models as $item):
            $result[] = $item-
>getTeacherId();
        endforeach;

        return $result;
    }

    public function
getTeachersData($id, $lang)
    {
        $result = [];
        $teachers = $this->getEM()-
>createQueryBuilder()
                ->select("f.nameRu",
"f.nameEn","f.descriptionRu","f.desc
riptionEn","f.positionRu","f.positio
nEn","f.photo")
                -
>from("Model\Entity\Teachers", "f")
                -
>join("Model\Entity\FacultiesRelTeac
hers", "fcr", "WITH", "f.id =
fcr.teacherId")
                ->where("f.status =
:status")
                -
>andWhere("fcr.facultyId = :id")
                -
```

```
>setParameter("status", "publish")
                ->setParameter("id",
$id)
                ->getQuery()
                ->getArrayResult();

        foreach ($teachers as $key
=> $item):
            $result[$key]["name"] =
$lang == "ru" ? $item['nameRu'] :
$item['nameEn'];
            $result[$key]['photo'] =
$item['photo'];

$result[$key]['position'] = $lang ==
"ru" ? $item['positionRu'] :
$item['positionEn'];

$result[$key]['description'] = $lang
== "ru" ? $item['descriptionRu'] :
$item['descriptionEn'];
        endforeach;

        return $result;
    }

    public function
getDepartmentsData($id, $lang)
    {
        $result = [];
        $teachers = $this->getEM()-
>createQueryBuilder()
                -
>select("f.titleRu",
"f.titleEn","f.route")
                -
>from("Model\Entity\Department",
"f")
                -
>join("Model\Entity\FacultiesRelDepa
rtment", "fcr", "WITH", "f.id =
fcr.departmentId")
                ->where("f.status =
:status")
                -
>andWhere("fcr.facultyId = :id")
                -
>setParameter("status", "publish")
                ->setParameter("id",
$id)
                ->getQuery()
                ->getArrayResult();

        foreach ($teachers as $key
=> $item):
            $result[$key]["title"] =
$lang == "ru" ? $item['titleRu'] :
$item['titleEn'];
            $result[$key]["route"] =
$item['route'];
        endforeach;

        return $result;
    }

    public function
getTitleById($id, $local)
    {
        $result = "";
        $model = $this->getEM()-
>getRepository("Model\Entity\Faculti
es")->find($id);

        if ($model) {
            $result = $local=="ru" ?
$model->getTitleRu() : $model-
>getTitleEn();
        }
        return $result;
    }


    public function getList($local)
    {
        $result = [];
        $models = $this->getEM()
                -
>getRepository("Model\Entity\Faculti
es")
                -
>findByStatus("publish");

        if ($models) {

            foreach ($models as
$item) {
                $result[$item-
>getId()] = $local == "ru_RU" ?
                    $item-
>getTitleRu() :
                    $item-
>getTitleEn();
            }

        }
        return $result;
    }

    public function search($string,
$lang)
    {
        $results = [];
        $fields = [
            'ru_RU'=>
                [

'title'=>'titleRu',

'content'=>'contentRu',
                ],
            "en_GB"=>
                [

'title'=>'titleEn',

'content'=>'contentEn',
                ]
        ];
        $field = $fields[$lang];
        $class = (new
\ReflectionClass($this))-
>getShortName();

        $results = $this->getEM()-
>createQueryBuilder()
                -
>select("p.".$field['title'].' as
title',"p.".$field['content'].' as
```

```
content','p.route')
                -
>from("Model\Entity\\{$class}","p")
                -
>where("p.".$field['title']." LIKE
:str")
                -
>orWhere("p.".$field['content']."
LIKE :str")
                ->andWhere("p.status
= :status")
                -
>setParameter("status", "publish")
                -
>setParameter("str",
"%".$string."%")
                ->getQuery()
                ->getArrayResult();
        return $results;
    }

    public function getEM()
    {
        return $this->_em;
    }
}
<?php

namespace Model\Repository;

use Doctrine\ORM\EntityRepository;

class Menu extends EntityRepository
{

    public function
save($postParams)
    {
        $model = $this->getEM()
                -
>getRepository("Model\Entity\Menu")
                -
>find($postParams['id']);

        $model-
>setCode($postParams['code']);

        $this->getEM()->flush();
    }

    public function getModules()
    {
        $result = [
            'faculties',
            'news',
            'department',
            'pages'
        ];

        $faculties = $this->getEM()
                -
>getRepository("Model\Entity\Faculti
es")
                -
>findBy(['status'=>"publish"]);

        $news = $this->getEM()
                -
>getRepository("Model\Entity\News")
                -
>findBy(['status'=>"publish"]);

        $department = $this->getEM()
                -
>getRepository("Model\Entity\Departm
ent")
                -
>findBy(['status'=>"publish"]);

        $pages = $this->getEM()
                -
>getRepository("Model\Entity\Pages")
                -
>findBy(['status'=>"publish"]);

        if ($faculties) {
            $result['faculties'] =
$this-
>generateArrForMenu($faculties,
"/faculties/");
        }

        if ($news) {
            $result['news'] = $this-
>generateArrForMenu($news,
"/news/");
        }

        if ($department) {
            $result['department'] =
$this-
>generateArrForMenu($department,
"/departments/");
        }

        if ($pages) {
            $result['pages'] =
$this->generateArrForMenu($pages,
"/");
        }
        /*$result['faculties'] = [
            'route'=>"/faculties",
            "title_en"=>"Faculties",
            "title_ru"=>"Факультеты"
        ];

        $result['news'] = [
            'route'=>"/news",
            "title_en"=>"News",
            "title_ru"=>"Новости"
        ];
        */

        return $result;


    }

    private function
generateArrForMenu($obj, $prefix)
    {
        $result = [];
        foreach ($obj as $item) {
            $result[] = [

'route'=>$prefix.$item->getRoute(),

'title_ru'=>$item->getTitleRu(),
```

```
'title_en'=>$item->getTitleEn(),
            ];
        }

        return $result;
    }

    public function getEM()
    {
        return $this->_em;
    }
}
<?php

namespace Model\Repository;

use Doctrine\ORM\EntityRepository;
use Zend\Crypt\Password\Bcrypt;
/**
 * Users
 *
 * This class was generated by the
Doctrine ORM. Add your own custom
 * repository methods below.
 */
class Users extends EntityRepository
{
    public function save($data,
$files)
    {
        $model = $this-
>find($data['id']);

        if (!$model) {
            $model = new
\Model\Entity\Users();
            $this-
>isEmailExists($data['email']);
            if
(empty($data['password'])) throw new
\Exception("You missed password");
        }

        if (empty($data['email']))
throw new \Exception("Email can not
be empty");
        if (empty($data['status']))
throw new \Exception("Status can not
be empty");
        if
(empty($data['firstname'])) throw
new \Exception("Firstname can not be
empty");
        if
(empty($data['lastname'])) throw new
\Exception("Lastname can not be
empty");

        $model-
>setEmail($data['email']);
        $model-
>setFirstname($data['firstname']);
        $model-
>setLastname($data['lastname']);
        $model-
>setFirstnameRu($data['firstnameRu']
);
        $model-
>setLastnameRu($data['lastnameRu']);

        if (!empty($data['photo']))
$model->setPhoto($data['photo']);



        if
(!empty($data['password']) &&
$data['password'] ==
$data['password_verify'])
        {
            $bcrypt = new Bcrypt();
            $passwordHash = $bcrypt-
>create($data['password']);
            $model-
>setPassword($passwordHash);
        }
        $model-
>setStatus($data['status']);
        $model-
>setPhone($data['phone']);
        $model->setDateCreated(new
\DateTime());

        $this->getEM()-
>persist($model);

        $this->getEM()->flush();

    }

    public function
isEmailExists($email)
    {
        $model = $this-
>findBy(['email'=>$email]);
        if (!empty($model)) throw
new \Exception("This email has
already occupied");
    }


    public function
getGridList($conditionals)
    {
        $where = $this-
>_getFilters($conditionals);

        $itemsCnt = $this->getEM()-
>createQueryBuilder()
            ->select("count(u.id)")
            -
>from("Model\Entity\Users", "u");
        if (!empty($where))
$itemsCnt = $itemsCnt-
>where($where);
        $itemsCnt = $itemsCnt-
>getQuery()
            -
>getSingleScalarResult();

        $items = $this->getEM()-
>createQueryBuilder()
            -
>select("u.id","u.lastname",
"u.lastnameRu", 'u.firstname',
'u.firstnameRu', 'u.email',
'u.status')
```

```
            -
>from("Model\Entity\Users", "u");
        if (!empty($where)) $items =
$items->where($where);
        $items = $items-
>orderBy("u.id", "ASC")
            ->getQuery()
            ->getArrayResult();

        return ['count'=>$itemsCnt,
'content'=>$items];
    }

    private function
_getFilters($data)
    {
        $where = [];
        if (!empty($data['status']))
{
            $where[] = "u.status =
'".$data['status']."'";
        }
        if (!empty($data['search']))
{
            $where[] = "(u.firstname
LIKE '%".$data['search']."%' OR ".
                "u.lastname LIKE
'%".$data['search']."%' OR ".
                "u.firstnameRu LIKE
'%".$data['search']."%' OR ".
                "u.lastnameRu LIKE
'%".$data['search']."%' )";
        }

        return implode(" AND ",
$where);
    }



    public function getEM()
    {
        return $this->_em;
    }
}
<?php
/**
 * @link
http://github.com/zendframework/Zend
SkeletonIndex for the canonical
source repository
 * @copyright Copyright (c) 2005-
2016 Zend Technologies USA Inc.
(http://www.zend.com)
 * @license
http://framework.zend.com/license/ne
w-bsd New BSD License
 */

namespace Cron;

return [
    'console' => [
        'router' => [
            'routes' => [
                'main-news' => [
                    'options' => [
         'route'    => 'send main
news',
          'defaults' => [

'controller' =>
Controller\CronController::class,
                                'action'
=> 'news'
                            ],
                        ],
      ],
        ],
    ],
        ],
    'controllers' => [
        'factories' => [

Controller\CronController::class =>
Controller\Factory\CronControllerFac
tory::class,

        ],
    ],
    'access_filter' => [
        'options' => [
            // Фильтр доступа может
работать в 'ограничительном'
(рекомендуется) или 'разрешающем'
            // режиме. В
ограничительном режиме все действия
контроллера должны быть явно
перечислены
            // под ключом
конфигурации 'access_filter', а
доступ к любому не перечисленному
действию
            // для неавторизованных
пользователей запрещен. В
разрешающем режиме, даже если
действие не
            // указано под ключом
'access_filter', доступ к нему
разрешен для всех (даже для
            // неавторизованных
пользователей. Рекомендуется
использовать более безопасный
ограничительный режим.
            'mode' => 'restrictive'
        ],
        'controllers' => [

Controller\IndexController::class =>
[
                ['actions' =>
['news'], 'allow' => '*'],
                // Позволяем
авторизованным пользователям
обращаться к действию "settings".
                ['actions' =>
['index'], 'allow' => '@']
            ],
        ]
    ],
    'translator' => [
   'translation_file_patterns' => [
            [
      'type'     => 'gettext',
      'base_dir' => __DIR__ .
'/../language',
      'pattern'  => '%s.mo',
            ],
```

```
            ],
    ],
     'view_manager' => [
         'template_path_stack' => [
             __DIR__ . '/../view',
         ],
     ],
];
<?php

namespace App\Department\Controller;

use
Zend\Mvc\Controller\AbstractActionCo
ntroller;
use Zend\View\Model\ViewModel;
use App\Index\Form\Subscribe as
Form;


class DepartmentController extends
AbstractActionController
{

    private $em;
    private $config;
    private $translator;

    public function
__construct($entityManager, $config,
$translator)
    {
        $this->em = $entityManager;
        $this->config = $config;
        $this->translator =
$translator;
    }
    public function indexAction()
    {
        $page = $this->getEvent()
            ->getRouteMatch()
            ->getParam('page');
        $lang = $this->extra()-
>getLang();

        $em = $this->em;
        $result = $em-
>getRepository("Model\Entity\Departm
ent")->findOneByRoute($page);

        if ($result && $result-
>getStatus() == 'publish') {
            $titleSeo =
($lang=="ru_RU") ? $result-
>getSeoTitleRu() : $result-
>getSeoTitleEn();
            $descriptionSeo =
($lang=="ru_RU") ? $result-
>getSeoDescriptionRu() : $result-
>getSeoDescriptionEn();
            $keywordsSeo =
($lang=="ru_RU") ? $result-
>getSeoKeywordsRu() : $result-
>getSeoKeywordsEn();
            $title =
($lang=="ru_RU") ? $result-
>getTitleRu() : $result-
>getTitleEn();
            $content =
($lang=="ru_RU") ? $result-
>getContentRu() : $result-
>getContentEn();

            return (new
ViewModel())->setVariables([
                'titleSeo'=>
$titleSeo,
                'descriptionSeo' =>
$descriptionSeo,

'keywordsSeo'=>$keywordsSeo,
                'title'=> $title,
                'form' => new
Form(),
                'content'=>$content,
                'lastNews'=>$this-
>extra()->getLastNews(),
                'breadcumps'=>$this-
>getBreadcumps($title)
            ]);
        }

        return $this-
>notFoundAction();
    }

    private function
getBreadcumps($title)
    {
        return [
            $this->translator-
>translate("Home")=>'/',
            $this->translator-
>translate("Departments")=> "#",
            $title => '#',
        ];
    }
}



<?php

namespace App\Department;

class Module
{
    const VERSION = '3.0.3-dev';

    public function getConfig()
    {
        return include __DIR__ .
'/../config/module.config.php';
    }

}


<?php $this->headTitle($title); ?>
<?php $this->headMeta()-
>setName('title', $titleSeo); ?>
<?php $this->headMeta()-
>setName('keywords', $keywordsSeo);
?>
<?php $this->headMeta()-
>setName('description',
$descriptionSeo); ?>
```

```
<!-- Breadcrumb -->
<div class="container">
     <ol class="breadcrumb">
                <?php $end =
end($breadcumps); foreach
($breadcumps as $title => $url): ?>
                    <?php if ($end
=== $url): ?>
                        <li
class="breadcrumb-item
active"><?=$title?></li>
                    <?php else: ?>
                        <li
class="breadcrumb-item"><a
href="/<?=$this->menu()-
>getLangShort()?><?=$url?>"><?=$titl
e?></a></li>
                    <?php endif; ?>
                <?php endforeach; ?>
            </ol>
</div>
<!-- end Breadcrumb -->

<!-- Page Content -->
<div id="page-content">

<div class="container">
        <div class="row">
            <!--MAIN Content-->
            <div class="col-md-8">
                <div id="page-main">
                    <section
id="right-sidebar">

<header><h2><?=$title?></h2></header
>

<?=$content?>
                    </section>
                </div><!-- /#page-
main -->
            </div><!-- /.col-md-8 --
>

            <!--SIDEBAR Content-->
            <div class="col-md-4">
                <div id="page-
sidebar" class="sidebar">
                    <?php if
(!empty($lastNews)): ?>
                    <aside
class="news-small" id="news-small">
                        <header>

<h2><?=$this->translate("Latest
News")?></h2>
                        </header>
                        <div
class="section-content">
                            <?php
foreach ($lastNews as $item): ?>

<article>

<figure class="date"><i class="fa
fa-file-
o"></i><?=$item['publishAt']-
>format("d.m.Y")?></figure>

<header><a href="/<?=$this->menu()-
>getLangShort()?>/news/<?=$item['rou
te']?>"><?=$item['title']?></a></hea
der>

</article>
                            <?php
endforeach;?>
                        </div><!--
/.section-content -->
                        <a
href="/<?=$this->menu()-
>getLangShort()?>/news" class="read-
more"><?=$this->translate("All
News")?></a>
                    </aside><!--
/.news-small -->
                    <?php endif; ?>
                    <aside
id="newsletter">
                        <header>

<h2><?=$this-
>translate("Newsletter")?></h2>
                            <div
class="section-content">
                                <div
class="newsletter">

<?= $this->form()->openTag($form);
?>

<div class="alert" role="alert"
data-result="data-result"
hidden=""></div>

<?= $this->formElement($form-
>get('cuserType')); ?>


<div class="input-group">

<?= $this->formElement($form-
>get('emailSubs')); ?>

<?= $this->formElement($form-
>get('csrf')); ?>

<span class="input-group-btn">

<button type="submit" class="btn"><i
class="fa fa-angle-
right"></i></button>

</span>

</div><!-- /input-group -->

<?= $this->form()->closeTag($form)
?>

</div><!-- /.newsletter -->


</div><!-- /.section-content -->
                        </header>
```

```
                    </aside><!--
/.newsletter -->

                </div><!-- /#sidebar
-->
            </div><!-- /.col-md-4 --
>
        </div><!-- /.row -->
    </div><!-- /.container -->
 </div>
```

```
<?php

namespace Cron\Controller;

use
Zend\Mvc\Controller\AbstractActionCo
ntroller;

class CronController extends
AbstractActionController
{
    private $em;
    private $newsCron;

    protected function getEM()
    {
        return $this->em;
    }

    public function
__construct($entityManager,
$cronNews)
    {
        $this->em = $entityManager;
        $this->newsCron = $cronNews;
    }

    public function newsAction()
    {
        $this->newsCron-
>sendMainNews();
        $this->newsCron->job();
        return;


    }

}
```

```
<?php

namespace User\Controller;

use
Zend\Mvc\Controller\AbstractActionCo
ntroller;
use Zend\View\Model\ViewModel;
use Zend\Authentication\Result;
use Zend\Uri\Uri;
use Zend\Mvc\MvcEvent;
use User\Form\LoginForm;

/**
 * This controller is responsible
for letting the user to log in and
log out.
 */
class AuthController extends
AbstractActionController
{
    /**
     * Entity manager.
     * @var
Doctrine\ORM\EntityManager
     */
    private $entityManager;

    /**
     * Auth manager.
     * @var User\Service\AuthManager
     */
    private $authManager;

    /**
     * Auth service.
     * @var
\Zend\Authentication\AuthenticationS
ervice
     */
    private $authService;

    /**
     * User manager.
     * @var User\Service\UserManager
     */
    private $userManager;

    /**
     * Constructor.
     */
    public function
__construct($entityManager,
$authManager, $authService,
$userManager)
    {
        $this->entityManager =
$entityManager;
        $this->authManager =
$authManager;
        $this->authService =
$authService;
        $this->userManager =
$userManager;
    }

    public function
onDispatch(MvcEvent $e)
    {
        // Вызываем метод базового
класса onDispatch() и получаем ответ
        $response =
parent::onDispatch($e);

        // Устанавливаем
альтернативный лэйаут
        $this->layout()-
>setTemplate('layout/layout_user');

        // Возвращаем ответ
        return $response;
    }
```

```
    /**
     * Authenticates user given
email address and password
credentials.
     */
    public function loginAction()
    {
       // die($this->index()-
>getLang());
        if ($this->identity()!=null)
{
            return $this-
>redirect()->toUrl("/manage");
        }
        // Retrieve the redirect URL
(if passed). We will redirect the
user to this
        // URL after successfull
login.
        $redirectUrl =
(string)$this->params()-
>fromQuery('redirectUrl', '');
        if
(strlen($redirectUrl)>2048) {
            throw new
\Exception(_("Too long redirectUrl
argument passed"));
        }

        // Check if we do not have
users in database at all. If so,
create
        // the 'Admin' user.
        //$this->userManager-
>createAdminUserIfNotExists();

        // Create login form
        $form = new LoginForm();
        $form->get('redirect_url')-
>setValue($redirectUrl);

        // Store login status.
        $isLoginError = false;

        // Check if user has
submitted the form
        if ($this->getRequest()-
>isPost()) {

            // Fill in the form with
POST data
            $data = $this->params()-
>fromPost();

            $form->setData($data);

            // Validate form
            if($form->isValid()) {

                // Get filtered and
validated data
                $data = $form-
>getData();

                // Perform login
attempt.
                try {
                    $result = $this-
>authManager->login($data['email'],
$data['password'],
$data['remember_me']);
                } catch (\Exception
$e) {

                    if ($e-
>getMessage() == "Already logged
in") {
                        $this-
>redirect()->toUrl("/manage");
                    }
                }
                //var_dump($result);
                // Check result.

                if ($result-
>getCode() == Result::SUCCESS) {

                    // Get redirect
URL.
                    $redirectUrl =
$this->params()-
>fromPost('redirect_url', '');

                    if
(!empty($redirectUrl)) {
                        // The below
check is to prevent possible
redirect attack
                        // (if
someone tries to redirect user to
another domain).
                        $uri = new
Uri($redirectUrl);
                        if (!$uri-
>isValid() || $uri->getHost()!=null)
                            throw
new \Exception(_('Incorrect redirect
URL: ') . $redirectUrl);
                    }

                    // If redirect
URL is provided, redirect the user
to that URL;
                    // otherwise
redirect to Home page.

if(empty($redirectUrl)) {
                        return
$this->redirect()->toUrl("/manage");
                    } else {
                        $this-
>redirect()->toUrl($redirectUrl);
                    }
                } else {
                    $isLoginError =
true;
                }
            } else {
                $isLoginError =
true;
            }
        }

        return new ViewModel([
            'form' => $form,
            'isLoginError' =>
$isLoginError,
```

```
            'redirectUrl' =>
$redirectUrl
        ]);
    }

    /**
     * The "logout" action performs
logout operation.
     */
    public function logoutAction()
    {
        $this->authManager-
>logout();

        return $this->redirect()-
>toRoute('login');
    }
}


<?php
namespace User\Controller;

use
Zend\Mvc\Controller\AbstractActionCo
ntroller;
use Zend\View\Model\ViewModel;
use User\Form\UserForm;
use User\Form\PasswordChangeForm;
use User\Form\PasswordResetForm;
use Zend\Mvc\MvcEvent;

/**
 * This controller is responsible
for user management (adding,
editing,
 * viewing users and changing user's
password).
 */
class UserController extends
AbstractActionController
{
    /**
     * Entity manager.
     * @var
Doctrine\ORM\EntityManager
     */
    private $entityManager;

    /**
     * User manager.
     * @var User\Service\UserManager
     */
    private $userManager;

    /**
     * Constructor.
     */
    public function
__construct($entityManager,
$userManager)
    {
        $this->entityManager =
$entityManager;
        $this->userManager =
$userManager;
    }

    public function
onDispatch(MvcEvent $e)
    {
        // Вызываем метод базового
класса onDispatch() и получаем ответ
        $response =
parent::onDispatch($e);

        // Устанавливаем
альтернативный лэйаут
        $this->layout()-
>setTemplate('layout/layout_user');

        // Возвращаем ответ
        return $response;
    }


    /**
     * This action displays a page
allowing to change user's password.
     */
    public function
changePasswordAction()
    {
        $id = (int)$this->params()-
>fromRoute('id', -1);
        if ($id<1) {
            $this->getResponse()-
>setStatusCode(404);
            return;
        }

        $user = $this-
>entityManager-
>getRepository("Model\Entity\Users")
                ->find($id);

        if ($user == null) {
            $this->getResponse()-
>setStatusCode(404);
            return;
        }

        // Create "change password"
form
        $form = new
PasswordChangeForm('change');

        // Check if user has
submitted the form
        if ($this->getRequest()-
>isPost()) {

            // Fill in the form with
POST data
            $data = $this->params()-
>fromPost();

            $form->setData($data);

            // Validate form
            if($form->isValid()) {

                // Get filtered and
validated data
                $data = $form-
>getData();
```

```
                // Try to change
password.
                if (!$this-
>userManager->changePassword($user,
$data)) {
                    $this-
>flashMessenger()->addErrorMessage(

_('Sorry, the old password is
incorrect. Could not set the new
password.'));
                } else {
                    $this-
>flashMessenger()-
>addSuccessMessage(

_('Changed the password
successfully.'));
                }

                // Redirect to
"view" page
                return $this-
>redirect()->toRoute('users',

['action'=>'view', 'id'=>$user-
>getId()]);
            }
        }

        return new ViewModel([
            'user' => $user,
            'form' => $form
        ]);
    }

    /**
     * This action displays the
"Reset Password" page.
     */
    public function
resetPasswordAction()
    {
        // Create form
        $form = new
PasswordResetForm();

        // Check if user has
submitted the form
        if ($this->getRequest()-
>isPost()) {

            // Fill in the form with
POST data
            $data = $this->params()-
>fromPost();

            $form->setData($data);

            // Validate form
            if($form->isValid()) {

                // Look for the user
with such email.
                $user = $this-
>entityManager-
>getRepository("Model\Entity\Users")
                    -
>findOneByEmail($data['email']);
                if ($user!=null) {
                    // Generate a
new password for user and send an E-
mail
                    // notification
about that.
                    $this-
>userManager-
>generatePasswordResetToken($user);

                    // Redirect to
"message" page
                    return $this-
>redirect()->toRoute('users',

['action'=>'message',
'id'=>'sent']);
                } else {
                    return $this-
>redirect()->toRoute('users',

['action'=>'message',
'id'=>'invalid-email']);
                }
            }
        }

        return new ViewModel([
            'form' => $form
        ]);
    }

    /**
     * This action displays an
informational message page.
     * For example "Your password
has been resetted" and so on.
     */
    public function messageAction()
    {
        // Get message ID from
route.
        $id = (string)$this-
>params()->fromRoute('id');

        // Validate input argument.
        if($id!='invalid-email' &&
$id!='sent' && $id!='set' &&
$id!='failed') {
            throw new
\Exception(_('Invalid message ID
specified'));
        }

        return new ViewModel([
            'id' => $id
        ]);
    }

    /**
     * This action displays the
"Reset Password" page.
     */
    public function
setPasswordAction()
    {
        $token = $this->params()-
>fromQuery('token', null);
```

```
        // Validate token length
        if ($token!=null &&
(!is_string($token) ||
strlen($token)!=32)) {
            throw new
\Exception(_('Invalid token type or
length'));
        }

        if($token===null ||
           !$this->userManager-
>validatePasswordResetToken($token))
{
            return $this-
>redirect()->toRoute('users',

['action'=>'message',
'id'=>'failed']);
        }

        // Create form
        $form = new
PasswordChangeForm('reset');

        // Check if user has
submitted the form
        if ($this->getRequest()-
>isPost()) {

            // Fill in the form with
POST data
            $data = $this->params()-
>fromPost();

            $form->setData($data);

            // Validate form
            if($form->isValid()) {

                $data = $form-
>getData();

                // Set new password
for the user.
                if ($this-
>userManager-
>setNewPasswordByToken($token,
$data['new_password'])) {

                    // Redirect to
"message" page
                    return $this-
>redirect()->toRoute('users',

['action'=>'message', 'id'=>'set']);
                } else {
                    // Redirect to
"message" page
                    return $this-
>redirect()->toRoute('users',

['action'=>'message',
'id'=>'failed']);
                }
            }
        }

        return new ViewModel([
            'form' => $form
        ]);
    }
}


<?php
namespace User\Validator;

use
Zend\Validator\AbstractValidator;
use User\Entity\User;
/**
 * This validator class is designed
for checking if there is an existing
user
 * with such an email.
 */
class UserExistsValidator extends
AbstractValidator
{
    /**
     * Available validator options.
     * @var array
     */
    protected $options = array(
        'entityManager' => null,
        'user' => null
    );

    // Validation failure message
IDs.
    const NOT_SCALAR  = 'notScalar';
    const USER_EXISTS =
'userExists';

    /**
     * Validation failure messages.
     * @var array
     */
    protected $messageTemplates =
array(
        self::NOT_SCALAR  => "The
email must be a scalar value",
        self::USER_EXISTS  =>
"Another user with such an email
already exists"
    );

    /**
     * Constructor.
     */
    public function
__construct($options = null)
    {
        // Set filter options (if
provided).
        if(is_array($options)) {

if(isset($options['entityManager']))
                $this-
>options['entityManager'] =
$options['entityManager'];

if(isset($options['user']))
                $this-
>options['user'] = $options['user'];
        }
```

```
        // Call the parent class
constructor

parent::__construct($options);
    }

    /**
     * Check if user exists.
     */
    public function isValid($value)
    {
        if(!is_scalar($value)) {
            $this-
>error(self::NOT_SCALAR);
            return false;
        }

        // Get Doctrine entity
manager.
        $entityManager = $this-
>options['entityManager'];

        $user = $entityManager-
>getRepository("Model\Entity\Users")
                -
>findOneByEmail($value);

        if($this-
>options['user']==null) {
            $isValid =
($user==null);
        } else {
            if($this-
>options['user']->getEmail()!=$value
&& $user!=null)
                $isValid = false;
            else
                $isValid = true;
        }

        // If there were an error,
set error message.
        if(!$isValid) {
            $this-
>error(self::USER_EXISTS);
        }

        // Return validation result.
        return $isValid;
    }
}
```

```
<?php
/**
 * @link
http://github.com/zendframework/Zend
SkeletonApplication for the
canonical source repository
 * @copyright Copyright (c) 2005-
2016 Zend Technologies USA Inc.
(http://www.zend.com)
 * @license
http://framework.zend.com/license/ne
w-bsd New BSD License
 */
```

```
namespace UserTest\Controller;

use User\Controller\UserController;
use Zend\Stdlib\ArrayUtils;
use
Zend\Test\PHPUnit\Controller\Abstrac
tHttpControllerTestCase;

class UserControllerTest extends
AbstractHttpControllerTestCase
{
    public function setUp()
    {
        // The module configuration
should still be applicable for
tests.
        // You can override
configuration here with test case
specific values,
        // such as sample view
templates, path stacks,
module_listener_options,
        // etc.
        $configOverrides = [];

        $this-
>setApplicationConfig(ArrayUtils::me
rge(
            include __DIR__ .
'/../../../../config/application.con
fig.php',
            $configOverrides
        ));

        parent::setUp();
    }

    public function
testIndexActionCanBeAccessed()
    {
        $this->dispatch('/manage',
'GET');
        $this-
>assertResponseStatusCode(200);
        $this-
>assertControllerName(UserController
::class);
        $this-
>assertControllerClass('UserControll
er');
        $this-
>assertMatchedRouteName('login');
    }

    public function
testIndexActionViewModelTemplateRend
eredWithinLayout()
    {
        $this->dispatch('/', 'GET');
        $this-
>assertQuery('.container
.jumbotron');
    }

    public function
testInvalidRouteDoesNotCrash()
    {
        $this-
>dispatch('/invalid/route', 'GET');
```

```
        $this->assertResponseStatusCode(404);
    }
}
```

```
<?php
namespace User\Form;

use Zend\Form\Form;
use Zend\Form\Fieldset;
use Zend\InputFilter\InputFilter;

/**
 * This form is used when changing user's password (to collect user's old password
 * and new password) or when resetting user's password (when user forgot his password).
 */
class PasswordChangeForm extends Form
{
    // There can be two scenarios - 'change' or 'reset'.
    private $scenario;

    /**
     * Constructor.
     * @param string $scenario Either 'change' or 'reset'.
     */
    public function __construct($scenario)
    {
        // Define form name
        parent::__construct('password-change-form');

        $this->scenario = $scenario;

        // Set POST method for this form
        $this->setAttribute('method', 'post');

        $this->addElements();
        $this->addInputFilter();
    }

    /**
     * This method adds elements to form (input fields and submit button).
     */
    protected function addElements()
    {
        // If scenario is 'change', we do not ask for old password.
        if ($this->scenario == 'change') {

            // Add "old_password" field
            $this->add([
                'type'  => 'password',
                'name' => 'old_password',
                'options' => [
                    'label' => 'Old Password',
                ],
            ]);
        }

        // Add "new_password" field
        $this->add([
            'type'  => 'password',
            'name' => 'new_password',
            'options' => [
                'label' => 'New Password',
            ],
        ]);

        // Add "confirm_new_password" field
        $this->add([
            'type'  => 'password',
            'name' => 'confirm_new_password',
            'options' => [
                'label' => 'Confirm new password',
            ],
        ]);

        // Add the CSRF field
        $this->add([
            'type' => 'csrf',
            'name' => 'csrf',
            'options' => [
                'csrf_options' => [
                'timeout' => 600
                ]
            ],
        ]);

        // Add the Submit button
        $this->add([
            'type'  => 'submit',
            'name' => 'submit',
            'attributes' => [
                'value' => 'Change Password'
            ],
        ]);
    }

    /**
     * This method creates input filter (used for form filtering/validation).
     */
    private function addInputFilter()
    {
        // Create main input filter
        $inputFilter = new InputFilter();
        $this->setInputFilter($inputFilter);
```

```
        if ($this->scenario ==
'change') {

            // Add input for
"old_password" field
            $inputFilter->add([
                    'name'     =>
'old_password',
                    'required' =>
true,
                    'filters'  => [
                    ],
                    'validators' =>
[
                        [
                            'name'
=> 'StringLength',

'options' => [

'min' => 6,

'max' => 64
                            ],
                        ],
                    ],
                ]);
        }

        // Add input for
"new_password" field
        $inputFilter->add([
                'name'     =>
'new_password',
                'required' => true,
                'filters'  => [
                ],
                'validators' => [
                    [
                        'name'     =>
'StringLength',
                        'options' =>
[
                            'min' =>
6,
                            'max' =>
64
                        ],
                    ],
                ],
            ]);

        // Add input for
"confirm_new_password" field
        $inputFilter->add([
                'name'     =>
'confirm_new_password',
                'required' => true,
                'filters'  => [
                ],
                'validators' => [
                    [
                        'name'     =>
'Identical',
                        'options' =>
[
                            'token'
=> 'new_password',
                        ],
                    ],
                ]);
    }
}
```

```
<?php
namespace Manage\Users\Form;

use Zend\Form\Form;

class UsersForm extends Form
{
    // Конструктор.
    public function __construct()
    {
        // Определяем имя формы

parent::__construct('contact-form');

        // Устанавливаем метод POST
для этой формы
        $this-
>setAttribute('method', 'post');
        $this->setAttribute('name',
'manageUsersForm');

        // (Опционально) задаем
действие для этой формы
        $this-
>setAttribute('action',
'/contactus');


        $this->addElements();


    }

    private function addElements()
    {
        $this->add([
            'name' => 'id',
            'type' => 'hidden',
   ]);

        $this->add([
       'type'  => 'text',
            'name' => 'email',
            'attributes' => [
                'id' => 'email',
                'class'    => 'form-
control',
            ],
            'options' => [
                'label' => 'Email',
            ],
        ]);

        $this->add([
            'type'  => 'text',
            'name' => 'phone',
            'attributes' => [
              'id' => 'subject',
              'class'=> 'form-
control',
```

```
            ],
            'options' => [
                'label' =>
_("Phone"),
            ],
        ]);

        $this->add([
            'type'  => 'text',
            'name' => 'firstname',
            'attributes' => [
              'id' => 'subject',
              'class'=> 'form-
control',
            ],
            'options' => [
                'label' => 'Имя по-
английски',
            ],
        ]);

        $this->add([
            'type'  => 'text',
            'name' => 'firstnameRu',
            'attributes' => [
              'id' => 'subject',
              'class'  => 'form-
control',
            ],
            'options' => [
                'label' => 'Имя по-
русски',
            ],
        ]);

        $this->add([
            'type'  => 'text',
            'name' => 'lastname',
            'attributes' => [
              'id' => 'subject',
              'class'  => 'form-
control',
            ],
            'options' => [
                'label' => 'Фамилия
по-английски',
            ],
        ]);

        $this->add([
            'type'  => 'text',
            'name' => 'lastnameRu',
            'attributes' => [
              'id' => 'subject',
              'class'  => 'form-
control',
            ],
            'options' => [
                'label' => 'Фамилия
по-английски',
            ],
        ]);


        $this->add([
            'type'  => 'select',
            'name' => 'status',
            'attributes' => [
                'id' => 'body',
                'class'    => 'form-
control',
            ],
            'options' => [
      'value_options' => [
                    ''           =>
_(' - Please, select status - '),
                    '1'    =>
_('Active'),
                    '2'    =>
_('Inactive'),
      ],
            ],
        ]);



        $this->add([
            'type'  => 'password',
            'name' => 'password',
            'attributes' => [
              'id' => 'subject',
              'class'  => 'form-
control',
            ],
            'options' => [
                'label' =>
_("Password"),
            ],
        ]);

        $this->add([
            'type'  => 'password',
            'name' =>
'password_verify',
            'attributes' => [
              'id' => 'subject',
              'class'  => 'form-
control',
            ],
            'options' => [
                'label' =>
_("Password verify"),
            ],
        ]);

        $this->add([
            'type'  => 'file',
            'name' => 'photo',
            'attributes' => [
              'id' => 'subject',
              'class'  => 'form-
control',
            ],
        ]);

        $this->add([
            'type'  => 'submit',
            'name' => 'submit',
            'attributes' => [
                'value' =>
'Сохранить',
            ],
        ]);
    }


    // Этот метод создает фильтр
входных данных (используемый для
```

```
фильтрации/валидации).
    private function
addInputFilter()
    {
        $inputFilter = new
InputFilter();
        $this-
>setInputFilter($inputFilter);

        $inputFilter->add([
            'name'     => 'email',
            'required' => true,
            'filters'  => [
                ['name' =>
'StringTrim'],
            ],
            'validators' => [
                [
                    'name' =>
'EmailAddress',
                    'options' => [
                        'allow' =>
\Zend\Validator\Hostname::ALLOW_DNS,
                        'useMxCheck'
=> false,
                    ],
                ],
            ],
        ]);

        $inputFilter->add([
            'name'     => 'subject',
            'required' => true,
            'filters'  => [
                ['name' =>
'StringTrim'],
                ['name' =>
'StripTags'],
                ['name' =>
'StripNewlines'],
            ],
            'validators' => [
                [
                    'name' =>
'StringLength',
                    'options' => [
                        'min' => 1,
                        'max' => 128
                    ],
                ],
            ],
        ]);

        $inputFilter->add([
            'name'     => 'body',
            'required' => true,
            'filters'  => [
                ['name' =>
'StripTags'],
            ],
            'validators' => [
                [
                    'name' =>
'StringLength',
                    'options' => [
                        'min' => 1,
                        'max' =>
4096
                    ],
                ],
            ],
        ]);
    }

}
```

```
<?php

namespace Manage\Teachers\Service;

use
Manage\Teachers\Concepts\TeachersCon
cepts as Concepts;

class TeachersGrid
{
   private $em;
        private $translator;
        private $thumbnail;

        public function
__construct($em, $translator,
$thumbnail)
        {
            $this->em = $em;
            $this->translator =
$translator;
            $this->thumbnail =
$thumbnail;
        }

   public function
getIndex($request)
   {

            $content   = [];
            $condition     = [];
            $limit      =
$request['rows'];
            $offset     =
$request['rows'] * $request['page']
- $request['rows'];

            $where = "p.status <>
'trash'";

            $postsIds  = [];

            $filter = false;


if(!empty($request['filters']['statu
s'])) {

                $where .=
!empty($where) ?
                    " AND " : "";

                $where .= "p.status
= '" . $request['filters']['status']
. "'";
                $filter  = true;

            }
```

```
if(!empty($request['filters']['searc
h'])) {
                $where .=
!empty($where) ?
                    " AND " : "";

                $where .= "(";
                $where .= "p.nameRu
LIKE '%" .
$request['filters']['search'] .
"%'";
                $where .= " OR ";
                $where .=
"p.descriptionRu LIKE '%" .
$request['filters']['search'] .
"%'";
                $where .= " OR ";
                $where .= "p.nameEn
LIKE '%" .
$request['filters']['search'] .
"%'";
                $where .= " OR ";
                $where .=
"p.descriptionEn LIKE '%" .
$request['filters']['search'] .
"%'";
                $where .= ")";
                $filter  = true;
            }

            $count = $this->em
                -
>createQueryBuilder()
                -
>select("count(p.id)")
                -
>from(Concepts::ENTITY_TEACHERS,
"p");

            if(!empty($where)) {
                $count = $count-
>where($where);
            }

            $count = $count-
>getQuery()
                -
>getSingleScalarResult();

            $entitiesPosts = $this-
>em
                -
>createQueryBuilder()
                ->select("p.id",
"p.nameRu as title","p.photo",
"p.status")
                -
>from(Concepts::ENTITY_TEACHERS,
"p");

                    if
(!empty($where)) {

$entitiesPosts = $entitiesPosts-
>where($where);
                    }

                    $entitiesPosts =
$entitiesPosts
                                -
>orderBy('p.id', 'DESC')
                                -
>groupBy('p.id')
                                -
>setFirstResult($offset)
                                -
>setMaxResults($limit)
                                -
>getQuery()
                                -
>getArrayResult();

                    $repositoryPost
= $this->em-
>getRepository(Concepts::ENTITY_TEAC
HERS);

                    foreach
($entitiesPosts as $entityPosts) {

$offset++;

                            $title =
$entityPosts['title'];


$content[] = [

'id'                            =>
$offset,

'photo'                         =>
"<center><img src='".$this-
>thumbnail-
>getJust100n100($entityPosts['photo'
])."'></center>",

'name'       => '<b> ' . $title .
'</b>',

'status'          => $this-
>getIndexStatuses($entityPosts['stat
us']),

'actions'         => $this-
>getIndexActions($entityPosts,
$allowDeleteNews),
                            ];
                    }

                    return ['count'
=> $count, 'rows' => $content];
    }



    private function
getIndexTime($objParams)
    {
        $result = '';

        $updateAt =
$objParams['createAt'];

        $updateAt   = $this-
>getLinearDate($updateAt);

        return "<center><span
```

```
style='font-size:8pt'>".$updateAt."</span></center>";
    }


    private function getIndexStatuses($status)
    {
        $result = $status;

        switch ($status) {
            case 'publish':
                $result = '<span class="label label-success">' . _('Active') . '</span>';
                break;

            case 'planing':
                $result = '<span class="label label-warning">' . _('Плановая') . '</span>';
                break;

            case 'note':
                $result = '<span class="label label-info">' . _('Inactive') . '</span>';
                break;

            case 'pending':
                $result = '<span class="label label-primary">' . _('На проверке') . '</span>';
                break;

            case 'private':
                $result = '<span class="label label-danger">' . _('На доработку') . '</span>';
                break;

            case 'auto-note':
                $result = '<span class="label label-info">' . _('Auto-черновик') . '</span>';
                break;

            default:
                $result = '<span class="label label-default">' . $status . '</span>';
                break;
        }

        return '<center>' . $result . '</center>';
    }

   private function getIndexActions($entity, $allowDeleteNews)
   {
       $id = $entity['id'];
                $view = $edit = $remove = '';
       $out = '';

       $out .= '<input type="hidden" name="id" value="' . $id . '">';


       $out .= '<div class="btn-toolbar">';

       $view = '<a class="btn btn-primary btn-xs action-button" '
           . 'modal-trigger="managePostsView" data-modal-url="/manage/teachers/modal-view/'
           . $id
           . '"><i class="el el-eye-open"></i></a>';

                        $edit = '<a class="btn btn-warning btn-xs action-button" '
                                    . ' href="/manage/teachers/edit/'
                                    . $id
                                    . '" target="_blank"><i class="fa fa-paint-brush fa-lg"></i></a>';



                        $remove = '<a class="btn btn-danger btn-xs action-button" '
                                    . 'modal-trigger="managePostsRemove" data-modal-url="/manage/teachers/modal-remove/'
                                    . $id
                                    . '"><i class="fa fa-trash fa-lg"></i></a>';

       $out .= $view;
       $out .= $edit;
       $out .= $remove;

       $out .= "</div>";

       return $out;
   }

    private function getLinearDate($date)
    {
        if(empty($date)) {
            return '';
        }

        $today = $this->translator->translate("Сегодня");
        $tommorow = $this->translator->translate("Завтра");
        $yesterday = $this->translator->translate("Вчера");

   $publishDate = $date->format('Y-m-d');

   $todayDate = (new \DateTime('now'))->format('Y-m-d');
```

```
   if($publishDate === (new
\DateTime('now'))->format('Y-m-d'))
{
            return $date-
>format('H:i') . ', '.$today;
   } elseif($publishDate === (new
\DateTime('now'))->modify('+1 day')-
>format('Y-m-d')) {
            return $date-
>format('H:i') . ', '.$tommorow;
   } elseif($publishDate === (new
\DateTime('now'))->modify('-1 day')-
>format('Y-m-d')) {
            return $date-
>format('H:i') . ', '.$yesterday;
   } elseif($date->format('Y') !==
(new \DateTime('now'))->format('Y'))
{
            $month = $this-
>getUkrainianMonth($date-
>format('M'));
            return $date-
>format('H:i, d') . ' ' . $month . '
' . $date->format('Y');
   } else {
            $month = $this-
>getUkrainianMonth($date-
>format('M'));
            return $date-
>format('H:i, d') . ' ' . $month;
   }

   return;
    }

    private function
getUkrainianMonth($month)
    {
   $months = [
            'Jan' => $this-
>translator->translate('January'),
            'Feb' => $this-
>translator->translate('February'),
            'Mar' => $this-
>translator->translate('March'),
            'Apr' => $this-
>translator->translate('April'),
            'May' => $this-
>translator->translate('May'),
            'Jun' => $this-
>translator->translate('June'),
            'Jul' => $this-
>translator->translate('July'),
            'Aug' => $this-
>translator->translate('August'),
            'Sep' => $this-
>translator->translate('September'),
            'Oct' => $this-
>translator->translate('October'),
            'Nov' => $this-
>translator->translate('November'),
            'Dec' => $this-
>translator->translate('December'),
   ];

   return $months[$month];
    }
```

```
    public function getEM()
    {
        return $this->em;
    }
}


<?php

namespace Manage\Teachers\Service;

use
Manage\Teachers\Concepts\TeachersCon
cepts as Concepts;

class TeachersHelper
{
   private $em;
        private $tranlsator;

        public function
__construct($em, $translator)
        {
            $this->em = $em;
            $this->tranlsator =
$translator;
        }

   public function
selectEmployees($grid = false)
   {
      $entitiesEmployee = $this->em
          -
>getRepository(Concepts::ENTITY_EMPL
OYEE)
         ->findBy(['status'=> 1]);
                $name = $this-
>tranlsator->translate(" - Filter:
Employee - ");
      $resultArray = [
         "" => $name,
      ];

      if(!empty($entitiesEmployee))
{
         foreach ($entitiesEmployee
as $entityEmployee) {

$resultArray[$entityEmployee-
>getId()] = $entityEmployee-
>getEmail();
         }
      }

      return $resultArray;
   }



        public function
selectStatuses()
   {

               $allowStatuses =
['note', 'publish'];
          $resultArray = [
         "" => $this->tranlsator-
>translate(" - Filter: Status - "),
```

```
        ];

        if(!empty($allowStatuses)) {
            foreach ($allowStatuses as
$item) {
                switch ($item) {

case 'publish':

$resultArray[$item] =  $this-
>tranlsator->translate('Active');

break;

case 'note':

$resultArray[$item] =  $this-
>tranlsator->translate('Inactive');

break;

default:

$resultArray[$item] = $item;

break;
                                    }
            }
        }
        return $resultArray;
    }
}
```

```
<?php

namespace Manage\Teachers\Concepts;

class TeachersConcepts
{
    const TEMP_CREATE_EDIT
= "manage/teachers/create-
edit.phtml";
    const TEMP_REMOVE
=
"manage/teachers/modal/remove.phtml"
;
    const TEMP_VIEW
=
"manage/teachers/modal/view.phtml";

    const AJAX_SAVE_URL
= "/manage/teachers/ajax-save";
    const AJAX_REMOVE_URL
= "/manage/teachers/ajax-remove";

    const ROUTE_TEACHERS
= 'manage\teachers';
    const ROUTE_404
= 'app\404';

    const ENTITY_TEACHERS
= "Model\Entity\Teachers";
    const ENTITY_EMPLOYEE
= "Model\Entity\Users";

    const GRID_TEACHERS
= "Manage\Grid\Teachers";

    const SERVICE_TEACHERS
= "Manage\Service\Teachers";
    const
SERVICE_TEACHERS_EXTENSIONS_LOCKER
=
"Manage\Service\Teachers\Extensions\
Locker";

    const HELPER_TEACHERS
= "Manage\Helper\Teachers";

    const STATUS_PLANING
= "planing";
    const STATUS_PUBLISH
= "publish";

    private $_translator;

    public function
__construct($translator)
    {
        $this->_translator =
$translator;
    }

    public function
getJsonFailStreamSave()
    {
        return [
            "result" => false,
      "message" => [
         $this->_translator-
>translate("This name already
exist"),
        ],
            ];
    }

    public function
getJsonSuccessTeachersSave($id)
    {
        return [
            "result"   => true,
            "message"  => [
      $this->_translator-
>translate("Teacher save
successfully"),
            ],
            "redirect"     =>
"/manage/teachers/edit/" . $id,
        ];
    }


    public function
getJsonSaveTeachersRouteFail()
    {
        return [
            "result"   => false,
            "message"  => [
                $this->_translator-
>translate("Такой адрес уже
существует. Смените название
новости."),
            ],
    ];
    }
```

```
    public function
jsonTeachersRemoveFail()
    {
   return [
            "result"    => false,
            "message"   => [
      $this->_translator-
>translate("Data for remove
invalid"),
            ],
   ];
    }

    public function
jsonTeachersRemoveSuccess()
    {
   return [
            "result"    => true,
            "message"   => [
      $this->_translator-
>translate("Teacher remove
successfully"),
            ],
   ];
    }
}


<?php

namespace
Manage\Settings\Controller;

use
Zend\Mvc\Controller\AbstractActionCo
ntroller;
use Zend\View\Model\ViewModel;
use Zend\Mvc\MvcEvent;
use Zend\View\Model\JsonModel;

class SettingsController extends
AbstractActionController
{
    private $em;
    private $translator;

    public function
onDispatch(MvcEvent $e)
    {
        $response =
parent::onDispatch($e);
        $this->layout()-
>setTemplate('layout/layout_admin');
        return $response;
    }

    protected function getEM()
    {
        return $this->em;
    }


    public function
__construct($entityManager,
$translator)
    {
        $this->em = $entityManager;
        $this->translator =
$translator;
    }

    public function indexAction()
    {
        $em = $this->getEM();

        $settings = $em-
>getRepository("Model\Entity\Setting
s")->findAll();
        $lang = $this->extra()-
>getLang();

        return (new ViewModel())-
>setTemplate("manage/settings/index/
index")
            ->setVariables([

'settings'=>$settings,
                'lang'=>$lang,
                'breadcumps'=>$this-
>getBreadcumps(),
            ]);
    }

    private function getBreadcumps()
: array
    {
        return [
            $this->translator-
>translate("Home")=>'/',
            $this->translator-
>translate("Settings")=>'/settings',
        ];
    }

    public function
ajaxUpdateAction()
    {
        $request = $this-
>getRequest();

        if ($request->isPost() &&
$request->isXmlHttpRequest()) {

            $postParams = $request-
>getPost();
            $lang = $this->extra()-
>getLang();

            $model = $this->getEM()-
>getRepository("Model\Entity\Setting
s")->find($postParams['id']);
            if ($model) {

                $model-
>setValue($postParams['value']);

                $this->getEM()-
>flush();
                return new
JsonModel(['status']);
            }
        }
        $this->notFoundAction();
    }
```

```
}
```

```
<?php

namespace Manage\Menu\Controller;

use Model\Entity\Menu;
use
Zend\Mvc\Controller\AbstractActionCo
ntroller;
use Zend\View\Model\ViewModel;
use Zend\Mvc\MvcEvent;
use Zend\View\Model\JsonModel;

class MenuController extends
AbstractActionController
{
    private $em;
    private $translator;

    public function
onDispatch(MvcEvent $e)
    {
        $response =
parent::onDispatch($e);
        $this->layout()-
>setTemplate('layout/layout_admin');
        return $response;
    }

    protected function getEM()
    {
        return $this->em;
    }


    public function
__construct($entityManager,
$translator)
    {
        $this->em = $entityManager;
        $this->translator =
$translator;
    }

    public function indexAction()
    {
        $menus = $this->getEM()-
>getRepository(Menu::class)-
>findAll();

        return (new ViewModel())-
>setVariables([
            'breadcumps'=>$this-
>getBreadcumps(),
            'menus'=>$menus,
        ]);
    }


    public function editAction()
    {

        $identifier = $this-
>getEvent()
            ->getRouteMatch()
            ->getParam('id');

            if(!empty($identifier))
{
                /** @var
\Model\Entity\Pages */

                $modules = $this-
>getEM()-
>getRepository("Model\Entity\Menu")-
>getModules();
                /** @var Menu $menu
*/
                $menu = $this-
>getEM()-
>getRepository("Model\Entity\Menu")-
>find($identifier);

                if (!$menu) return
$this->notFoundAction();

                $lang = $this-
>extra()->getLang();

                $title = $lang ==
"ru_RU" ? $menu->getNameRu() :
$menu->getNameEn();

                return (new
ViewModel())-
>setTemplate("manage/menu/menu/creat
e-edit.phtml")
                    ->setVariables([

'breadcumps'=>$this-
>getBreadcumpsEdit($title),

'modules'=>$modules,
                        'menu' =>
$menu,
                    ]);
            }

        return $this-
>notFoundAction();
    }

    public function ajaxSaveAction()
    {
        $request = $this-
>getRequest();

        if($request->isPost() &&
$request->isXmlHttpRequest())
        {
            $postParams = $request-
>getPost();

            $entity = $this->getEM()
                -
>getRepository("Model\Entity\Menu")
                -
>save($postParams);

            return new
JsonModel(['result'=>true]);
        }
```

```
        return $this->notFoundAction();
    }


    private function getBreadcumps()
    {
        return [
            $this->translator->translate("Home")=>'/manage/index',
            $this->translator->translate("Menu editor")=>'/manage/menu/index',
            $this->translator->translate("Grid")=> '#',
        ];
    }

    private function getBreadcumpsEdit($title)
    {
        return [
            $this->translator->translate("Home")=>'/manage/index',
            $this->translator->translate("Menu editor")=>'/manage/menu/index',
            $title=> '#',
        ];
    }
}
```

```
<?php

namespace Manage\Subscribers\Report;

class Report
{
    public function setReport($data)
{

        $xls = \PHPExcel_IOFactory::load(getcwd() . '/public/assets/excel/report_subscribers.xlsx');

        $xls->setActiveSheetIndex(0);
        // Get active sheet
        $sheet = $xls->getActiveSheet();
         // Add title in the sheet
        $sheet->setTitle('Подписчики');

        // Write information about report
        $sheet->setCellValue("C2","За период:".$data['date']);

        $arr = $data['subscribers'];
        $i = 4; // start position

        foreach ($arr as $item) {
            // feel each line
            $sheet->setCellValue("A".$i, $item['id']);
            $sheet->setCellValue("B".$i, $item['createAt']->format("d.m.Y H:i:s"));
            $sheet->setCellValue('C'.$i, $item['email']);
            $sheet->setCellValue('D'.$i, $item['ip']);
            $sheet->setCellValue('E'.$i, ($item['isActive']=="1") ? "Да" : "Нет");

            $i++;
        }

        $xls->setActiveSheetIndex(0);

        $href = $this->saveResult($xls, $data);
        return $href;
    }

    private function convertDate($date) {

        return date('d.m.Y',strtotime($date));

    }

    private function convertNameOfDay($day) {

        $days = [
            'Monday'=> 'Понедельник',
            'Tuesday'=> 'Вторник',
            'Wednesday'=> 'Среда',
            'Thursday'=> 'Четверг',
            'Friday'=> 'Пятница',
            'Saturday'=> 'Суббота',
            'Sunday'=> 'Воскресенье',
        ];
        return isset($days[$day]) ? $days[$day] : $day;

    }

    public function saveResult($xls, $data) {

        $fileName = "Subscribers_".date("d_m_Y")."_".time();
        $dir = getcwd().'/public/uploads/exports/'.date("m")."/";

        if(!is_dir($dir)) {
            mkdir($dir, 0777, true);
        }
        $objWriter =
```

```
\PHPExcel_IOFactory::createWriter($x
ls, 'Excel5');
        $objWriter-
>save($dir.$fileName.'.xls');

        return
'/uploads/exports/'.date("m").'/'.$f
ileName.".xls";
    }
}
```

```
<?php

namespace
Manage\Subscribers\Controller;

use
Manage\Subscribers\Form\Subscribers
as Form;
use Model\Entity\Subscribers;
use Zend\View\Model\ViewModel;
use Zend\View\Model\JsonModel;
use Zend\Mvc\MvcEvent;

class SubscribersController extends
\Zend\Mvc\Controller\AbstractActionC
ontroller
{
    private $em;
    private $translator;
    private $grid;

    public function __construct($em,
$translator, $grid)
    {
        $this->em = $em;
        $this->translator =
$translator;
        $this->grid = $grid;
    }

    public function
onDispatch(MvcEvent $e)
    {
        $response =
parent::onDispatch($e);
        $this->layout()-
>setTemplate('layout/layout_admin');
        return $response;
    }

    public function indexAction()
    {
        $form = new Form();

        return (new
ViewModel(['form' => $form]))-
>setTemplate("manage/subscribers/ind
ex.phtml")
            -
>setVariables(['breadcumps'=>$this-
>getBreadcumps()]);
        }

    private function getBreadcumps()
    {
        return [
            $this->translator-
>translate("Home")=>'/manage/index',
            $this->translator-
>translate("Subscribers")=>
'/manage/subscribers/index',
            $this->translator-
>translate("Grid")=> "#",
        ];
    }

    public function
modalRemoveAction()
    {
        $request = $this-
>getRequest();

        if ($request->isPost() &&
$request->isXmlHttpRequest()) {

            $id = $this->getEvent()
                ->getRouteMatch()
                ->getParam('id');

            $entityPages = $this-
>getEM()
                -
>find("Model\Entity\Subscribers",
$id);

            $form = (new
Form("/manage/subscribers/ajax-
remove"))
                ->setData([
                    "id" =>
$entityPages->getId(),
            ]);

            return (new ViewModel())
                ->setTerminal(true)
                -
>setTemplate("manage/subscribers/mod
al/remove.phtml")
                ->setVariables([
                    'item' =>
$entityPages,
                    'form' => $form,
                ]);
        }

        $this->redirect()-
>toRoute("manage\index");
    }


    /*
     * Make reports by all or chosen
posts
     *
     */
    public function
modalReportAction() {
        $request = $this-
>getRequest();

        if ($request->isPost()&&
$request->isXmlHttpRequest()) {

            return (new
ViewModel())->setTerminal(true)-
```

```
>setTemplate('manage/subscribers/mod
al/report.phtml');
        }
            $this->redirect()-
>toRoute('manage\index');
    }

    /*
    * Make export to excel by dates
    */
    public function
ajaxGetReportAction()
    {
        $request = $this-
>getRequest();

        if ($request->isPost() &&
$request->isXmlHttpRequest()) {

            $data = $request-
>getPost();

            $result = [];

            if (empty($data['date-
start']) || empty($data['date-
end']))
                return new
JsonModel(['result'=>false,'msg'=>$t
his->translator->translate("Pick 2
dates")]);

            $data['date-start'] .= '
00:00:00';
            $data['date-end'] .= '
23:59:59';
            $from = new
\DateTime($data['date-start']);
            $to = new
\DateTime($data['date-end']);

            if ($from > $to) return
new JsonModel(['result'=>false,
"msg"=> $this->translator-
>translate("Pick the correct
dates")]);


            $subscribers = $this-
>getEM()
                -
>createQueryBuilder()
                -
>select('t.email','t.id','t.ip','t.i
sActive', 't.createAt')
                -
>from('Model\Entity\Subscribers',
't')
                ->where("t.createAt
>= :date_start")
                -
>andWhere("t.createAt <= :date_end
")
                -
>setParameter("date_start", $from-
>format("Y-m.d")." 00:00:00")
                -
>setParameter("date_end", $to-
>format("Y-m.d")." 23:59:59")
                ->getQuery()
                ->getArrayResult();

            $result['subscribers'] =
$subscribers;
            $result['date'] = $from-
>format("d.m.y H:i:s")." - ".$to-
>format("d.m.y H:i:s");

            $href = $this-
>report($result);
            return new
JsonModel(['result'=>true,
'href'=>$href]);

    }
    $this->notFoundAction();
    }

    private function report($data) {
        $model = (new
\Manage\Subscribers\Report\Report())
->setReport($data);
        return $model;
    }


    public function
modalActivateAction()
    {
        $request = $this-
>getRequest();

        if ($request->isPost() &&
$request->isXmlHttpRequest()) {

            $id = $this->getEvent()
                ->getRouteMatch()
                ->getParam('id');

            $entityPages = $this-
>getEM()
                -
>find("Model\Entity\Subscribers",
$id);

            $form = (new
Form("/manage/subscribers/ajax-
activate-deactivate"))
                ->setData([
                    "id" =>
$entityPages->getId(),
                    "status" => 1,
            ]);

            return (new ViewModel())
                ->setTerminal(true)
                -
>setTemplate("manage/subscribers/mod
al/activate-deactivate.phtml")
                ->setVariables([
                    'item' =>
$entityPages,
                    'form' => $form,
                ]);
        }

        $this->notFoundAction();
    }
```

```
    public function
modalDeactivateAction()
    {
        $request = $this-
>getRequest();

        if ($request->isPost() &&
$request->isXmlHttpRequest()) {

            $id = $this->getEvent()
                ->getRouteMatch()
                ->getParam('id');

            $entityPages = $this-
>getEM()
                -
>find("Model\Entity\Subscribers",
$id);

            $form = (new
Form("/manage/subscribers/ajax-
activate-deactivate"))
                ->setData([
                    "id" =>
$entityPages->getId(),
                    "status" => 0,
            ]);

            return (new ViewModel())
                ->setTerminal(true)
                -
>setTemplate("manage/subscribers/mod
al/activate-deactivate.phtml")
                ->setVariables([
                    'item' =>
$entityPages,
                    'form' => $form,
                ]);
        }

        $this->notFoundAction();
    }

    public function
ajaxActivateDeactivateAction()
    {
        $request = $this-
>getRequest();

        if ($request->isPost()&&
$request->isXmlHttpRequest()) {
            $postParams = $request-
>getPost();
            /** @var Subscribers
$result */
            $result = $this->getEM()
                -
>getRepository("Model\Entity\Subscri
bers")
                -
>find($postParams['id']);

            if($result) {
                $result-
>setIsActive($postParams['status']);
                $this->getEM()-
>flush();
                return new
JsonModel(['result'=>true]);
            } else {
                return new
JsonModel(['result'=>false]);
            }
        }

        $this->notFoundAction();
    }

    public function
ajaxRemoveAction()
    {
        $request = $this-
>getRequest();

        if ($request->isPost()&&
$request->isXmlHttpRequest()) {
            $postParams = $request-
>getPost();
            $result = $this->getEM()
                -
>getRepository("Model\Entity\Subscri
bers")
                -
>find($postParams['id']);

            if($result) {
                $this->getEM()-
>remove($result);
                $this->getEM()-
>flush();
                return new
JsonModel(['result'=>true]);
            } else {
                return new
JsonModel(['result'=>false]);
            }
        }

       $this->notFoundAction();
    }

    public function ajaxGridAction()
    {
        $request = $this-
>getRequest();

        if ($request->isPost() &&
$request->isXmlHttpRequest()) {
            /** @var
\Manage\Pages\Grid\Pages */
            $managePagesGrid =
$this->grid;

            $response =
$managePagesGrid->getIndex($request-
>getPost());

            return new
JsonModel($response);
        }

        $this->notFoundAction();
    }

    private function getEM()
    {
```

```
            return $this->em;
        }
    }
```