**Зміст**

[Вступ 2](#_Toc32434602)

[1 Загальна характеристика підприємства практики 3](#_Toc32434603)

[2 Організація приміщення та робочого місця 5](#_Toc32434604)

[3 Техніка безпеки 8](#_Toc32434605)

[3.1 Вимоги техніки безпеки перед початком роботи 10](#_Toc32434606)

[3.2 Вимоги техніки безпеки під час роботи 10](#_Toc32434607)

[3.3 Вимоги техніки безпеки при завершенні роботи 11](#_Toc32434608)

[4 Інформаційна діяльність підприємства 12](#_Toc32434609)

[5 Інформаційна діяльність на робочому місці 15](#_Toc32434610)

[6 Звіт з індивідуального завдання 16](#_Toc32434611)

[6.1 Постановка задачі 16](#_Toc32434612)

[6.2 Виконання завдання 16](#_Toc32434613)

[Висновки 24](#_Toc32434614)

[Список використаних джерел 25](#_Toc32434615)

Вступ

Одним із головних завдань навчального закладу є підготовка молодих спеціалістів до майбутньої роботи. Технологічна практика дозволяє студенту навчитися пристосовуватися до праці на робочому місці. Дає початкові уявлення про структуру, матеріальну й технічну базу, а також технологічний процес підприємства. Також, робота на підприємстві дозволяє засвоїти на практиці знання з охорони праці, що є одним із найважливіших аспектів безпечної роботи для здоров’я працівників та навколишнього середовища.

Звичайно, окрім впровадження в технологічний процес підприємства, одним із головних завдань даної практики є практичне використання набутих знань, що виходять із змісту обраної спеціальності. Це проявляється у самому процесі діяльності на підприємстві, так і при виконанні індивідуального завдання.

Технологічна практика розпочинається з ознайомлення студентів з її метою, планом проходження, інструктажем з техніки безпеки, і згодом видається індивідуальне завдання. Практика проводиться на різноманітних підприємствах та в різних організаціях різних галузей господарювання, а саме таких як галузі освіти, охорони здоровя, праці, державного управління, які є оснащені відповідним чином. Також можливе проведення практики на базах навчальних закладів.

Основним завданням практики є оволодіння практикантами сучасних методів розробки ПЗ зокрема, та способами організації підприємницької діяльності в цілому.

Знання, набуті в результаті проходження технологічної практики є хорошим фундаментом для подальшої професійної діяльності. Набутий досвід дасть змогу в майбутньому швидше адаптовуватися до іншого виробничого середовища, комфортніше себе почувати на новому місці роботи, у новому робочому колективі, більш легко розбиратись із новими незнайомими технологіями та фреймворками.

1. Загальна характеристика підприємства практики
   1. Мета та рік створення

Підприємство займається наданням широкого спектру послуг з розробки програмного забезпечення, автоматизації та інформатизації різноманітних комп’ютерних систем. У ньому працює близько 100 співробітників різних кваліфікаційних напрямів, що дає змогу ефективно виконувати проекти різної складності в конкретно визначені строки. Офіс підприємства знаходиться в м. Хмельницькому.

MassMedia Group була заснована в 2007 році та розпочиналась як невелика веб-студія. На ранніх етапах було лише 2 працівника – Сергій та Олександр. Згодом до них приєднався ще Андрій Войтович, який до цього був їхнім замовником. Основною використовуваною мовою чого часу в компанії була мова PHP, яка й досі зараз переважає в відділі розробки компанії.

* 1. Форма власності

На законному рівні компанія є існує просто як ФОП Вержбицький С.В. Тобто, MassMedia Group – маркетингова назва компанії. Фізична особа підприємець (ФОП) – згідно з українським законодавством, фізична особа, яка реалізує свою здатність до праці шляхом самостійної, на власний ризик діяльності з метою отримання прибутку.

Окрім цього, MassMedia Group співпрацює зі своїми робітниками за контрактом так як із підприємцями. Тобто, кожен працівник організації повинен також бути оформленим як ФОП.

* 1. Основні види продукції, послуг чи видів діяльності

Зараз основною сферою роботи є розробка різних ступенів складності веб додатків переважно у сфері автоматизації бізнесу. Переважна більшість проектів – outstaff, тобто компанія є проміжною ланкою між розробниками та клієнтами. Для розробників складається чітке, зрозуміле їм технічне завдання, а для клієнтів – певні рекомендації по розвитку їх бізнесу. Тобто в даному ключі, MassMedia виступає також консалтинговою компанією.

В якості технологічних рішень, на поточний момент часу переважають:

* PHP (Laravel, Symfony);
* JavaScript (React, Angular, Vue);
* Node.js;
* Ruby;
* PostgreSQL;
* MySQL;
* Redis;
* Elasticsearch.
  1. Характеристика підрозділів підприємства

Організаційна структура підприємства складається з трьох співвласників, управлінського персоналу різних відділів та персоналу по кожному відділу. Схема структури підприємства зображена на рисунку 1.

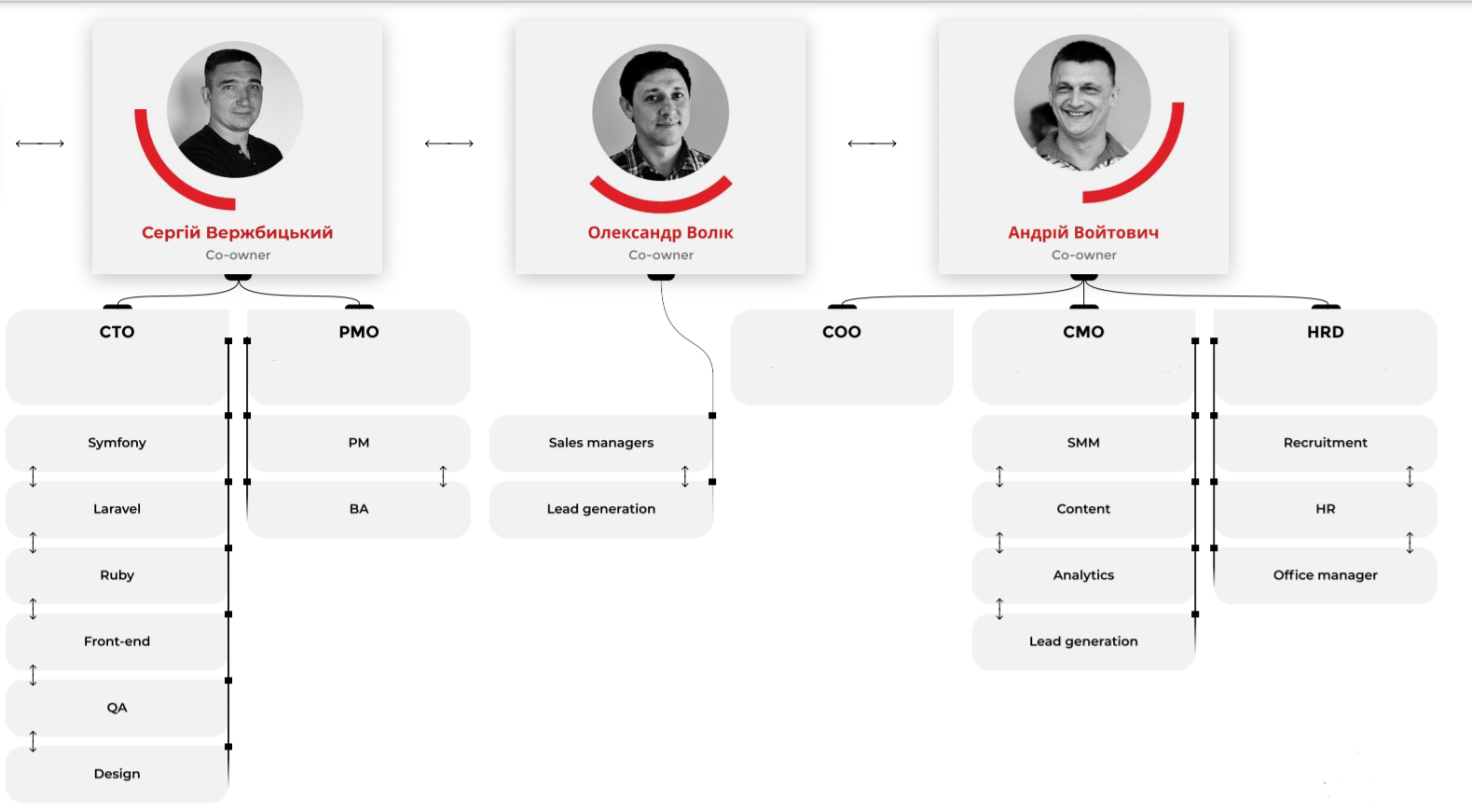


Рисунок 1 – Схема структури підприємства

Таким чином, підприємство складається з шести підрозділів, кожен з яких виконує свою роль в загальній справі розробки ПЗ. Більшість підрозділів розподіляється на відділи. Наприклад, CTO (chief technology officer), тобто технічний директор над виконавцями проектів, відповідає за набір ефективної команди згідно технічних навиків для виконання того чи іншого проекту, PMO (project management officer), тобто директор менеджерів проектів, відповідальний за організацію роботи відділу менеджменту, бізнес-аналітики та суміжних відділів. Такий розподіл зроблений для максимально ефективного виконання та менеджменту замовлених проектів.

* 1. Конкурентні переваги підприємства

Підприємство робить все можливе для розвитку дружньої атмосфери в своєму колективі, щоб співробітники ефективніше працювали в команді. На даний момент часу компанія налічує понад 70 працівників. З технічної сторони, кожному розробнику складається індивідуальний план розвитку, який кожен розробник виконує для покращення своїх знань. Також, компанія надає можливість своїм працівникам відвідувати курси англійської за кошти організації.

В даний момент підприємство активно розширюється, з’являються нові команди розробки та спеціалізації, спеціалісти які працюють в інших сферах та з іншим стеком технологій.

1. Організація приміщення та робочого місця

Організація робочого місця — це система заходів щодо його спеціалізації, оснащення необхідними засобами і предметами праці, їхнього розміщення на робочому місці, його зовнішнього оформлення і створення належних умов праці. Робоче місце – це частина простору, оснащена основним і допоміжним технологічним обладнанням, інвентарем, інструментом, робочими меблями, необхідними для виконання завдання в якій робітник здійснює трудову діяльність і проводить переважну частину робочого часу. Правильна організація робочого місця має вагомий вплив на ефективність праці на підприємстві.

Основними напрямами в організації робочих місць вважають:

* ефективне розміщення устаткування, оснащення, предметів праці;
* раціональну спеціалізацію;
* освітлення робочої площі;
* обслуговування;
* умови безпечної й високопродуктивної праці.

Під час роботи з персональним комп’ютером головну роль відіграє дотримання правильного режиму праці та відпочинку. При перенавантаженні та напруженій роботі виникає напруга зорового апарату, головні болі, порушення сну, втомленість, хворобливі відчуття в поясниці, областях шиї, руках та інше.

Відповідно до добового циклу працездатності найвищий її рівень припадає на ранкові і денні години: з 8 до 12 години першої половини дня, та з 14 до 17 години другої половини дня. У вечірні години працездатність знижується, досягаючи свого мінімуму вночі. Графік залежності продуктивності праці від годин роботи зображено на рисунку 2.

З метою попередження зниження працездатності, перерва на відпочинок призначається до настання втоми організму. На роботах, які вимагають великої напруги й уваги, рекомендуються часті, але короткі перерви по 5-10 хвилин.

На роботах, що протікають у сприятливих умовах праці, рекомендується активний відпочинок. Найбільш ефективною формою активного відпочинку є виробнича гімнастика.

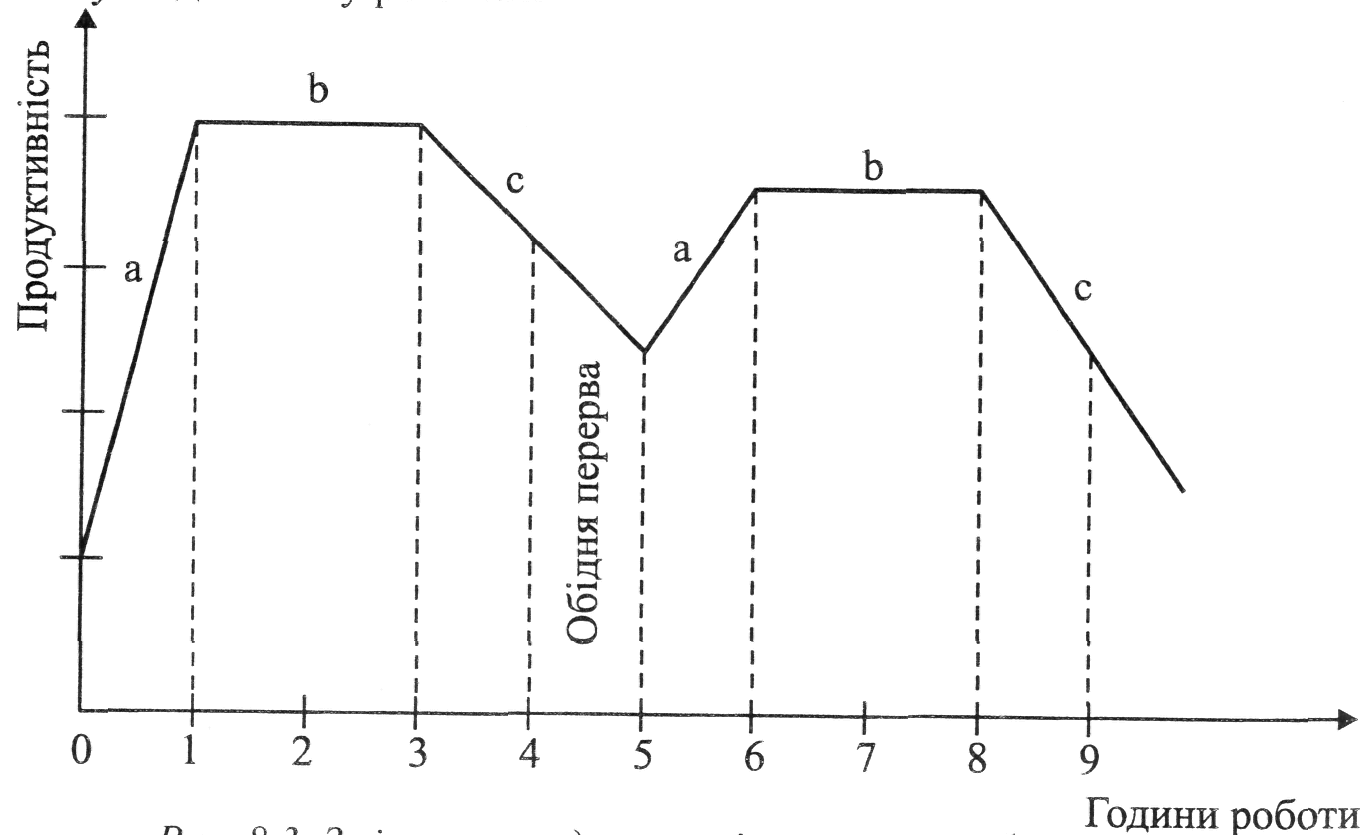


Рисунок 2 – Графік продуктивності праці

Зважаючи на положення тіла людини в сидячому положенні, можна сформулювати вимоги для оптимального робочого місця з точки зору збереження здоров'я. Повинна бути можливість зміни положення тіла, тобто вільне переміщення корпусу і кінцівок тіла відносно один одного. Висота столу повинна відповідати зросту працівника, тобто робоче місце повинно допускати можливість регулювання висоти для конкретного працівника. Зазвичай висота столу знаходиться у межах від 40 до 55 см.

Також немаловажливим є розміщення монітора. Він повинен знаходитись в такому положенні, щоб працівник зберігав правильну поставу. Найоптимальніша відстань – 50 сантиметрів. Монітор не повинен знаходитись занадто високо або низько. Повний кут огляду повинен бути приблизно 30 градусів вниз від горизонтальної прямої від очей. Правильне та неправильне оформлення робочого місця зображено на рисунку 3.

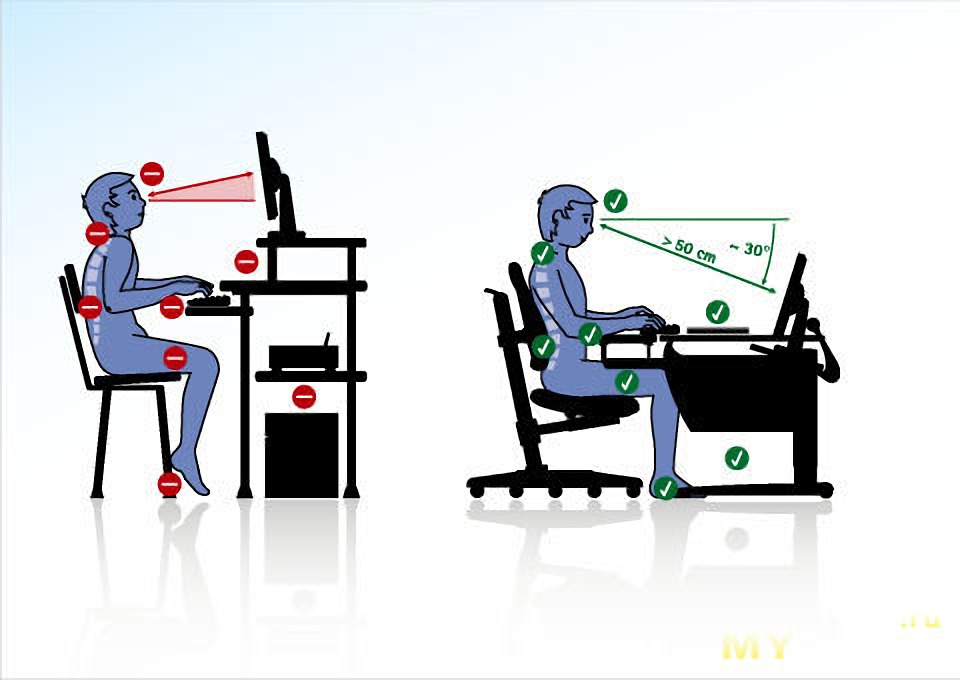


Рисунок 3 – Робоче місце

Для проходження практики було використано персональний комп’ютер з такою апаратною конфігурацією:

* оперативна пам'ять 8 Гб;
* процесор Intel Core i3 7th gen;
* відеокарта nVidia GeForce 940MX, 4 ГБ, інтегрована;
* дисплей 15.6" LED, матовий.

Встановлена операційна система та програмне забезпечення:

* операційна система Ubuntu Focal Fossa 20.04;
* середовище розробки PhpStorm;
* бази даних MySQL, PostgreSql;
* системи управління базами даних Dbeaver, Mysql Workbench;
* система контейнеризації Docker;
* браузери Google Chrome, Mozilla Firefox;
* месенджери Slack, Telegram, Skype.

1. Техніка безпеки

Приступаючи до роботи, необхідно пам’ятати, що це високовартісне обладнання, яке потребує обережного ставлення. Необхідно дотримуватись правил техніки безпеки, щоб запобігти виникненню небезпечних ситуацій, які можуть загрожувати життю або здоров’ю працівників.

Під час роботи за комп’ютером необхідно чітко дотримуватись правил безпеки та інструкцій з експлуатації апаратури.

Працювати на клавіатурі чистими сухими руками, не натискувати ті чи інші клавіші без потреби або навмання. Натискати клавіші необхідно не прикладаючи зайвої сили задля запобігання їх пошкоджень або більш швидкого зносу. Ці ж вимоги також стосуються й миші. При роботі з дисплеями, потрібно оберігати їх від ударів, дії магнітного поля або тепла.

У разі появи запаху горілого, незвичайних звуків або самовільного вмикання або вимикання пристроїв забороняється виконувати самовільний ремонт. Потрібно негайно вимкнути комп’ютер і повідомити про це керівника.

Основними правилами по техніці безпеки є:

* до роботи в приміщенні допускаються працівники з якими був проведений інструктаж по техніці безпеки;
* при будь-яких неполадках електронного обладнання необхідно відключити його від електромережі і повідомити про це керівнику відділу;
* забороняється вмикання і експлуатація машин при несправних блоках, які мають порушення цілісності корпусу або ізоляції провідників, а також з несправною індикацією включення живлення;
* при ввімкненому живленні мережі, обладнання забороняється включати і відключати роз’єми кабелів. Також при увімкненому живленні забороняється проводити ремонтні роботи, знімати щити і кришки обладнання;
* забороняється залишати ввімкнене обладнання без догляду.

Вимоги техніки безпеки перед початком роботи

Перед початком роботи потрібно:

* перевірити надійність встановлення апаратури на робочому столі;
* відрегулювати освітленість робочого місця;
* відрегулювати та зафіксувати висоту крісла, зручний для користувача нахил його спинки;
* перевірити загальний стан апаратури, перевірити справність електропроводки, з'єднувальних шнурів, штепсельних вилок, розеток;
* відрегулювати яскравість свічення монітора, мінімальний розмір світної точки, фокусування, контрастність. Не слід робити зображення надто яскравим, щоб не втомлювати очей;
* забороняється вмикати та використовувати комп’ютер та інші пристрої у випадку виявлення неполадок;

З метою уникнення перевантаження організму робочий день користувача персонального комп’ютера повинен проходити у раціональному режимі праці та відпочинку, який передбачає дотримання регламентованих перерв, їх активне проведення, систематичне проведення виробничої гімнастики, рівномірний розподіл зав­дань.

Загальний чає роботи з відеотерміналом не повинен перевищувати 50% тривалості робочого дня. Якщо виконання роботи пов'язане тільки з використанням комп'ютера, то при неможливості зміни діяльності необхідно робити перерви та паузи.

Для робіт, які виконуються з великим навантаженням, слід робити 10-15 хвилинну перерву через кожну годину, для малоінтенсивної роботи там перерви слід робити через дві години. Кількість мікропауз (до 1 хвилини) слід визначати індивідуально.

Вимоги техніки безпеки під час роботи

Після перевірки справності апаратури, потрібно увімкнути комп’ютер та розпочати роботу, із дотриманням інструкцій експлуатації.

Забороняється:

* замінювати і знімні елементи або вузли та проводити перемонтаж при ввімкненому комп'ютері;
* самостійно ремонтувати апаратуру, адже ремонт апаратури здійснюється спеціалістами з технічного обслуговування комп'ютера;
* з'єднувати і роз'єднувати вилки та розетки первинних мереж електроживлення, які знаходяться під напругою;
* ставити будь-які легкозаймисті предмети поряд із приладами, що можуть спричинити загоряння;
* знімати кришки, які закривають доступ до струмопровідних частин мережі первинного електроживлення при ввімкненому обладнанні;
* закривати будь-чим вентиляційні отвори апаратури, так як це може призвести до її перегрівання і виходу з ладу;
* залишати комп'ютер у ввімкненому стані без нагляду;
* ставити рідину на робочу техніку або біля неї.

Вимоги техніки безпеки при завершенні роботи

Вимоги безпеки після закінчення роботи:

* завершити та зберегти роботу на жорсткий диск комп'ютера або при використанні хмарного сховища, вивантажити зміни в хмару;
* прибрати робоче місце, розставивши використовувані пристрої на свої відповідні місця;
* вимкнути електроживлення комп’ютера згідно з інструкцією експлуатації;
* про виявлені недоліки у роботі ПК протягом робочого часу необхідно повідомити керівників.

З цього випливає, що належне виконання встановлених інструкцій з техніки безпеки, що направлені на створення безпечних і здорових умов праці, можливе лише за умови відповідності приміщень нормам та обізнаності всього персоналу про такі норми. Вкрай важливо, щоб власники та керівники підприємств всіх рівнів це розуміли і відповідально підходили до вирішення цього питання. Однак не менш важливим чинником є відповідальне ставлення працівників до правил техніки безпеки, так як легковаження цими принципами може призвести до серйозних наслідків.

Дотримання вимог безпечних умов праці, приведення робочих приміщень та місць працівників у відповідність до загальних вимог законодавства у сфері охорони праці є безпосереднім обов’язком як роботодавця, так і його підлеглих. Але слід пам’ятати, що навіть при дотриманні всіх норм та правил, все ж є вірогідність нещасного випадку, тому працівники повинні бути максимально пильними на своєму робочому місці.

1. Інформаційна діяльність підприємства

Технологічний процес – це сукупність послідовних робіт та операцій і методів та умов їх виконання, які забезпечують безперервність і ритмічність технології основного виробництва. Якщо звернутися до самого визначення терміну технологія, то можна дійти до висновку, що мета технології полягає в тому, щоб розкласти на складові елементи процес досягнення якого-небудь результату. Технологія застосовна всюди, де є досягнення, прагнення до результату, але усвідомлене використання технологічного підходу було справжньою революцією. До появи технології панувало мистецтво. Людина робила щось таке, що виходило лише в неї. З допомогою ж технології все те, що доступно тільки обраним, обдарованим, стає доступно всім.

Інформаційні технології включають в себе широкий клас дисциплін та галузей діяльності, що відносяться до технологій керування, накопичення, обробки і передачі інформації. Інформаційна технологія — процес, що використовує сукупність засобів і методів збору, накопичення, обробки та передачі даних, тобто первинної інформації, для отримання інформації нової якості про стан об'єкту, процесу або явища, тобто інформаційного продукту. Цей процес складається з чітко регламентованої послідовності виконання операцій, дій, етапів різного ступеня складності над даними, що зберігаються на комп'ютерах. Основною метою інформаційної технології є отримання необхідної для користувача інформації в результаті обробки та можливо певної модифікації первинної інформації.

Застосування відкритих інформаційних систем, розрахованих на використання всього масиву інформації, доступної в даний момент суспільству в певній його сфері, дозволяє удосконалити механізми управління суспільним устроєм, сприяє гуманізації і демократизації суспільства, підвищує рівень добробуту його членів. Процеси, що відбуваються у зв'язку з інформатизацією суспільства, сприяють не тільки прискоренню науково-технічного прогресу, інтелектуалізації всіх видів людської діяльності, а й створенню якісно нового інформаційного середовища соціуму, що забезпечує розвиток творчого потенціалу індивіда.

Під час проходження практики, backend-розробники відповідають за правильну та стабільну роботу серверної частини веб-додатків. Саме вони мають безперешкодний доступ до інформації в БД, що кладе на них відповідальність за безпеку збереження інформації. Розробники налаштовують сховища даних, автоматизують певні процеси по обробці інформації, встановлюють на налаштовують захист від несанкціонованого доступу.

З іншої сторони, frontend-розробники відповідають за візуальну частину веб-додатків, налаштовують клієнт-серверну взаємодію. Основна мета діяльності розробників цієї області — коректне відображення інформації на екранах кінцевих користувачів, зручність використання додатка.

Менджмент проеків організований за допомогою CircleCI — сучасна платформа безперервної інтеграції та безперервної доставки (CI / CD). Безперервна інтеграція — це практика, яка заохочує розробників часто інтегрувати свій код у головну гілку спільного сховища. Замість того, щоб розробити окремі функції та інтегрувати їх в кінці циклу розробки, код інтегрується у спільне сховище кожним розробником кілька разів протягом дня. Головним елементом circleci є робочий процес (англ. workflow) — певний набір кроків, описаних у спеціальному файлі конфігурації, що буде виконуватись кожного разу для збірки та автоматизованого тестування програмного продукта.

Під час написання коду, кожен розробник використовує інтегроване середовище розробки (IDE, Integrated Development Environment) — програмний додаток, що надає комплексні засоби для програмістів з метою пришвидшення та покращення розробки програмного забезпечення. Зазвичай IDE складається щонайменше з редактора вихідного коду, засобів автоматизації збірки та налагоджувача. Деякі середовища розробки, такі як Eclipse, містять необхідний компілятор, інтерпретатор, і навіть веб-сервер. На підприємстві в основному використовуються такі середовища розробки як PhpStorm та WebStorm.

Компанія MassMedia Group займається розробкою додатків для автоматизації бізнесу. Тому однієї з головних передумов розробки є вивчення, аналіз та уточнення технічного завдання, адже його від його розуміння залежить правильність та якість виконаної роботи.

Для управління більшості проектів застосовується Scrum-методологія Agile, схема якої зображена на рисунку 4.



Рисунок 4 – Agile

Agile — це ітераційний підхід до управління проектами та розробки програмного забезпечення, який допомагає командам швидше і з меншою кількістю проблем доставляти функціонал своїм клієнтам. Замість того, щоб робити запуск всього продукту в один момент часу, спритна команда додає частинки виконаної роботи в кожному спринті, а не відкладає все на останній день. Вимоги, плани та результати постійно оцінюються, тому команди мають природний механізм реагування на швидкі зміни.

1. Інформаційна діяльність на робочому місці

Під час проходження практики та виконання індивідуального завдання було вивчено та використано настпуні технології:

* мова програмування PHP;
* php-фреймворк Laravel;
* підсистема Laravel Nova;
* менеджер управління пакетами composer;
* система контейнеризації Docker;
* база даних MySQL;
* фреймвоки для тестування PHPUnit 9.

Розробка велась в інтегрованому середовищі розробки PHPStorm, яке було також налаштовано перед роботою. Також для редактора коду було встановлено плагіни для ефективнішої роботи: Laravel Plugin, Laravel IDEA, Symfony plugin, .env support та багато інших. Під час всього періоду практики неодноразово доводилось звертатись до документації PHP, документації з використання Docker, документації фреймворка Laravel та його підсистеми для адмін-панелі Laravel Nova.

1. Звіт з індивідуального завдання

Постановка задачі

Під час проходження технологічної практики індивідуальним завданням було доповнення функціоналу системи тестування студентів та зміна інтерфейсу адмін-панелі для більш зручного користування. Доповнення функціоналу повинно включати:

* вставка в тести елементів у вигляді зображень;
* зміна логіки виведення тестових питань з можливістю налаштування в адмін-панелі;
* можливість обмеження кількості спроб проходження теста, яку встановлює викладач;
* зручніший інтерфейс з можливістю гнучкої фільтрації записів.

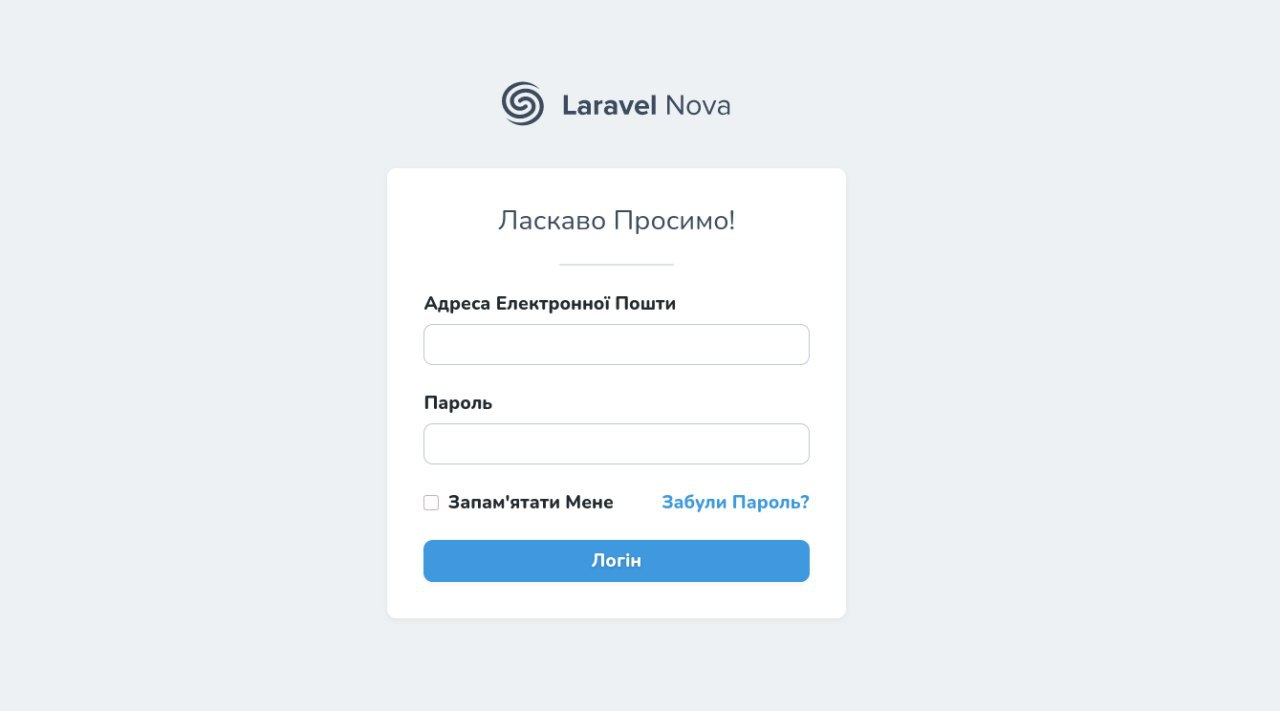
Технологіями виступають: мова програмування PHP, фреймворк Laravel, підсистема Laravel Nova, а також інструмент для локальної розробки та розгортання виступають Docker та Docker Compose.

Виконання завдання

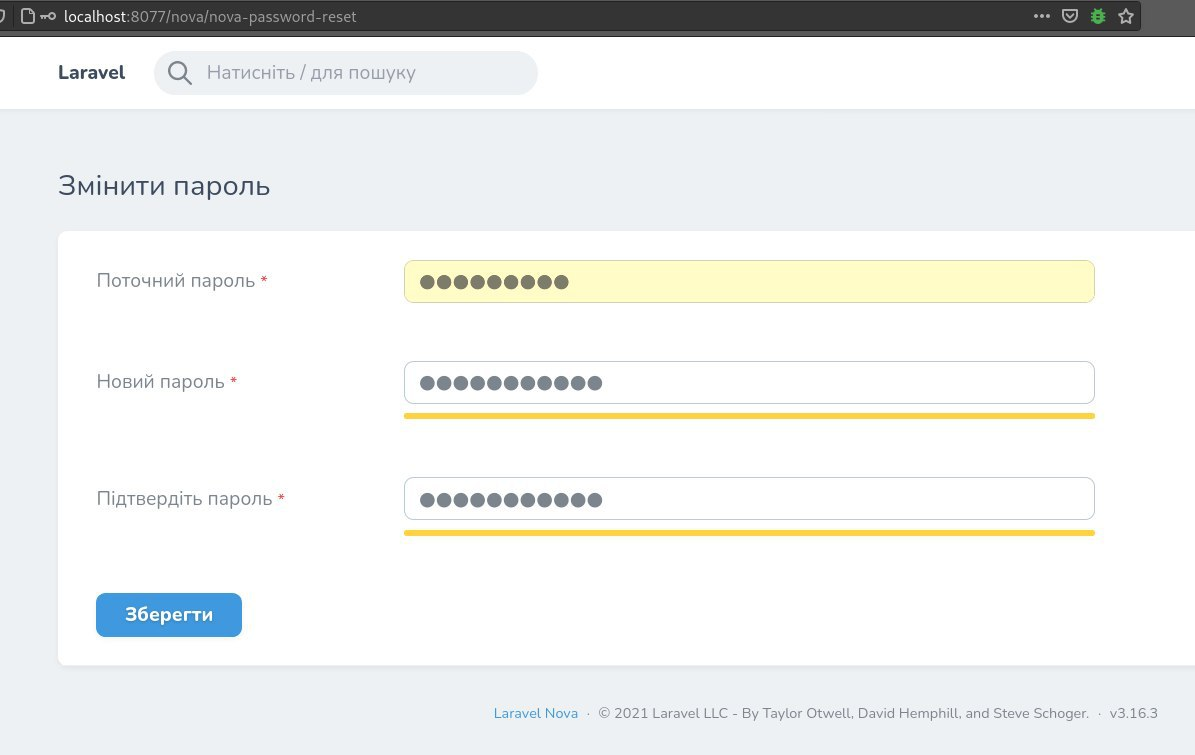
У цьому розділі описано можливі варіації роботи програми та показано результатати виконання програми.

Перш за все, було повністю перероблено адміністративну частину сайту з використанням Laravel Nova. При цьому, попередня версія адмін-панелі все ще є доступна для використання по роуті `/admin`. Оновлена ж версія включає в себе весь попередній функціонал та доповнює його новим. Знаходиться вона по роуті `/nova`. Особливість нової версії в тому, що вона мінімізує дублювання інформації. Наприклад, в попередньому варіанті для додавання питань для теста потрібно було обрати предмет, потім обрати тест, після чого додавати питання, те ж саме стосується перегляду результатів. Тепер же, ми можемо просто знайти тест ввівши частину його назви і одразу виконувати певні дії над ним, хоча попередній метод роботи також підтримується.

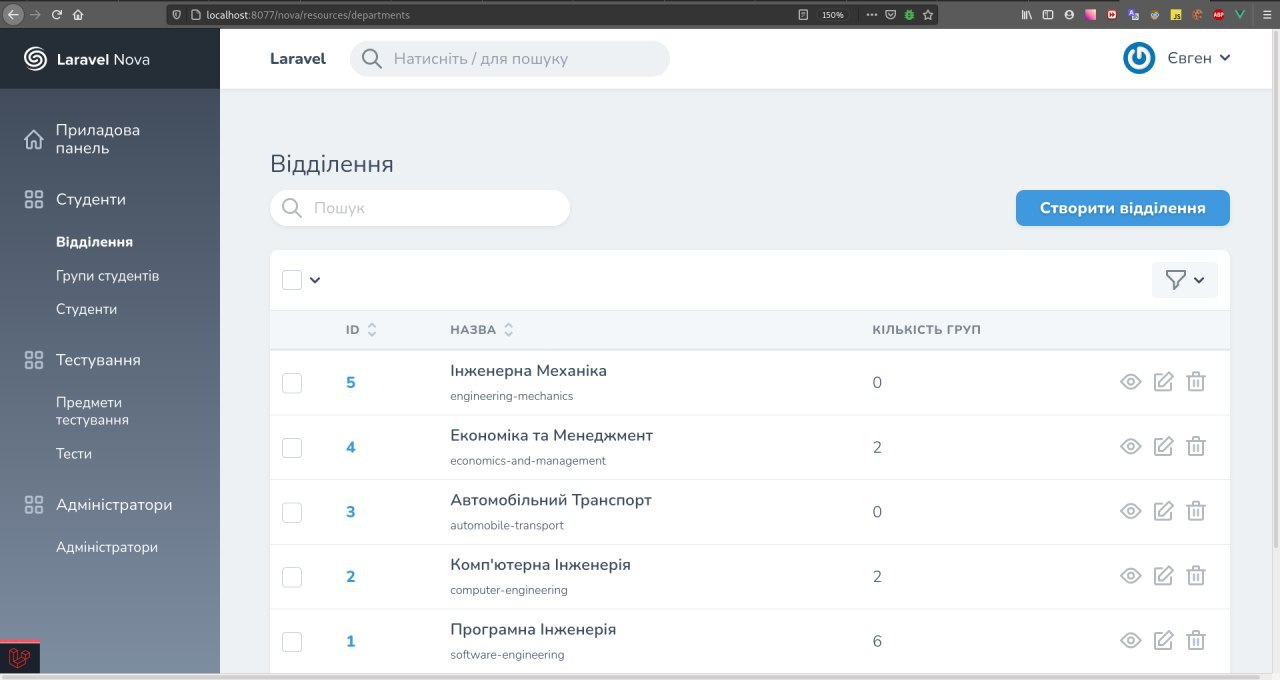
Коли адміністратор переходить на `/nova` він може аутентифікуватись.

  
Рисунок 5 – Сторінка входу в адмін-панель

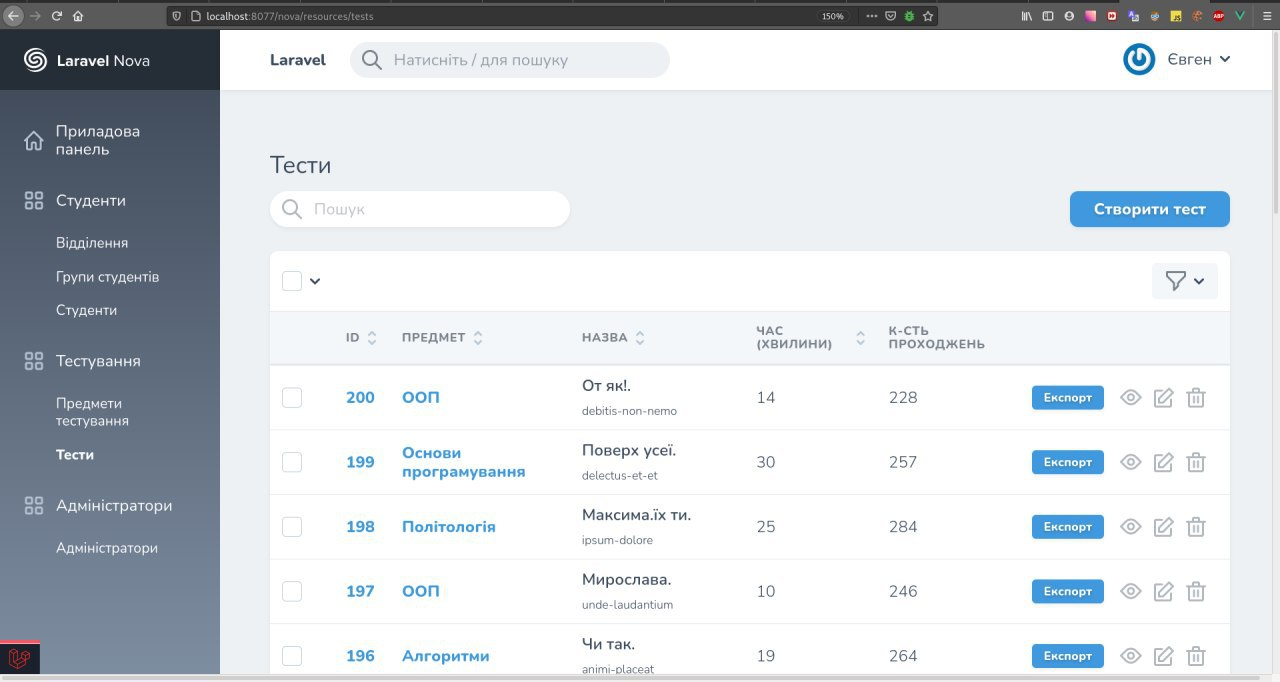
Згодом після першого входження в аккаунт, адміністратору пропонується змінити його пароль:

  
Рисунок 6 – Сторінка зміни паролю

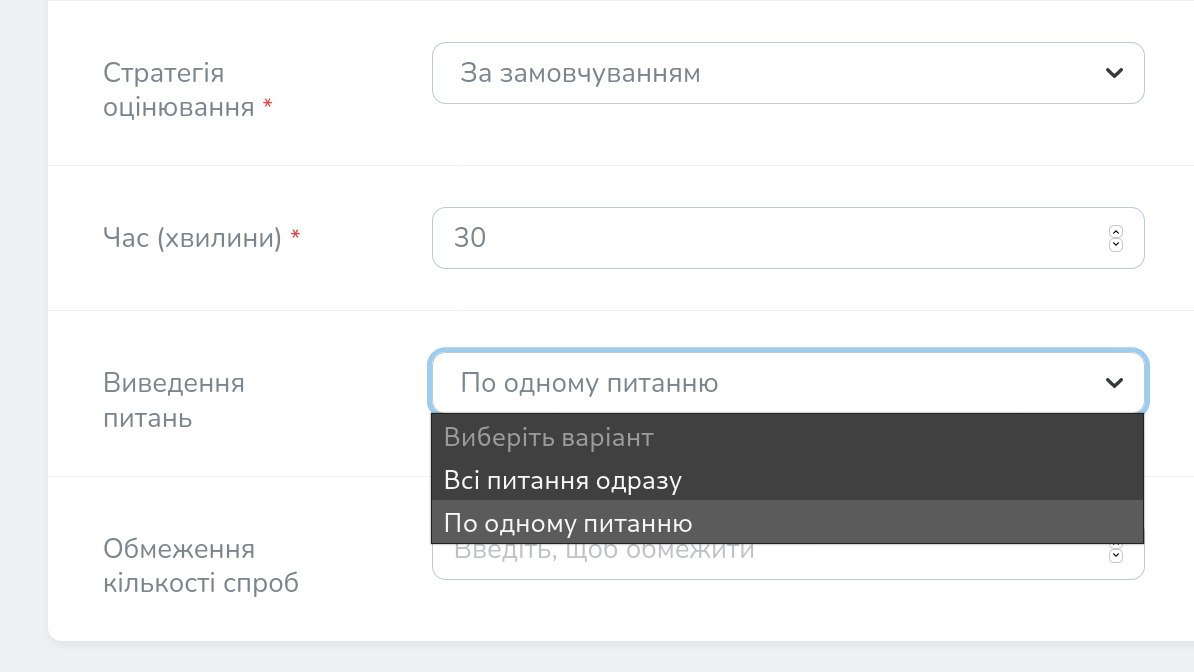
Тепер викладач може приступити до безпосередньої роботи з системою тестування.

  
Рисунок 7 – Новий вигляд сторінки відділень

Перейшовши на сторінку «Тести», викладач може переглянути список доступних йому тестів та певну додаткову інформацію як наприклад кількість проходжень – це зображено на рисунку 8. Важливою відмінністю є те, що тепер адміністратор може напряму переглянути всі його тести, а не заходити окремо в кожний предмет. Звідси ж можна робити експорт у файл.

  
Рисунок 8 – Перегляд тестів викладачем

На сторінці редагування тесту було додано новий варіант конфігурації «Виведення питань» як зображено на рисунку 9. За його допомогою можна налаштувати стратегію, згідно з якою студентам будуть відображатись питання під час проходження – всі одразу, чи по одному питанню на сторінку.

  
Рисунок 9 – Налаштуваня виведення питань

Також було додано параметр конфігурації для обмеження кількості спроб здачі тесту. Приклад налаштування зображено на рисунку 10. Додатково, в студента при виборі теста буде відображатись лічильник кількості спроб.

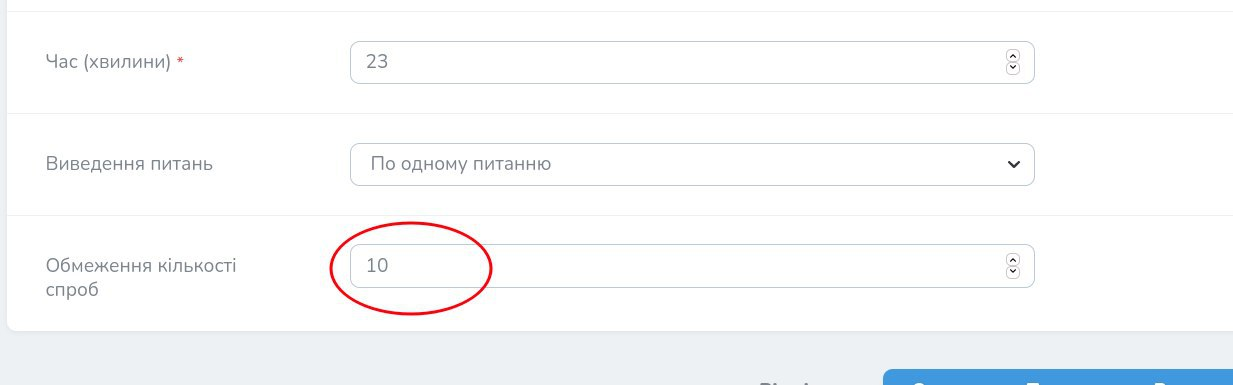


Рисунок 10 – Обмеження кількості проходжень

Коли студент обрав предмет для тестування і обирає тест, йому буде показано скільки спроб ще залишилось, це показано на рисуноку 11. Після чергового проходження тесту, цей лічильник зменшиться на одиницю, і так з кожним проходженням буде зменшуватись до тих пір, поки зовсім не зникне зі списку доступних для проходження тестів.

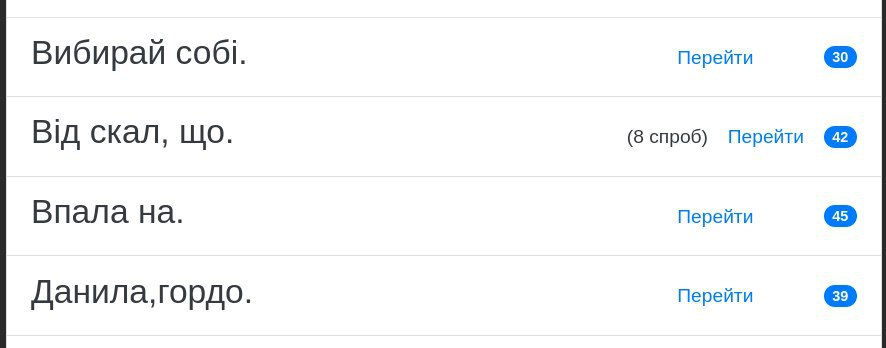
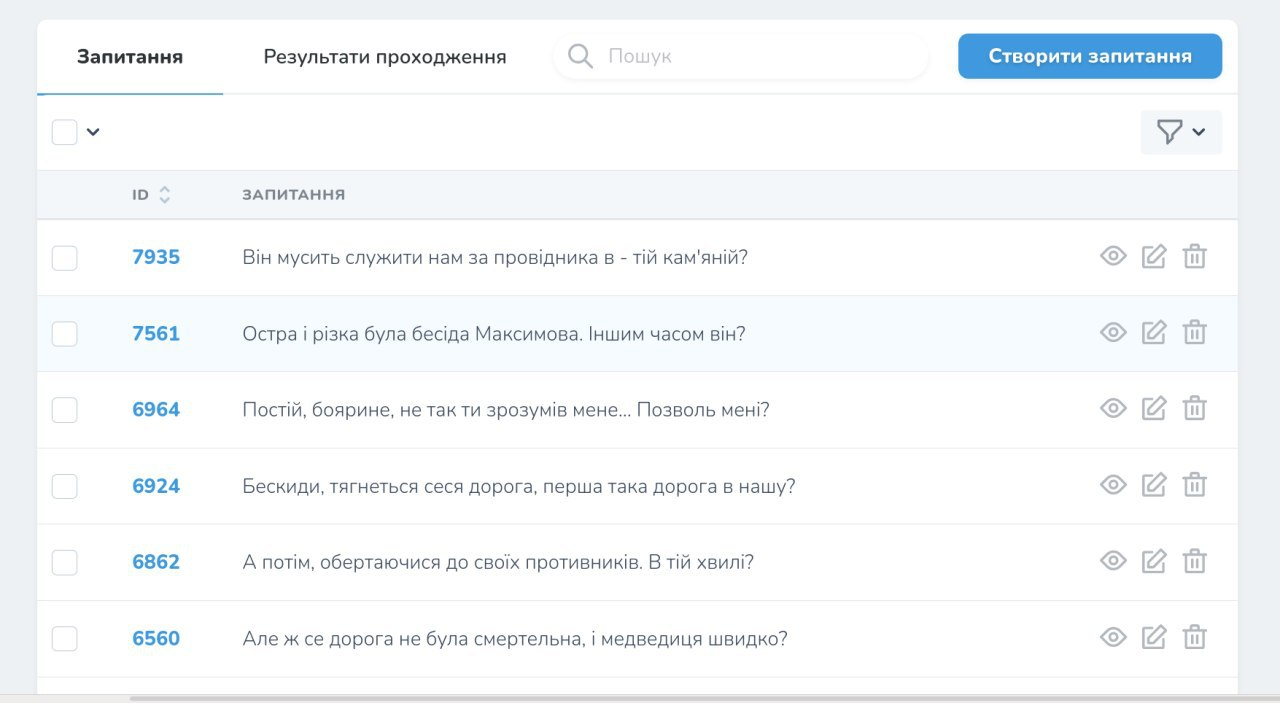


Рисунок 11 – Кількість невикористаних спроб

Іншою доволі вагомою зміною є спосіб роботи з питаннями, зображений на рисунку 12. Попередній спосіб був недосконалим там, що викладач може випадково залишити сторінку так і не зберігши всі нові питання, які він додав, адже редагування відбувалось для всіх питань одразу. Тепер же, можливо створювати та редагувати питання лише по одному за раз.

  
Рисунок 12 – Список усіх питань тесту

При створенні та редагуванні запитання викладач тепер має змогу форматувати текст запитання та додавати мультимедійні елементи такі як фото та відео.

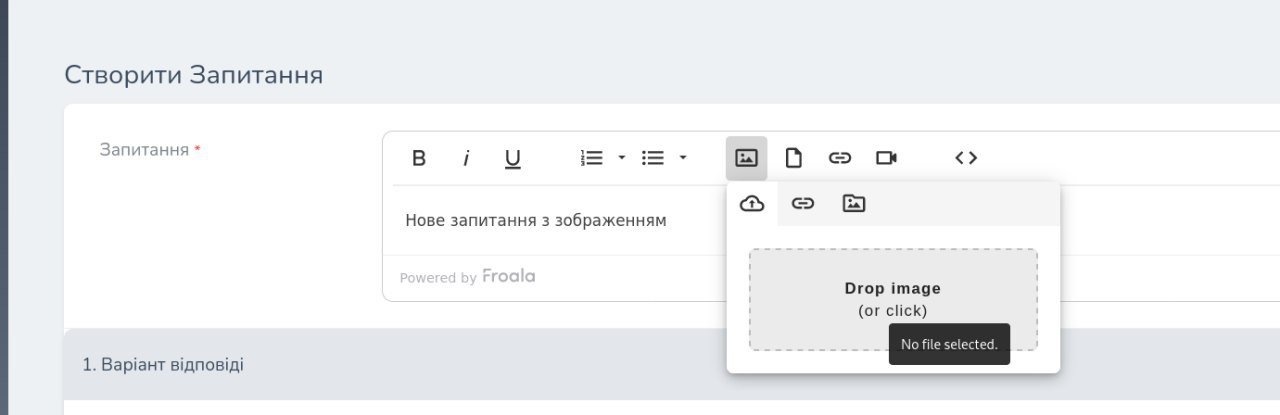


Рисунок 13 – Вставка зображення в текст

Хоча й редагування питань тепер на окремій сторінці, робота з варіантами відповідей залишена там же, де й робота із запитанням. Сторінка редагування запитання та варіантів відповідей зображена на рисунку 14. Це зроблено для правильного контролю інваріантів. Наприклад, кожне питання повинно мати хоча б один обраний як правильний варіант відповіді.

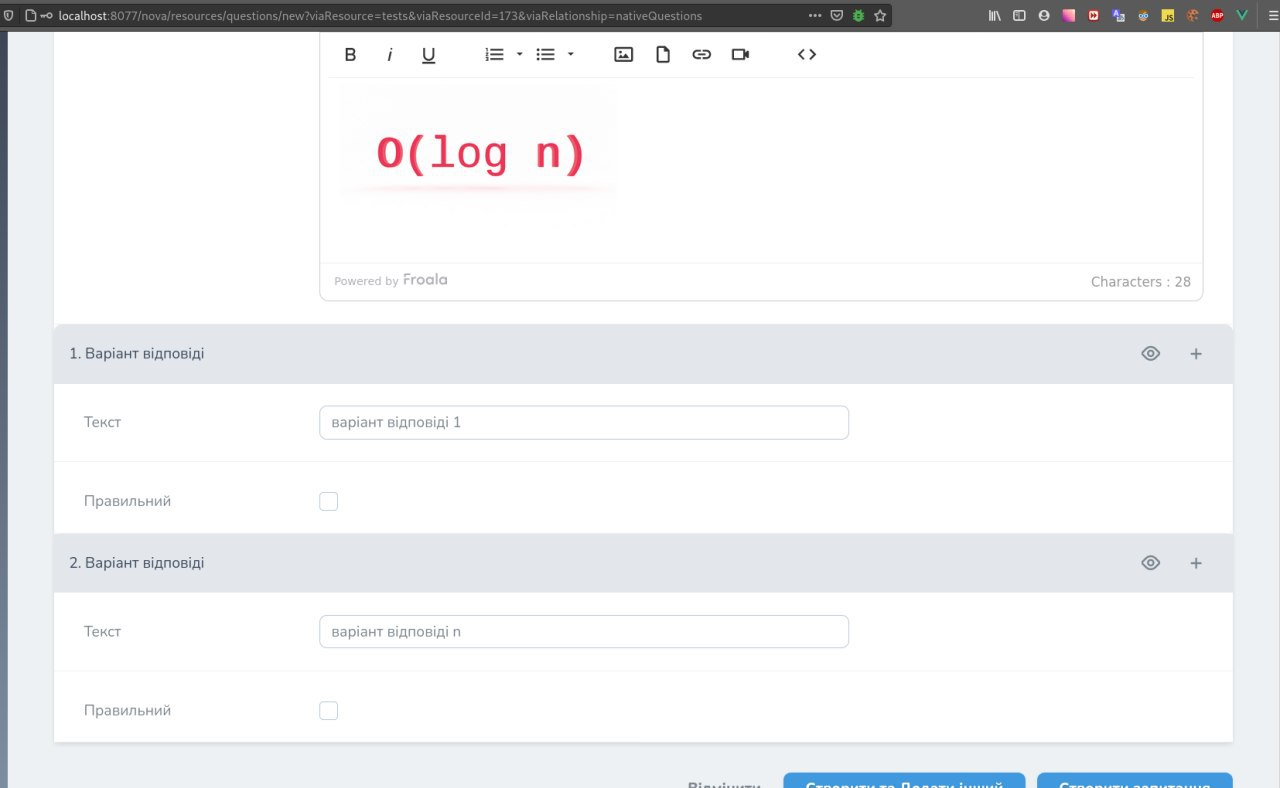
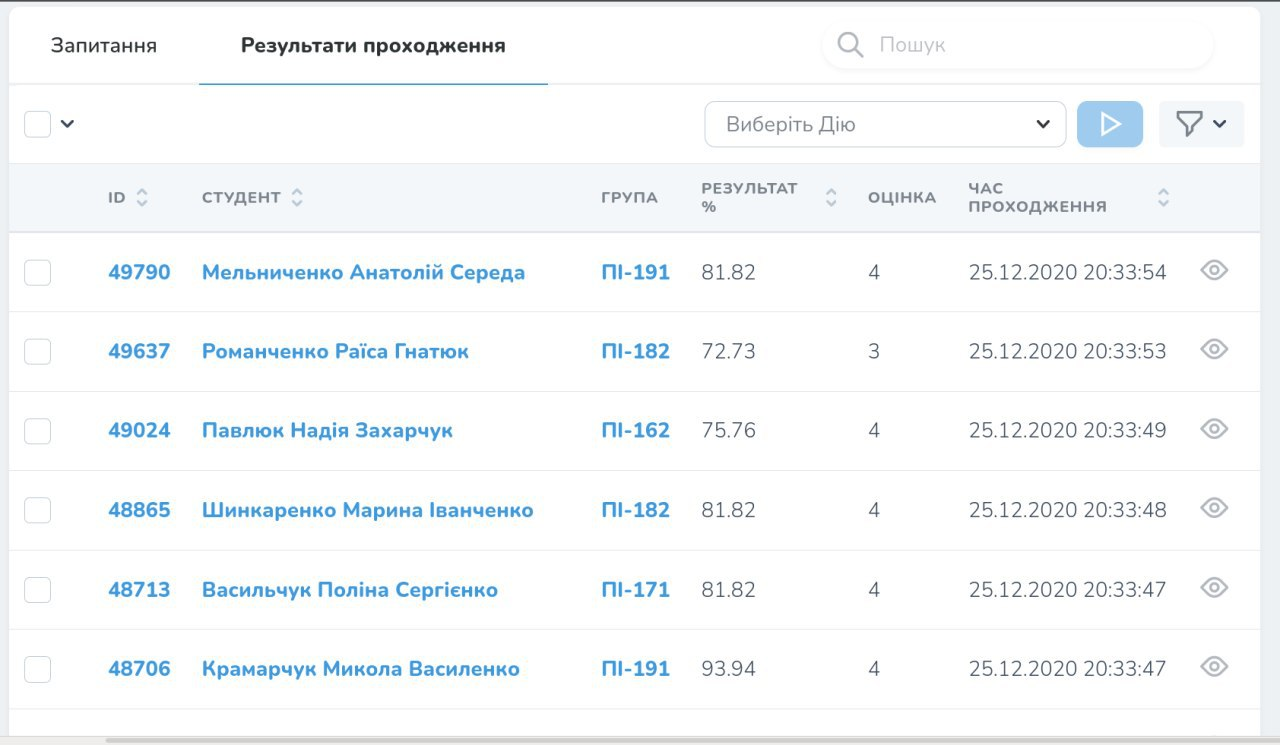
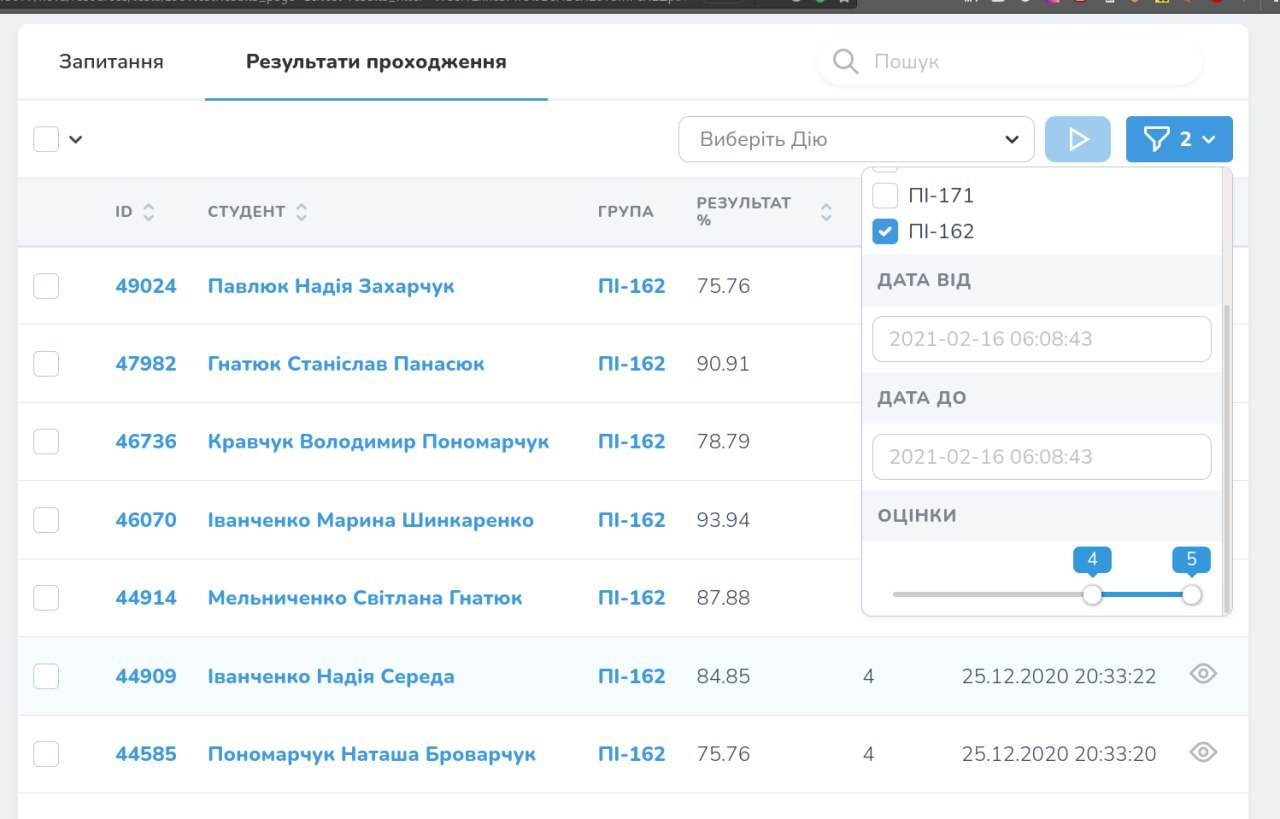


Рисунок 14 – Редагування варіантів відповідей питання

Перегляд результатів тестування зроблено на тій же сторінці, що й загальна інформація про тест, однак на іншій від запитань вкладці. Це зображено на рисунку 15. За замовчуванням, результати проходження виводяться в порядку спадання по даті створення, тобто спочатку нові. Однак, можна посортувата за власним бажанням по іншому полю. Наприклад, посортувавши по відсотках результату, можна побачити найкращі результати студентів.

  
Рисунок 15 – Перегляд результатів проходження

Також, на цій сторінці передбачено фільтри результатів по даті проходження, групі, а також оцінкам. Зображено на рисунку 16. При фільтрації по групі, можна буде згенерувати відповідний звіт згідно фільтрів. Основною функціональною відмінністю від варіанту попередньої версії тестової системи є те, що тепер викладач може обмежити результати відомості записами певного проміжку часу. Тобто, наприклад, за конкретний день, або за конкретну годину у випадку якщо тест здавався на парі. З іншої, технічної сторони, змінився принцип обрахунків результатів. До цього, з бази даних вибирались всі проходження конкретного теста і для них програмно обраховувались результати. Тепер же, всі обрахунки результатів покладені на базу даних, і додаток вибирає лише ті записи, які йому потрібні зараз, а не всі по тесту. Це дає значне покращення швидкості.

  
Рисунок 16 – Фільтрація результатів проходження

Клас app/Nova/Test.php:

**<?php  
  
namespace** App\Nova;  
  
**use** App\Models\Tests\TestEloquentBuilder;  
**use** App\Nova\Actions\AttachTestsQuestionsToTest;  
**use** App\Nova\Actions\ExportTestIntoFile;  
**use** App\Nova\Actions\ImportTestFromFile;  
**use** App\Nova\Fields\Custom\Test\AdditionalQuestionsRelationField;  
**use** App\Nova\Fields\Custom\Test\AttemptsPerUserField;  
**use** App\Nova\Fields\Custom\Test\GradingStrategyField;  
**use** App\Nova\Fields\Custom\Test\GradingTableField;  
**use** App\Nova\Fields\Custom\Test\NameField;  
**use** App\Nova\Fields\Custom\Test\NameStackReadOnly;  
**use** App\Nova\Fields\Custom\Test\PassTimeField;  
**use** App\Nova\Fields\Custom\Test\ResultsCountReadOnly;  
**use** App\Nova\Fields\Custom\Test\TestDisplayStrategyField;  
**use** App\Nova\Fields\Custom\Test\TestSubjectField;  
**use** App\Nova\Fields\Custom\Test\TestTypeField;  
**use** App\Nova\Fields\Custom\Test\UriAliasField;  
**use** Eminiarts\Tabs\Tabs;  
**use** Epartment\NovaDependencyContainer\NovaDependencyContainer;  
**use** Illuminate\Database\Eloquent\Builder;  
**use** Illuminate\Http\Request;  
**use** Laravel\Nova\Fields\HasMany;  
**use** Laravel\Nova\Fields\ID;  
**use** Laravel\Nova\Http\Requests\NovaRequest;  
  
**class** Test **extends** Resource  
{  
 **public static** *$group* = **'Тестування'**;  
  
 **public static int** $groupPriority = 2;  
  
 **public static** *$model* = \App\Models\Test::**class**;  
  
 */\*\** ***@var*** *\App\Models\Test \*/* **public $resource**;  
  
 **public static** *$title* = **'name'**;  
  
 **public static** *$preventFormAbandonment* = **true**;  
  
 **public static** *$perPageViaRelationship* = 25;  
  
 **public static** *$search* = [  
 **'id'**,  
 **'name'**,  
 ];  
  
 */\*\*  
 \** ***@param*** *NovaRequest $request  
 \** ***@param*** *TestEloquentBuilder $query  
 \** ***@return*** *Builder  
 \*/* **public static function** indexQuery(NovaRequest $request, $query)  
 {  
 **return** $query->withCount(**'testResults'**)  
 ->with(**'subject'**)  
 ->with(**'marksPercents'**)  
 ->with(**'nativeQuestions.answerOptions'**)  
 ->availableForAdmin($request->user());  
 }  
  
 */\*\*  
 \** ***@param*** *NovaRequest $request  
 \** ***@param*** *TestEloquentBuilder $query  
 \** ***@return*** *Builder  
 \*/* **public static function** detailQuery(NovaRequest $request, $query): Builder  
 {  
 **return** $query->withCount(**'testResults'**)  
 ->with(**'subject'**);  
 }  
  
 **public function** fields(Request $request)  
 {  
 **return** [  
 ID::*make*()->sortable(),  
  
 TestSubjectField::*make*(),  
  
 NameStackReadOnly::*make*(),  
  
 NameField::*make*(),  
  
 UriAliasField::*make*(),  
  
 TestTypeField::*make*(),  
  
 GradingStrategyField::*make*(),  
  
 AdditionalQuestionsRelationField::*make*(),  
  
 NovaDependencyContainer::*make*([GradingTableField::*make*()])  
 ->dependsOn(**'mark\_evaluator\_type'**, **'custom'**),  
  
 PassTimeField::*make*(),  
  
 TestDisplayStrategyField::*make*(),  
  
 AttemptsPerUserField::*make*(),  
  
 ResultsCountReadOnly::*make*(),  
  
 **new** Tabs(  
 **'Relationships'**,  
 [  
 HasMany::*make*(**'Запитання'**, **'nativeQuestions'**, Question::**class**),  
  
 HasMany::*make*(**'Результати проходження'**, **'testResults'**, TestResult::**class**)  
 ->singularLabel(**'проходження'**),  
 ]  
 ),  
  
 ];  
 }  
  
 **public static function** label()  
 {  
 **return 'Тести'**;  
 }  
  
 **public static function** singularLabel()  
 {  
 **return 'Тест'**;  
 }  
  
 **public function** cards(Request $request): **array** {  
 **return** [];  
 }  
  
 **public function** filters(Request $request): **array** {  
 **return** [  
 **new** Filters\SubjectsFilter(**'test\_subject\_id'**),  
 ];  
 }  
  
 **public function** lenses(Request $request): **array** {  
 **return** [  
 ];  
 }  
  
 **public function** actions(Request $request): **array** {  
 **return** [  
 (**new** AttachTestsQuestionsToTest())->onlyOnIndex(),  
 (**new** ExportTestIntoFile())->onlyOnTableRow()->showOnDetail(),  
 (**new** ImportTestFromFile())->onlyOnDetail(),  
 ];  
 }  
}

}

Класи полів тесту:

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** App\Rules\Containers\Test\TestUriSlugRules;  
**use** Laravel\Nova\Fields\Slug;  
  
**final class** UriAliasField  
{  
 **public static function** make()  
 {  
 **return** Slug::*make*(**'Uri-псевдонім'**, **'uri\_alias'**)  
 ->from(**'name'**)  
 ->creationRules(TestUriSlugRules::*forCreate*())  
 ->updateRules(TestUriSlugRules::*forUpdate*())  
 ->hideFromDetail()  
 ->hideFromIndex();  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** App\Models\Test;  
**use** App\Rules\Containers\Test\TestTypeRules;  
**use** Laravel\Nova\Fields\Select;  
  
**final class** TestTypeField  
{  
 **public static function** make()  
 {  
 **return** Select::*make*(**'Тип теста'**, **'type'**)  
 ->hideFromIndex()  
 ->displayUsingLabels()  
 ->options(  
 *array\_combine*(  
 Test::***TYPES***,  
 *array\_map*(  
 **static fn**($t) => \_\_($t),  
 Test::***TYPES*** )  
 )  
 )->default(Test::***TYPE\_STANDALONE***)  
 ->rules(**new** TestTypeRules());  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** App\Nova\TestSubject;  
**use** Laravel\Nova\Fields\BelongsTo;  
  
**final class** TestSubjectField  
{  
 **public static function** make()  
 {  
 **return** BelongsTo::*make*(**'Предмет'**, **'subject'**, TestSubject::**class**)  
 ->sortable();  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** App\Models\Test;  
**use** App\Rules\Containers\Test\TestDisplayStrategyRules;  
**use** App\Rules\Containers\Test\TestTypeRules;  
**use** Laravel\Nova\Fields\Select;  
  
**final class** TestDisplayStrategyField  
{  
 **public static function** make()  
 {  
 **return** Select::*make*(**'Виведення питань'**, **'display\_strategy'**)  
 ->hideFromIndex()  
 ->displayUsingLabels()  
 ->options(  
 *array\_combine*(  
 Test::***DISPLAY\_STRATEGIES***,  
 *array\_map*(  
 **static fn**($s) => \_\_(**'tests.strategies.'**.$s),  
 Test::***DISPLAY\_STRATEGIES*** )  
 )  
 )->default(Test::***DISPLAY\_ONE\_BY\_ONE***)  
 ->rules(**new** TestDisplayStrategyRules());  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** Laravel\Nova\Fields\Number;  
  
**final class** ResultsCountReadOnly  
{  
 **public static function** make()  
 {  
 **return** Number::*make*(**'К-сть проходжень'**, **'test\_results\_count'**)  
 ->exceptOnForms()  
 ->readonly();  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** App\Rules\Containers\Test\TestPassTimeRules;  
**use** Laravel\Nova\Fields\Number;  
  
**final class** PassTimeField  
{  
 **public static function** make()  
 {  
 **return** Number::*make*(**'Час (хвилини)'**, **'time'**)  
 ->placeholder(**''**)  
 ->rules(**new** TestPassTimeRules())  
 ->sortable();  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** Laravel\Nova\Fields\Line;  
**use** Laravel\Nova\Fields\Stack;  
  
**final class** NameStackReadOnly  
{  
 **public static function** make()  
 {  
 **return** Stack::*make*(  
 **'Назва'**,  
 **'name'**,  
 [  
 Line::*make*(**'Name'**, **'name'**)->asHeading(),  
 Line::*make*(**'Slug'**, **'uri\_alias'**)->asSmall(),  
 ]  
 )->sortable();  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** App\Rules\Containers\Test\TestNameRules;  
**use** Laravel\Nova\Fields\Text;  
  
**final class** NameField  
{  
 **public static function** make()  
 {  
 **return** Text::*make*(**'Назва'**, **'name'**)  
 ->rules(**new** TestNameRules())  
 ->hideFromDetail()  
 ->hideFromIndex();  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** App\Models\MarkPercent;  
**use** App\Models\Test;  
**use** App\Rules\Containers\Test\TestGradingTableRules;  
**use** App\Services\Tests\Grading\GradingTableService;  
**use** Fourstacks\NovaRepeatableFields\Repeater;  
**use** Illuminate\Database\Eloquent\Collection;  
  
**final class** GradingTableField  
{  
 **public static function** make()  
 {  
 **return** Repeater::*make*(**'Таблиця оцінювання'**, **'correlation\_table'**)  
 ->rules((**new** TestGradingTableRules())->usingJson())  
 ->initialRows(1)  
 ->addField(  
 [  
 **'label'** => **'Оцінка'**,  
 **'name'** => **'mark'**,  
 **'type'** => **'number'**,  
 **'attributes'** => [  
 **'required'** => **'required'**,  
 ],  
 ]  
 )  
 ->addField(  
 [  
 **'label'** => **'Відсоток'**,  
 **'name'** => **'percent'**,  
 **'type'** => **'number'**,  
 **'attributes'** => [  
 **'step'** => 0.01,  
 **'required'** => **'required'**,  
 ],  
 ]  
 )->resolveUsing(  
 **function** ($value, Test $resource, $attribute) {  
 $marksPercents = $resource->**marksPercents** ?? collect();  
  
 **return** $marksPercents->toJson();  
 }  
 )->fillUsing(  
 **function** ($request, $model, $attribute, $requestAttribute) {  
 $table = collect(  
 optional(  
 $request->{$requestAttribute},  
 **static fn**($json) => *json\_decode*($json, **true**)  
 ) ?? []  
 );  
  
 $table = Collection::*make*(  
 $table->map(  
 **function** ($attrs) {  
 **return new** MarkPercent($attrs);  
 }  
 )  
 );  
  
 resolve(GradingTableService::**class**)->attachForTest($model, $table);  
 }  
 );  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** App\Rules\Containers\Test\TestMarkEvaluatorTypeRules;  
**use** Laravel\Nova\Fields\Select;  
  
**final class** GradingStrategyField  
{  
 **public static function** make()  
 {  
 **return** Select::*make*(**'Стратегія оцінювання'**, **'mark\_evaluator\_type'**)  
 ->displayUsingLabels()  
 ->options(  
 *array\_combine*(  
 \App\Models\Test::***EVALUATOR\_TYPES***,  
 *array\_map*(  
 **static fn**($t) => \_\_($t),  
 \App\Models\Test::***EVALUATOR\_LABELS*** )  
 )  
 )->default(\App\Models\Test::***EVALUATOR\_TYPE\_DEFAULT***)  
 ->rules(**new** TestMarkEvaluatorTypeRules())  
 ->hideFromIndex();  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** Laravel\Nova\Fields\Number;  
  
**final class** AttemptsPerUserField  
{  
 **public static function** make()  
 {  
 **return** Number::*make*(**'Обмеження кількості спроб'**, **'attempts\_per\_user'**)  
 ->placeholder(**'Введіть, щоб обмежити'**)  
 ->hideFromIndex()  
 ->min(1)  
 ->max(1000)  
 ->nullable();  
 }  
}

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Nova\Fields\Custom\Test;  
  
**use** App\Nova\Test;  
**use** Laravel\Nova\Fields\BelongsToMany;  
**use** Laravel\Nova\Fields\Number;  
**use** Laravel\Nova\Http\Requests\ResourceDetailRequest;  
  
**final class** AdditionalQuestionsRelationField  
{  
 **public static function** make()  
 {  
 **return** BelongsToMany::*make*(**'Тести для додаткових запитань'**, **'tests'**, Test::**class**)  
 ->singularLabel(**'Тест для додаткових запитань'**)  
 ->fields(  
 **fn**() => [  
 Number::*make*(**'К-сть Запитань'**, **'questions\_quantity'**)  
 ->placeholder(**''**)  
 ->min(1)  
 ->max(999)  
 ]  
 )->showOnDetail(  
 **function** (ResourceDetailRequest $request, \App\Models\Test $resource) {  
 **return** $resource->**type** === \App\Models\Test::***TYPE\_COMPOSED***;  
 }  
 )->searchable();  
 }  
}

Компонент app/Nova/TestResult.php:

**<?php  
  
declare**(strict\_types=1);  
  
**namespace** App\Nova;  
  
**use** App\Models\StudentGroup **as** StudentGroupModel;  
**use** App\Models\StudentGroups\StudentGroupEloquentBuilder;  
**use** App\Models\Students\StudentEloquentBuilder;  
**use** App\Models\TestResult **as** TestResultModel;  
**use** App\Models\TestResults\TestResultQueryBuilder;  
**use** App\Nova\Actions\DownloadTestResultReport;  
**use** App\Nova\Actions\DownloadTestResultsGroupReport;  
**use** App\Nova\Filters\FromTimestampFilter;  
**use** App\Nova\Filters\StudentGroupsFilter;  
**use** App\Nova\Filters\TestResultMarksFilter;  
**use** App\Nova\Filters\ToTimestampFilter;  
**use** Illuminate\Database\Eloquent\Builder;  
**use** Illuminate\Database\Eloquent\Relations\Relation;  
**use** Illuminate\Http\Request;  
**use** Laravel\Nova\Fields\BelongsTo;  
**use** Laravel\Nova\Fields\DateTime;  
**use** Laravel\Nova\Fields\HasMany;  
**use** Laravel\Nova\Fields\ID;  
**use** Laravel\Nova\Fields\Number;  
**use** Laravel\Nova\Http\Requests\NovaRequest;  
  
**final class** TestResult **extends** Resource  
{  
 **public static** *$model* = *TestResultModel*::**class**;  
 **public static** *$perPageViaRelationship* = 40;  
  
 **public static** *$title* = **'id'**;  
  
 **public static** *$search* = [  
 **'id'** ];  
  
 */\*\** ***@var*** *TestResultModel \*/* **public $resource**;  
  
 */\*\*  
 \** ***@param*** *NovaRequest $request  
 \** ***@param*** *TestResultQueryBuilder $query  
 \*  
 \** ***@return*** *Builder  
 \*/* **public static function** indexQuery(NovaRequest $request, $query)  
 {  
 **return** $query->with(  
 [  
 **'test'**,  
 **'user.studentGroup'**,  
 ]  
 )->withResultPercents();  
 }  
  
 */\*\*  
 \** ***@param*** *NovaRequest $request  
 \** ***@param*** *TestResultQueryBuilder $query  
 \*  
 \** ***@return*** *Builder  
 \*/* **public static function** detailQuery(NovaRequest $request, $query)  
 {  
 **return** $query->with(  
 [  
 **'test'**,  
 **'user.studentGroup'**,  
 ]  
 )->withResultPercents();  
 }  
  
 **public function** fields(Request $request)  
 {  
 **return** [  
 ID::*make*()->sortable(),  
  
 BelongsTo::*make*(**'Тест'**, **'test'**, Test::**class**),  
  
 BelongsTo::*make*(**'Студент'**, **'user'**, Student::**class**)  
 ->sortable(),  
  
 Fields\BelongsTo::*make*(**'Група'**, **'user.studentGroup'**, StudentGroup::**class**)  
 ->hideWhenCreating()  
 ->hideWhenUpdating(),  
  
 Number::*make*(**'Результат %'**, **'result\_percents'**)  
 ->sortable(),  
  
 Number::*make*(**'Оцінка'**, **'result\_mark'**),  
  
 DateTime::*make*(**'Час проходження'**, **'created\_at'**)  
 ->sortable()  
 ->format(**'DD.MM.YYYY HH:mm:ss'**),  
  
 HasMany::*make*(**'Задані питання'**, **'askedQuestions'**, AskedQuestion::**class**),  
 ];  
 }  
  
 **public static function** label()  
 {  
 **return 'Результати проходження'**;  
 }  
  
 **public static function** singularLabel()  
 {  
 **return 'Проходження'**;  
 }  
  
 **public static function** authorizedToCreate(Request $request)  
 {  
 **return false**;  
 }  
  
 **public function** authorizedToUpdate(Request $request)  
 {  
 **return false**;  
 }  
  
 **public function** authorizedToDelete(Request $request)  
 {  
 **return false**;  
 }  
  
 **public function** cards(Request $request)  
 {  
 **return** [];  
 }  
  
 **public function** filters(Request $request): **array** {  
 **return** [  
 **new** StudentGroupsFilter(  
 **function** ($query, **array** $groupIds) {  
 */\*\** ***@var*** *Builder|Relation|TestResultModel $query \*/* **return** $query->whereHas(  
 **'user'**,  
 **fn**($userQuery) => $userQuery->whereIn(**'student\_group\_id'**, $groupIds)  
 );  
 },  
 *StudentGroupModel*::**query**()  
 ->when(  
 **'tests'** === $request->get(**'viaResource'**),  
 **fn**(StudentGroupEloquentBuilder $query) => $query->whereHas(  
 **'students'**,  
 **function** ($studentQuery) **use** ($request) {  
 */\*\** ***@var*** *StudentEloquentBuilder $studentQuery \*/* **return** $studentQuery->whereHas(  
 **'testResults'**,  
 **function** ($testResultsQuery) **use** ($request) {  
 */\*\** ***@var*** *TestResultQueryBuilder $testResultsQuery \*/* $testResultsQuery->where(**'test\_id'**, $request->get(**'viaResourceId'**));  
 },  
 );  
 },  
 ),  
 ),  
 ),  
 **new** FromTimestampFilter(**'created\_at'**, **'Дата від'**),  
 **new** ToTimestampFilter(**'created\_at'**, **'Дата до'**),  
 ...$this->additionalFilters($request),  
 ];  
 }  
  
 **public function** lenses(Request $request)  
 {  
 **return** [];  
 }  
  
 **public function** actions(Request $request)  
 {  
 **return** [  
 (**new** DownloadTestResultReport())->canRun(**static fn**() => **true**),  
 (**new** DownloadTestResultsGroupReport())->canRun(**static fn**() => **true**),  
 ];  
 }  
  
 **private function** additionalFilters(Request $request): **array** {  
 $additional = [];  
  
 **if** ($request->get(**'viaResource'**) === **'tests'**) {  
 $additional[] = **new** TestResultMarksFilter((**int**)$request->get(**'viaResourceId'**));  
 }  
  
 **return** $additional;  
 }  
}

Білдер запитів app/Models/TestResults/TestResultQueryBuilder.php:

**<?php  
  
  
namespace** App\Models\TestResults;  
  
  
**use** App\Lib\Filters\Eloquent\ResultFilter;  
**use** App\Lib\Traits\FilteredScope;  
**use** App\Models\Test;  
**use** App\Models\TestResult;  
**use** Illuminate\Database\Eloquent\Builder;  
**use** Illuminate\Database\Eloquent\Collection;  
**use** Illuminate\Database\Query\JoinClause;  
  
*/\*\** ***@mixin*** *TestResult \*/***class** TestResultQueryBuilder **extends** Builder  
{  
 **use** FilteredScope {  
 scopeFiltered **as private**;  
 }  
  
 **public function** ofTest($test): **self** {  
 $testId = *is\_numeric*($test) ? $test : $test->**id**;  
  
 **return** $this->whereHas(  
 **'test'**,  
 **static function** (Builder $query) **use** ($testId) {  
 */\*\*  
 \** ***@var*** *Builder|Test $query  
 \*/* $query->**withTrashed**();  
 $query->where(**'id'**, $testId);  
 }  
 );  
 }  
  
 **public function** recent($count): **self** {  
 **return** $this->latest()->limit($count);  
 }  
  
 */\*\*  
 \** ***@param*** *ResultFilter $filters  
 \** ***@param*** *callable|null $callback  
 \*  
 \** ***@return*** *Collection|TestResult[]  
 \*/* **public function** filtered(ResultFilter $filters, **callable** $callback = **null**): Collection  
 {  
 **return** $this->scopeFiltered($this, $filters, $callback);  
 }  
  
 **public function** withResultPercents(): **self** {  
 **return** $this->appendSelect(  
 $this->raw(**'test\_result\_in\_percents(test\_results.id) as result\_percents'**)  
 );  
 }  
  
 **public function** whereMarkPercentBetween(**float** $min, **float** $max): **self** {  
 **return** $this->whereRaw(**"test\_result\_in\_percents(test\_results.id) between** $min **and** $max**"**);  
 }  
  
 **public function** whereMarkPercentAtLeast(**float** $min): **self** {  
 **return** $this->whereRaw(**"test\_result\_in\_percents(test\_results.id) >=** $min**"**);  
 }  
  
 **public function** whereMarkPercentAtMost(**float** $max): **self** {  
 **return** $this->whereRaw(**"test\_result\_in\_percents(test\_results.id) <=** $max**"**);  
 }  
  
 **public function** lastResultsByEachUser(): **self** {  
 $innerQuery = (**clone** $this)  
 ->reorder()  
 ->select(**'user\_id'**)  
 ->selectRaw(**'max(created\_at) as last\_passage\_at'**)  
 ->groupBy(**'user\_id'**);  
 **return** $this  
 ->joinSub(  
 $innerQuery,  
 **'recent\_passage'**,  
 **fn**(JoinClause $join) => $join->on(**'test\_results.user\_id'**, **'='**, **'recent\_passage.user\_id'**)  
 ->on(**'test\_results.created\_at'**, **'='**, **'recent\_passage.last\_passage\_at'**)  
 );  
 }  
}

Білдер запитів app/Models/Tests/TestEloquentBuilder.php:

**<?php  
  
declare**(strict\_types=1);  
  
  
**namespace** App\Models\Tests;  
  
**use** App\Models\Administrator;  
**use** App\Models\Query\CustomEloquentBuilder;  
**use** App\Models\Subjects\SubjectEloquentBuilder;  
**use** App\Models\Test;  
**use** App\Models\TestResult;  
**use** App\Models\User;  
  
*/\*\** ***@mixin*** *Test \*/***final class** TestEloquentBuilder **extends** CustomEloquentBuilder  
{  
 **public function** withUserResultsCount(User $user): **self** {  
 **return** $this->withCount(  
 [  
 **'testResults as user\_results\_count'** => **function** ($q) **use** ($user) {  
 */\*\** ***@var*** *TestResult $q \*/* $q->where(**'user\_id'**, $user->**id**);  
 }  
 ]  
 );  
 }  
  
 **public function** availableForAdmin(Administrator $administrator): **self** {  
 **if** ($administrator->can(**'viewAll'**, Test::**class**)) {  
 **return** $this;  
 }  
  
 **return** $this->whereHas(  
 **'subject'**,  
 **static fn**(SubjectEloquentBuilder $builder) => $builder->availableForAdmin($administrator)  
 )->where(**'created\_by'**, $administrator->**id**);  
 }  
}

Всі зовнішні параметри або змінні винесені і зберігаються в файлі .env:

APP\_NAME=Laravel  
APP\_ENV=local  
APP\_KEY=base64:7IcGYThuY+5fS4bmKQPL6enOC9nAWMSURvJL1XyPI54=  
APP\_DEBUG=true  
APP\_URL=http://localhost:8077  
  
LOG\_CHANNEL=daily  
  
DB\_CONNECTION=mysql  
DB\_HOST=db  
DB\_PORT=3306  
DB\_DATABASE=quizzy\_db  
DB\_USERNAME=quizzy  
DB\_PASSWORD=p@$$w0rd  
  
BROADCAST\_DRIVER=log  
CACHE\_DRIVER=file  
QUEUE\_CONNECTION=sync  
SESSION\_DRIVER=file  
SESSION\_LIFETIME=120  
  
REDIS\_HOST=127.0.0.1  
REDIS\_PASSWORD=null  
REDIS\_PORT=6379  
  
MAIL\_DRIVER=smtp  
MAIL\_HOST=smtp.mailtrap.io  
MAIL\_PORT=2525  
MAIL\_USERNAME=null  
MAIL\_PASSWORD=null  
MAIL\_ENCRYPTION=null  
MAIL\_FROM\_ADDRESS=null  
MAIL\_FROM\_NAME="${APP\_NAME}"  
  
AWS\_ACCESS\_KEY\_ID=  
AWS\_SECRET\_ACCESS\_KEY=  
AWS\_DEFAULT\_REGION=us-east-1  
AWS\_BUCKET=  
  
PUSHER\_APP\_ID=  
PUSHER\_APP\_KEY=  
PUSHER\_APP\_SECRET=  
PUSHER\_APP\_CLUSTER=mt1  
  
MIX\_PUSHER\_APP\_KEY="${PUSHER\_APP\_KEY}"  
MIX\_PUSHER\_APP\_CLUSTER="${PUSHER\_APP\_CLUSTER}"

Висновки

Технологічна практика корисна для студентів тим, що кожен студент може закріпити і проявити на практиці в зовсім новій та незнайомій ситуації свої знання. Практика виконання реальних завдань є найважливішою частиною навчального процесу для розробників програмного забезпечення. Завдяки їй, студенти навчаються використовувати набір вивченої інформації для вирішення реальних завдань.

Практика дає основу для подальшої професійної діяльності, так як вона передбачає своєю основною метою закріплення отриманих в процесі навчання знань, набуття практичних навичок вирішення проблем. Завдяки їй студенти отримують початковий досвід роботи в сфері інформаційних технологій, що неодмінно веде до підвищення рівня професійної підготовки.

Під час проходження практики було проведено ознайомлення з режимом роботи підприємства, використовуваними технологіями, способом взаємодії між працівниками, основними виконуваними завданнями та загальним внеском результатів виконання завдань на проект зокрема та на компанію в цілому. На практиці була змога спробувати себе в якості розробника програмного забезпечення, архітектора інфраструктури коду та бази даних для нового проекту. Закріпив свої знання в розробці серверної частини веб-додатків, освоїв основні принципи архітектури декомпонованих систем, дізнався деякі тонкощі роботи з різними системами баз даних, їх сильними сторонами та певними обмеженнями.

Отриманий у результаті практики досвід, дає змогу кожному студенту в майбутньому краще адаптуватися до певного виробничого середовища, легко влитись в новий колектив, зменшити стрес від зміни обстановки й налагодити свій робочий розпорядок для отримання найефективніших результатів. В результаті чого добре себе зарекомендувати як хорошого спеціаліста і отримати більше шансів для отримання вакантного місця.

Список використаних джерел

* 1. Роберт Мартин. Чистый код. создание, анализ и рефакторинг
  2. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения
  3. https://te.dsp.gov.ua/vplyv-pratsi-ta-vidpochynku-na-organizm-lyudyny/ – стаття про вплив режиму праці та відпочинку на організм людини
  4. <https://www.php.net/docs.php/>- документація php
  5. https://laravel.com/docs/ - документація фреймворку Laravel
  6. <https://refactoring.guru/> - сайт зі схемами для рефакторингу
  7. <https://proglib.io/> - сайт з корисною інформацією по розробці
  8. <https://circleci.com/docs/2.0/> - документація circleci