МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Сервіс проходження опитування підсистеми "Кабінет СумДУ"»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Проценко О.Б.**

Студента групи ІН – 71 **Ваценко А. В.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-71 спеціальності " Комп'ютерні науки" денної форми навчання Ваценка Андрія Володимировича.

Тема: "Сервіс проходження опитування підсистеми "Кабінет СумДУ""

Затверджена наказом по СумДУ

№_____ від _____2021 р.

Зміст пояснювальної записки: 1) огляд технологій, що застосовуються для проходження опитувань 2) постановка завдання й формування завдань дослідження; 3) огляд технологій, що використовуються під час розробки кліент-серверних додатків; 4)моделювання системи проходження опитування; 5) розробка сервісу опитування; 6) аналіз результатів.

Дата видачі завдання "_____"_____2021р.

Керівник випускної роботи _____ Проценко О.Б.

Завдання прийняв до виконання _____ Ваценко А. В.

РЕФЕРАТ

Записка: 69 стор., 20 рис., 1 додаток, 15 джерел.

Об’єкт дослідження — клієнт-серверна програма

Мета роботи — створення клієнт-серверного додатку для проходження опитування студентам СумДУ

Методи дослідження — системно-інформаційний аналіз, інформаційне моделювання та комп'ютерний експеримент

Результати — проведено вивчення і аналіз літератури, методів та інструментів, за допомогою яких можливо створити клієнтський додаток та серверну частину для збереження данних. За результатами проведеного дослідження вирішено створити сервіс, що складається з SPA додатку та серверу REST API. Дана розробка дозволяє авторизованим користувачам проходити опитування, отримувати список активних опитувань. Для користувачів с розширеними правами створення опитувань, питань. Сервіс створений с застосуванням мови PHP на базі фрейворку Laravel, та React для клієнтської частини.

СЕРВІС ПРОХОДЖЕННЯ ОПИТУВАННЯ ПІДСИСТЕМИ
"КАБІНЕТ СУМДУ"
A SURVEY INFORMATION SYSTEM OF SUBSYTEM "CABINET SSU"

ЗМІСТ

ВСТУП

Робота присвячена розробці інформаційної системи для проходження опитувань студентів СумДУ, що дозволить індетифікувати студента системи та заборонити не санкціонований доступ до проходження опитувань.

Такий додаток є ефективними засобом для зручного проходження опитування на теми актутальні студентам та співробітникам СумДУ, що дозволить проводити опитуваня для великої аудиторії на ранньому етапі. Також для користування сервісом потрібно бути студентом або співробітником університету, що обмежує кількість користувачів.

Сама ідея проходження опитувань не нова. Такі рішення можна побачити в проектах “Google Forms” та подібних. На жаль, такі сервіси не дають змогу авторизувати користувача, тому в даній роботі буде розглянуто створення сервісу проходження опитування для студентів та робітників СумДУ з використанням кліент-серверних технологій та фреймворків Laravel, React.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Технології взаємодії серверу та клієнта

Існує доволі багато способів до написання клієнт-серверної програми. Одними з найпопулярніших протоколів для взаємодії є ftp, ssh, http. Протоколи дозволяють отримувати данні та виконувати команди на віддаленому сервері.

Протокол доступу ssh – використовується для доступу до віддалених ресурсів. Цей протокол доречно використовувати для виконання команд на сервері і він не може бути використаний для написання програми, що використовує клієнт-серверну комунікацію.

Протокол ftp – використовується для доступу до файлів на сервері та відмінно може скачувати файлі такі як відео та рисунки. Основною відмінністю протоколу є запит файлу за повним ім’ям. Таким чином цей протокол не може бути використаний при написанні програми, але він знаходить своє призначення при роботі з файлами на віддалених серверах та роботі з ними [1-4].

Широке розповсюдження в світі веб програмування набув протокол http та став незамінним в створенні будь-яких кліент-серверних програм. Цей протокол належить до сімейства stateless протоколів. Протокол передає данні від сервера до клієнта та навпаки. Протокол може використовуватися як транспортний рівень для інших протоколів Soap, xml-rpc та інші.

Характерними рисами протоколу http є:

- Асиметричність, тобто одна із сторін є ініціатором, для комунікації потрібно щоб клієнт з’єднався з сервером та почав комунікацію з ним;
- Комунікація відбувається за допомогою запитів. В цей спосіб дані передаю між клієнтом та сервером.
- Відсутність збереженнь стану між запитами.

- Доступ до даних відбувається за допомогою URI, які є спецільними вказівниками для доступу до ресурсів.
- Розгалуджена система методів та специфікація, яка явно описує призначення кожного методу(GET, POST та інші).

Оскільки для передачі даних протоколом HTTP не було можливості дізнатися про структуру повідомлення, а передача текстового рядка не була кращим варіантом, були створені формати JSON та XML. Клієнт отримує дані від серверу в одному із форматів та перетворює їх в об’єкти на веб-сторінці [5].

JSON — є текстовим форматом обміну даними між комп’ютерами в мережі. В основі JSON лежить текст, в цей спосіб полегшується його сприйняття людиною. Формат дозволяє описувати об'єкти та масиви, рядки, числа та інше. Тому широко застосовується для передачі інформації з заданою структурою. Може піддаватися серіалізації та стисненню під час передачі по мережі.

Завдяки лаконічності формат JSON здобув більшу популярність порівняно з XML та є кращим для серіалізації комплексних структур.

JSON складається з таких полів:

- Поля ключ та значення. Більшість мов програмування уже мають готові структури такі як хеш-таблиця, рядок, словник , список з ключем.
- Лист значень. Прикладами реалізації є масиви та послідовності.

Завдяки універсальності в структурі даних JSON здобув більшу популярність порівняно з іншими технологіями передачі даних. Також формат дозволяє передавати інформацію між двома мовами програмування так як структура його універсальна і є велика кількість бібліотек що можуть зчитати дані в цьому форматі.

Рисунок 1.1 - Структура json об'єкту

У JSON складається з:

Об'єктів — це набір пар ключ значення які виділяються { } та включають потрібну інформацію перелічену через кому.

Масивів — є списком даних однакові за типом, що об’єднані [та закінчуються знаком]. Також можуть бути вкладені інші об’єкти чи рядки.

Рядків — що представляють собою набір символів в послідовності unciode[13].

Протокол HTTP працює поверх протоколів транспортного рівня TCP і UDP. Протокол встановлює TCP сесію на кожній запит, але в більш пізніх версіях протоколу можливе отримання більш ніж 1-го файлу в сесії. Можливе отримання файлів відео та аудіо за допомогою UDP з HTTP. HTTP підтримує зміну протоколу на інший, прикладом може бути підвищення(upgarde) HTTP на Websocket protocol.

Для підтримання авторизації користувачів можливе використання заголовків. Існує два поширені способи авторизації користувачів за допомогою HTTP: авторизація за кукою та авторизація зо токеном. За домогою куки можливо авторизувати користувача в запиті так, як в заголовоку при відправці запита буде надіслана кука. В такий же спосіб відбувається авторизація по токену. [3].

1.2 Сервіси для проходження опитувань(Огляд наявних рішень)

Одним з найпопулярніших сервісів для опитувань є “Google forms”. Цей сервіс дозволяє створювати і проводити опитування для будь якої кількості користувачів.

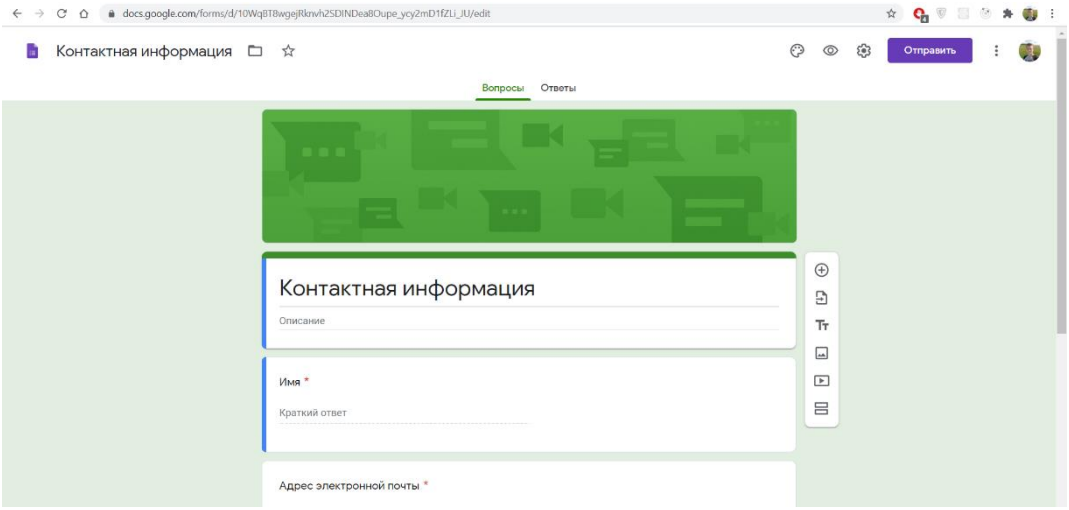

Рисунок 1.2 - Інтерфейс “Google Forms”

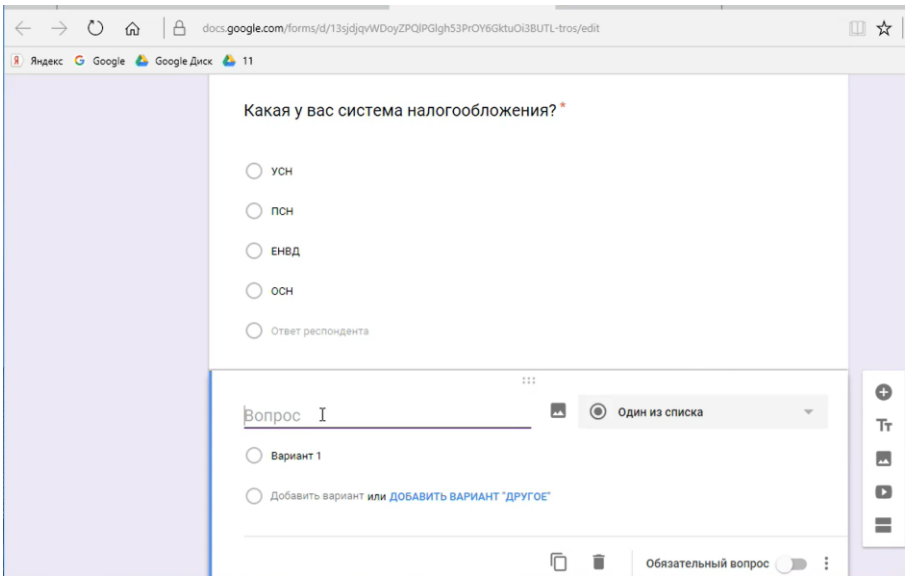

Рисунок 1.3 - Створення питання

“Google Forms” дозволяють пройти опитування кожному користувачу який має аккаунт google. В такий спосіб не можливо встановити особу що пройшла це опитування, а також отримати достовірні результати, так як особа може проходити тестування декілька разів та ввести не своє ім’я.

Сервіс під назвою Surveymonkey надає користувачам можливість створити власні опитування. Одним з критичних недоліків сервісу являється ціна яку потрібно платити незважаючи на користування сервісом.

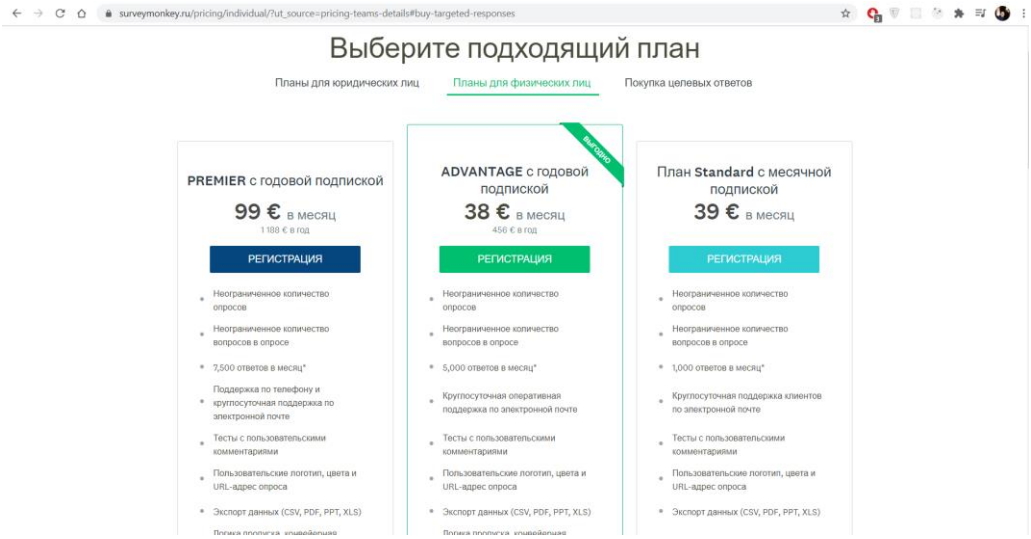

Рисунок 1.4 - Плани підписки surveymonkey

Інтерфейс для створення опитувань достатньо складний хоча і має велику кількість функцій для створення опитування. Сервіс надає функціонал до створення опитувань за шаблоном з запропонованих, в такий спосіб можна не зменшити час на продумування структури опитування.

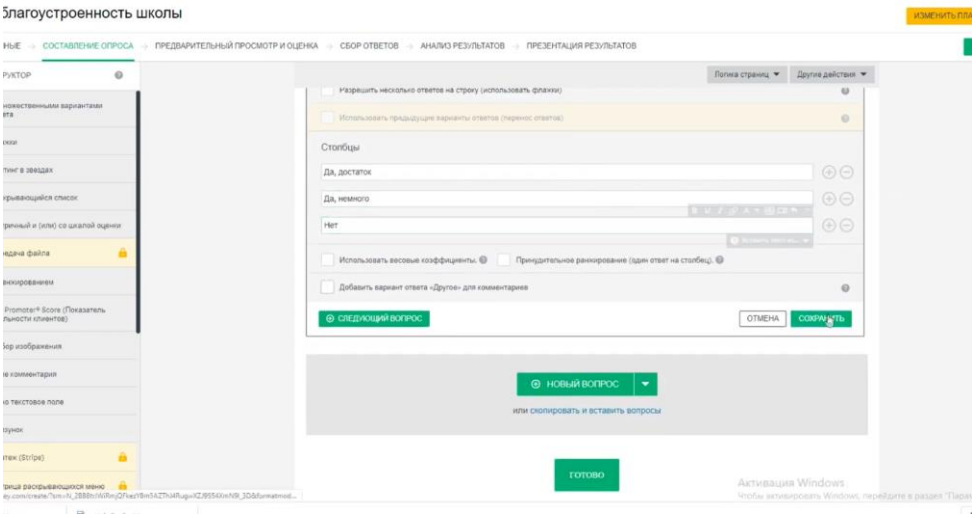

Рисунок 1.5 - Інтерфейс surveymonkey

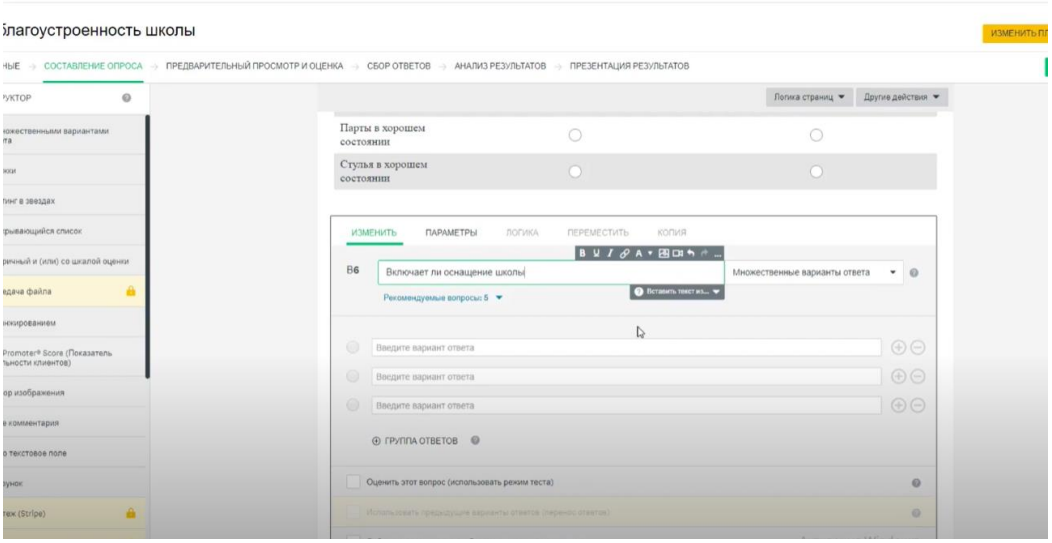

Рисунок 1.6 - Створення/редагування питання на платформі surveymonkey

На ринку сервісів для проходження опитувань багато відомих гравців вже мають пропозицію для користувачів. Основна ціль проходження опитувань великою кількість контингенту. Серед сервісів опитувачів, що мають зручний та ефективний безкоштовний доступ виділяють “Google Forms”. Але, наприклад, досить популярний сервіс “Surveymonkey” має широкий функціонал, але не має безкоштовної версії.

Більшість клієнтів мають такі функції:

- Створення опитувань різних типів: відкритого формату, декілька відповідей та з 1 правильною відповіддю;

- Редагування питань на які відповіли користувачів;
- Збір результатів про відповіді;
- експорт до csv файлу результатів опитувань;
- інші.

Всі ці функції доступні для користувачів, але не дають змоги авторизувати користувача – будь-хто зможе пройти опитування, функціонал не завжди зручний до застосування і не дозволяє експорт великої кількості даних, користувачі мають змогу пройти опитування декілька разів. Тому виникла необхідність в розробці сервісу що зможе ліквідувати згадані недоліки при цьому збереже та покращить функціонал для проходження опитувань.

1.3 Постановка задачі

Ключовою цілю є розроблення сервісу опитування, що матиме інтеграцію з кабінетом та включатиме функціонал для вирішення поставлених задач. Процес виконання завдання поділено на наступні етапи:

1. Виконати огляд технологій, що застосовуються для створення REST API та SPA програми.
2. Порівняти можливості наявних сервісів та їх функціонал.
3. Дослідити особливості роботи фреймворку Laravel, визначити та розглянути технології необхідні для розробки серверного додатку.
4. Змоделювати систему отримання данних по API.
5. Виконати програмну реалізацію системи.
6. Проаналізувати результати.

Для проведення опитування студентів в Сумському державному університеті потрібно складати питання в текстовому редакторі, а потім роздруковувати їх і роздавати студентам. Цей процес займає багато часу, адже потрібно витрачати час працівників університету, щоб роздати, а потім перевірити результати опитувань, а також обробити їх. Проведення опитування – це довготривалий процес який не може гарантувати що

потрібна кількість студентів зможе пройти его для репрезентативної вибірки. В той же час не можливо збирати результати студентів на дистанційній формі навчання. Так як в часи карантину не має змоги провести будь яке опитування, адже студенти в цілях безпеки не відвідують навчальний заклад. Проблема проведення опитувань за умови, що вони не залежать від відвідувань закладу стоїть доволі гостро. Тому потрібно створити сервіс, який дозволить проводити їх дистанційно та оброблювати результати віддалено.

Задача створити api сервірсу, який зможе ідентифіковувати користувачів університету. Це дозволить проходити опитування тільки студентам та працівникам вузу. Користувачі повинні бути ідентифіковані за api “cabinet.sumdu.edu.ua”, щоб однозначно визначить користувача, який зможе проходити опитування. Таким чином перед нами поставлена задача авторизувати користувачів та заборонити іншим проходити опитування від імені іншої людини. Сервіс зобовязаний надавати змогу створення опитування та проходження опитування іншими користувачами, що мають змогу, а отже потрібно розділити їх та класифікувати. Потрібно створити структуру бази та написати програмний код, що дозволяв би деяким користувачам не тільки створювати опитування, а й вносити питання та отримувати результати в режимі реального часу.

2 ВИБІР МЕТОДУ РІШЕННЯ

2.1 Вибір технології

Для програмування веб-інтерфейсів використовуються достатньо багато мов програмування серед яких: Flutter, Microsoft Silverlight, Flash, Vue, Angular, React та інші. Microsoft Silverlight, Flash мають багато проблем пов’язаних з безпекою та не підтримуються компаніями що їх винайшли.

Flutter, достатньо молодий фреймворк від компанії Google, має досить коротку історію та не зарекомендував себе в сфері створення веб-інтерфейсів. Має обмежену кількість бібліотек, які можуть не підтримувати створення веб сторінок. Фреймворк не має достатньої кількості проектів написаних під веб та не може розглядатися як технологія створення клієнтської частини сервісу, адже на ньому не створено достатньо готових рішень для використання в реальному проекті.

Angular не було обрано так як він ставить програміста в залежність від свої бібліотек та не дозволяє використовувати інші, більш того розвиток проекту і підтримка на цій технології вимагає багато часу та не має докладної документації. Фреймворк достатньо сильно змінюється в залежності від версії, що може стати проблемою при переході на нову версію та оновленні залежностей.

Vue має обмежену документацію на англійській мові та мало готових прикладів для використання та ознайомлення. Потрібно відзначити що кількість ресурсів написана на vue недостатня - це робить складнішим обмін знань в рамках розробки на данному фреймворку.

Серед технологій для розробки бекенду був вибір між Symfony та Laravel. З недоліків Symfony потрібно відзначити викокий рівень входу та ORM(doctrine), яка є ресурсоємкою та може викликати проблеми з пам’ятю на сервері.

Беручи до уваги вище сказане, вибір був зроблений на користь Laravel та React. В якості мов програмування було обрано веб-технології, які

працюють з застосуванням html, css, js та php. Перевагами цих технологій є їх відкритість та доступність, завдяки чому більшість проектів в вебі використовують саме ці технології. Для розробки клієнтської частини була використана мова програмування javascript та фреймворк React.

Процес отримання даних побудований за архітектурою "клієнт – сервер", де в ролі сервера виступає бекенд написаний на мові програмування php та з застосуванням фреймворку Laravel. Бекенд був створений в вигляді API, що дозволяє зменшити навантаження на нього.

2.2 Вибір мови програмування

Для створення веб додатків у веб програмуванні поширена мова Php. Вона дозволяє створювати веб сторінки з кодом, що потім буде виконаний і відданий клієнту, як html код. Php створений для написання програм та сервісів для виконання на стороні веб серверу (nginx, apache). Після виконання клієнту буде відданий html файл або json, який являється результатом виконання скрипта.

Абревіатура PHP розшифровується як "Hypertext Preprocessor". Синтаксис мови схожий з іншими мовами такими як C, Java і Perl. PHP простий до опанування. Ключовою перевагою мови є надання розробнику можливості швидкого оформлення веб-сторінки. Важливим фактором що сприяє популярності php є створення HTML документів з вкрапленнями команд PHP.

Також потрібно відзначити можливість створення конфігурації серверу таким чином, щоб HTML-файли оброблялися процесором PHP, так що клієнти не зможуть дізнатися, чи отримують вони статичний HTML-файл або результат виконання скрипту.

На відміну від інших мов, PHP дозволяє створювати якісні Web-додатки за дуже малі терміни, отримуючи продукти, які легко модифікуються і підтримуються в майбутньому. PHP виступає простим для освоєння, і в той же час здатний відповідати критеріям професійних

програмістів. Мова постійно вдосконалюється і нові версії покращують синтаксис що дозволяє писати більш короткий та підтримуваний код.

Цифри продуктивності, які приводять розробники, показує, що версія PHP 7 показує себе значно швидше, в порівнянні з попередніми версіями. Ось кілька пунктів:

• Офіційні тести PHP демонструють, що PHP 7 продуктивніший в обробці запитів в секунду в порівнянні з PHP 5.6.

• Рефакторинг внутрішніх структур даних і додавання додаткового етапу перед компіляцією коду у вигляді абстрактного синтаксичного дерева - Abstract Syntax Tree (AST), привели до чудової продуктивності і більш ефективному розподілу пам'яті.

• Додано класи винятків (Exception і Error) - якщо в попередніх версіях PHP не було можливості обробляти фатальні помилки, то в новому релізі дії, що приводять викидання виключення не призведуть до зупинки програми, а отже не відбудеться завершення скрипту.

Одним з кращих прикладів розвитку мови програмування можна відзначити деякі функції що були додані в нових версіях. В PHP 7.4 були додані:

• Стрілочні функції (Arrow Functions) - створений короткий і зрозумілий синтаксис стрілочних функцій, який змінить стиль написання коду. Новий синтаксис – покращує мову програмування та дозволяє створювати код, який легше читати і робить його більш компактним.

• Оператор розгортування (Speed Operator) – оператор дозволяє розкривати масиви і працювати з кожним елементом окремо. Такий ефект дозволить працювати з основними структурами мови в короткий та зручний спосіб [6].

2.3 Вибір СУБД

Основна функція СУБД - контроль над коректною роботою реляційної інформаційною базою. Кожна модель є певною системою зв'язку між вмістом

різних таблиць. Щоб якісно обробляти інформацію, така модель повинна відповідати певній схемі. Записи складаються в стовпці і включають в себе оригінальний ключ, який зберігається в таблиці.

Найбільш популярними і загальнодоступними СУБД вважаються MySQL і PostgreSQL. Ключовими недоліками PostgreSQL недостатня ефективність при виконанні запитів та складності при обновленні до нових версій. На відмінність від PostgresSQL MySQL проста в використанні – не потребує зусиль щоб встановити та має широкий функціонал який може бути потрібен для реалізації проекту. Так як в проекті використовуються мова програмування php, СУБД що зарекомедувала в роботі з мовою програмування php є саме MySQL.

Одна з найвідоміших повноцінних версій сервера СУБД MySQL надзвичайно продуктивна, вона успішно функціонує в тандемі з різними веб-платформами і додатками. Завдяки доступному функціоналу SQL, технологія управління системою проста. У ній доступний широкий інструментарій для проектування додатків.

Серед головних переваг MySQL варто відзначити:

- комфортність користування;
- високий ступінь захисту, підтримувана в більшості випадків за замовчуванням;
- просте масштабування об'ємних даних;
- швидкість

Якщо говорити про недоліки, то одні з відчутних з них це - деякі обмеження по функціям та проблеми з обробкою даних, від яких безпосередньо залежить оптимізація mysql запитів.

База даних забезпучує наступні операції:

- розподілені операції;
- надійний захист інформації;
- розробка веб-сайтів та додатків, оптимізація їх роботи;

- індивідуальні рішення в роботі над нестандартними проектами.

Mysql використовують в своїй роботі два механізма збереженя данних та їх обробки: InnoDB, MyISAM.

InnoDB - надає стандартні функції транзакцій, сумісних з ACID, разом з підтримкою зовнішніх ключів, стандартно включений у більшість двійкових файлів, що створюються MySQL.

InnoDB став продуктом корпорації Oracle після придбання фінської компанії Innobase у жовтні 2005 р. Програмне забезпечення має подвійну ліцензію; і розповсюджується під загальною публічною ліцензією GNU, але може також мати ліцензію для сторін, які бажають поєднати InnoDB у власному програмному забезпеченні. Важливими перевагами являються пыдтримка InnoDB таких функцій:

- Транзакції SQL
- Зовнішні ключі
- Повнотекстові індекси пошуку, починаючи з MySQL 5.6 та MariaDB 10.0
- Просторові операції, що відповідають стандарту OpenGIS
- Віртуальні стовпці

MyISAM був дефолтною системою зберігання даних для версій системи управління реляційними базами даних MySQL до версії 5.5, базувався на старому коді ISAM, та мав безліч корисних розширень.

MyISAM оптимізований для середовищ із великою кількістю операцій читання, де мало записів або взагалі відсутні. Типовою задаяею, в якій можна віддати перевагу MyISAM, є сховище даних, так як воно включає запити до дуже великих таблиць, і зміни таких таблиць проводиться, коли база даних не використовується (зазвичай вночі).

Причиною того, що MyISAM дозволяє швидко виконувати записи читання, є структура його індексів: кожен запис вказує на запис у файлі даних, а покажчик зміщується з самого початку файлу. Таким чином записи

можна швидко прочитати, особливо коли формат виправлений. Таким чином, ряди мають постійну довжину. Вставки теж прості, оскільки нові зміни додаються до кінця файлу з данними. Але на противагу видалення записів та оновлення є більш складними: видалення залишають порожній простір, в іншому випадку зміщення рядків зміниться; те саме справедливе для оновлень, оскільки довжина рядків стає коротшою; якщо оновлення робить рядок довшим, рядок фрагментований. Завдяки такому механізму індекси MyISAM зазвичай точні. Додатково MyISAM підтримує FULLTEXT індекси та OpenGIS типи даних.

2.4 Вибір середовища розробки

PhpStorm — інтегроване середовище розробки (IDE) для платформи мови програмування PHP, створене корпорацією JetBrains.

PhpStorm прийшов на зміну платформи Eclipse. Середовище побудоване на базі продуктів JetBrains має інтуїтивний інтерфейс який є візитною картою продуктів компанії та не потребує багато часу, щоб почати розробку. PhpStorm встановлюється з великою кількістю вбудованих програм які полегшують розробку програмних продуктів.

Середовище надає засоби для розробки застосунків не тільки для серверу на php, але й дає можливість до написання коду на react/js, а отже створювати та підтримувати кліентські додатки.

Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для створення та запуску скриптів. Середовище встановлюється з вбудованим sql редактором, в якому відбувається рефакторинг схеми данних, генерація скриптів та інше. Також PhpStorm спрощує роботу з системами контролю версій завдяки вбудованому локальному git плагіну, який має простий інтерфейс та інтеграцію з більшістю систем контролю версій (Github, Gitlab). Однією з особливостей є локальна історія змін. Це покращую роботу зі зміненими фрагментами коду та показую внесені зміни.

PhpStorm глибоко аналізує структуру коду, підтримуючи всі можливості мови PHP як в нових, так і в legacy-проектах. Редактор підтримує автодоповнення коду і рефакторинг, запобігає помилкам під час розробки. Одноманітні завдання зручно виконувати прямо в PhpStorm. IDE інтегрована з системами контролю версій, підтримує віддалене розгортання, бази даних і SQL, інструменти командного рядка, Docker, Composer, REST-клієнт і багато інших інструментів. Середовище безпечно перетворює код за допомогою надійних методів рефакторинга: перейменування, переміщення та видалення, вилучення методів, введення змінних, переміщення елементів вгору / вниз, зміни сигнатури і інших. Рефакторинг, що враховує особливості конкретної мови допоможе застосувати зміни по всьому проекту за пару кліків. При цьому будь-яке перетворення можна скасувати.

До складу також включені пристосовані під особливості платформи Laravel розширені інструменти рерайтингу коду, перевірки та оновлення сумісності з минулими випусками, пошук проблем з продуктивністю, моніторинг споживання пам'яті та оцінка зручності використання. У редакторі працює режим швидкого внесення правок та постановки міток до яких можна повернутися. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою фреймворку.

Наявний великий обсяг додаткових модулів та розширень. Головною метою яких є покращення та автоматизація процесу роботи програміста з кодом та фреймворком. В такий спосіб розробник отримує розширений функціонал який може покращити роботу з новою технлогією.

Графічний дебагер PhpStorm не вимагає додаткових налаштувань. Він дуже наочно візуалізує, що відбувається у вашому додатку на кожному етапі дебагінгу. Дебагер працює з Xdebug і Zend Debugger і може використовуватися як локально, так і віддалено. IDE також підтримує модульне тестування з PHPUnit, BDD з Behat і інтегрується з профілювальником.

Деякі особливості середовища PhpStorm:

- Підтримка SQL і широкого спектру баз даних (Зміна структури бази даних, створення коду для міграції структури данних, експорт результатів sql запиту в файл/буфер обміну, можливість працювати з процедурами виконання та редагування в середовищі).
- Віддалене управління та розміщення додатків і автоматична синхронізація з використанням FTP, SFTP, FTPS і ін. протоколів.
- Підтримка системам управління версіями (Git - включаючи треті сервіси GitHub, Subversion, Mercurial, Perforce, CVS, TFS), що дозволяє робити команди git, наприклад commit, merge, diff і інші, прямо з PhpStorm.
- PHP UML (Діаграми класів UML для PHP коду з рефакторингом, що викликаються прямо з діаграми).
- Інтергація з Docker, Vagrant, SSH консолі і віддалених інструментів
- Google App Engine підтримуванна в IDE.
- Використання різних сполучень клавіш для збільшення ефективності.
- Автодоповнення для ECMAScript, HTML і CSS.
- Підтримка HTML5.
- Правки в коді відображаються відразу в браузері без перезавантаження сторінки(Live edit).
- Підтримка CSS / SASS / SCSS / LESS (автодоповнення коду, підсвічування помилок, валідація та т. Д.).
- Emmet.
- Зручні переходи по коду і пошук використання (перехід до оголошення змінної / ідентифікатором, пошук випадків використання)
- Підтримка сучасних стандартів ECMAScript.
- Підтримка фреймворків Laravel, Yii2 та інших.

2.4 Вибір backend фреймворку

Laravel - це PHP фреймворк з відкритим кодом, створений Тейлором Отвеллом для розробки додатків що наслідують шаблон MVC.

Він був створений як аналог фреймворку Codeigniter, який не мав в достатній кількості корисних функцій для розробки веб-додатків. В якості ядра Laravel виступають компоненти іншого фреймворка - Symfony (навігація, підхід до створення бази даних) [7].

За допомогою менеджера пакетів Composer, фреймворк Laravel дозволяє розширювати вбудований функціонал і підключати інші компоненти (готові рішення) для веб-додатку.

Eloquent ORM створена, як реалізація ActiveRecord, дозволяє встановити відносини між об'єктами бази даних та веб-додатками і створювати зручні запити для маніпуляції даними [8].

Механізм автозавантаження класів дозволяє не підключати вручну файли через include і зменшує кількість класів що використовуються в проекті.

Доступна система міграцій спрощує розгортання та оновлення бази даних і спрошує маніпулювання структурою бази. Також в фреймворку наявний механізм додавання тестових даних через сіди [9].

У Laravel є вбудована підтримка движка шаблонів Blade, за допомогою якого можна робити HTML шаблони з включеннями необхідних структур використовуючи спеціальний blade синтаксис.

Для створення структур фреймворку за короткий час доцільно використовувати Artisan – командний інтерфейс для введення вбудованих команд, а також створення своїх власних.

Широкий набір функцій для роботи з даними. З подібних функцій можна відзначити dd() яка виводить в зручному вигляді інформацію про об’єкт Laravel/Php. Функція виводить інформацію змінної в більш зрозумілій

формі, розділяючи дані на дерево атрибутів і значень, з можливістю пошуку і переходу по ним [10].

2.5 Вибір frontend фреймворку

React - JavaScript-бібліотека для роботи з інтерфейсами користувача (UI), яку створили розробники Facebook. Бібліотеку почали використовувати на сайті цієї соціальної мережі в 2011 році. А в 2013 році Facebook відкрив вихідний код React.

За допомогою React розробники створюють клієнтську частину, яка змінює відображення без перезавантаження сторінки. Завдяки цьому додаток швидко реагує на дії користувача, наприклад, заповнення форм, застосування фільтрів, додавання товарів в корзину і так далі.

React застосовують для відтворення компонентів для інтерфейсу користувача. Також бібліотека може повністю управляти фронтендом. В цьому випадку React використовують з бібліотеками для управління станом і роутинга, наприклад, Redux і React Router.

Ключові особливості React: декларативність, універсальність, компонентний підхід, віртуальний DOM, JSX

Одна з ключових особливостей React - універсальність. Бібліотеку можна використовувати на сервері і на мобільних платформах за допомогою React Native. Це принцип Learn Once, Write Anywhere або «Навчіться один раз, пишіть де завгодно».

Ще одна важлива особливість бібліотеки - декларативність. За допомогою React розробник описує, як компоненти інтерфейсу виглядають в різних станах. Декларативний підхід скорочує код і робить його зрозумілим.

React заснований на компонентах, це ще одна ключова особливість бібліотеки. Кожен компонент повертає частину призначеного для користувача інтерфейсу зі своїм станом. Об'єднуючи компоненти, програміст створює завершений інтерфейс веб-додатки.

Важлива особливість React - використання JSX. Це розширення синтаксису JavaScript, яке зручно використовувати для опису інтерфейсу. JSX схожий на HTML, проте це все-таки JavaScript. Приклад JSX можна побачити нижче:

JSX дозволяє писати JavaScript код за допомогою готових компонентів, які практично повністю повторюють HTML. Це спрощує розробку.

До важливих особливостей React відноситься використання віртуального DOM (Virtual DOM). Віртуальний DOM - об'єкт, в якому зберігається інформація про стан інтерфейсу. При зміні стану, наприклад, після відправки форми або натискання кнопки, React розраховує різницю між старим і новим станом. Після цього бібліотека малює нове стан. Використання віртуального DOM дозволяє бібліотеці ефективно оновлювати реальний DOM.

Бібліотека передає дані від великих до меньших. В такий спосіб потік даних йде від батьківського компонента до нащадків, а отже дочерні компоненти не зможуть впливати на батьківські.

React надзвичайно легка бібліотека, що дозволяє користувачам завантажувати меншу кількість коду та збільшує швидкодію сервісу. Відмінною рисою є легкість міграції між версіями. Також можливо користуватися офіційними засобами для автоматизації процесу міграції.

React максимально гнучкий і багатий в плані функціоналу. Розробники фреймворку постійно підтримують його та виступають на конференціях формуючи величезне комьюніті розробників, що збільшує популярність фреймворку. Переходи на нові версії бібліотеки досить легкі та не потребую зусиль розробників, а залежності постійно оновлюються для того щоб відповідати новітнім стандартам безпеки.

В поєднанні з ES6/7 може показувати підвищену швидкодію та працювати під підвищеним навантаженні. Але така швидкість не підвищує поріг входу і дає можливість новим розробникам писати професіональний

код відразу ж. Документація фреймворку розгалуджена та переведена на більшість мов світу.

React - популярна бібліотека для роботи з UI. До її основних особливостей відноситься декларативність, компонентний підхід, універсальність, використання віртуального DOM і JSX. React бібліотека входить до числа найпопулярніших технологій для розробки frontend та написання клієнтської частини.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Розробка моделі інформаційної системи

Одним з найважливіших етапів створення сервісу є створення нормалізованої бази даних в якій будуть зберігатися данні користувачів. Ключовими вимогами до бази даних:

1. Створення таблиці користувачів яка дозволить зберігати та авторизувати користувачів в рамках сервісу.
2. Створення таблиці опитувань в якій буде зберігатися інформація про створені адміністратором опитування з інформацією про властивості опитування.
3. Таблиця с питаннями повинна бути пов’язана з опитуваннями та належати до одного з опитувань.
4. У кожного питання повинні бути наперед задані варіанти відповідей з яких кожен користувач зможе вибрати відповідь на питання опитування.
5. Для кожного варіанту потрібно зберігати відповідь для опитування. Таким чином відповіді будуть доступні для аналізу та обробки.
6. Створення користувачів з правами та властивостями, які передаються з кабінету та ідентифікувати їх за факультетом, групою, спеціальністю.

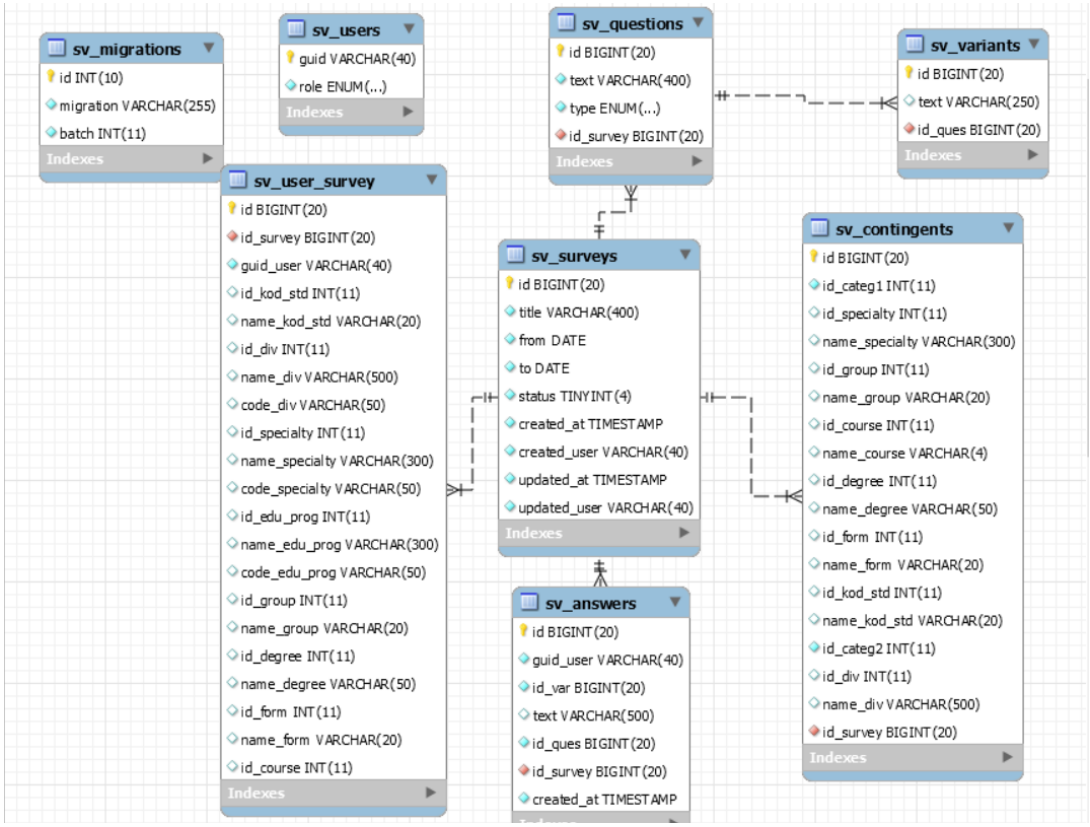

Рисунок 3.1 - Схема бази даних побудована з використанням mysql workbench

В Laravel основні таблиці формуються на основі міграцій, які розробник повинен написати. Також Laravel має зручний інтерфейс створення міграцій за допомогою artisan. Цей інтерфейс за допомогою консолі дозволяє створювати міграції та більш того створювати їх з правильною назвою класу.

```
php artisan make:migration create_flights_table
```

Виконавши таку команду можна створити міграцію. В основному одна міграція відповідає 1 таблиці в базі даних. Також консольний інтерфейс artisan дає можливість виконувати всі міграції до бази даних зручною командою.

```
php artisan migrate
```

Після створення таблиць для того, щоб змінити деякі данні в таблицях існує два способи щоб зробити це. В залежності від того як розміщений проект та чи перебуває проект в production.

Для локального тестування бази даних допустимо перезапустити міграції. Такую дію можливо провести за допомогою команди artisan:

```
php artisan migrate:refresh
```

В такий спосіб можливо перезапустити всі міграції проекту, але це призведе до того, що всі дані проекту будуть стерті. Тому такий спосіб не може бути застосований до проекту який знаходиться в production, та яким користуються користувачі. В такому випадку потрібно створити нову міграцію та виконати її.

3.2 Розробка API backend на Laravel

Створимо міграції відповідно до вимог бази даних.

В таблиці користувачів повинен зберігатися унікальний ідентифікатор користувача guid. Ідентифікатор надходить з кабінета та дає можливість ідентифікувати користувача та зробити висновок про категорію до якої він належить. Кожний користувач повинен бути авторизований як користувач сервісу або адміністратор. Адміністратор матиме розширені права, а саме зможе створювати опитування та питання, назначати контингент та дивитися результати опитування.

2014_10_12_0_create_users_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{

    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->string('guid', 40)->primary();
            $table->enum('role', ['admin', 'user'])->default('user');
        });
    }
```

```
     /**
      * Reverse the migrations.
      *
      * @return void
      */
     public function down()
     {
         Schema::dropIfExists('users');
     }
 }
```

В таблиці surveys будуть зберігатися дані про опитування, які користувачі сервісу мають змогу пройти. В опитування повинна бути назва, яку вирішено було зробити текстовим полем з довжиною 400 символів. Також опитування повинно мати дату закінчення та дату початку. Разом з цими полями потрібно зберігати статус опитування так як опитування може бути призупинене та не доступне для проходження користувачами.

2020_04_24_1_create_surveys_table.php

```
<?php
  use Illuminate\Database\Migrations\Migration;
  use Illuminate\Database\Schema\Blueprint;
  use Illuminate\Support\Facades\Schema;
  class CreateSurveysTable extends Migration
  {    public function up()
      { Schema::create('surveys', function (Blueprint $table) {
              $table->id();
              $table->string('title', 400);
              $table->date('from')->index();
              $table->date('to')->index();
              $table->tinyInteger('status');
              $table->timestamp('created_at')->useCurrent();
              $table->string('created_user',40);
              $table->timestamp('updated_at')->useCurrent();
              $table->string('updated_user',40);
          });
      }
      public function down(){ Schema::dropIfExists('surveys');}
   }
```

В таблиці питань основними полями є назва питання та його тип з вибором однієї чи кількох варіантів відповіді. Однією з відмінностей є ключ, який вказує на зв’язок з опитуванням до якого належить.

2020_04_24_2_create_questions_table.php

```
<?php

  use Illuminate\Database\Migrations\Migration;
  use Illuminate\Database\Schema\Blueprint;
  use Illuminate\Support\Facades\Schema;

  class CreateQuestionsTable extends Migration
```

```
  {
      /**
       * Run the migrations.
       *
       * @return void
       */
      public function up()
      {
          Schema::create('questions', function (Blueprint $table) {
              $table->id();
              $table->string('text', 400);
              $table->enum('type', ['multi' ,'single']);
              $table->foreignId('id_survey')->constrained('surveys')-
  >onDelete('cascade');
              $table->index('id_survey');
          });
      }


      public function down()
      {
          Schema::dropIfExists('questions');
      }
  }
```

В таблиці відповідей будуть зберігатися відповіді для кожного користувача що пройшов опитування. Для цілей збереження та знаходження потрібного користувача в таблицю додано поле “guid_user” і створено індекс задля пришвидшення виконання запитів. Також кожен рядок таблиці має мітку часу та посилання на питання до якого надана відповідь, варіант на який надана та опитування до якого відноситься.

2020_04_24_4_create_answers_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateAnswersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('answers', function (Blueprint $table) {
            $table->id();
            $table->string('guid_user', 40)->index('guid_user_idx');
            $table->bigInteger('id_var')->index();
            $table->bigInteger('id_ques')->index();
            $table->foreignId('id_survey')->constrained('surveys')->onDelete('cascade');
            $table->timestamp('created_at')->useCurrent();
            $table->index('id_survey');
```

```
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('answers');
    }
}
```

Таблиця контингентів відповідає за інформацію про користувачів і визначає до якого класу користувачів можна віднести користувача. Саме в цій таблиці будуть зберігатися данні про клас користувача, чи є користувач викладачем, студентом. Також деякі поля(id_div) можуть вказувати інформацію про підрозділ університету до якого належить студент чи викладач. В сукупності ці поля дозволяють точно визначити до якого класу належить користувач.

2020_04_24_5_create_contingents_table.php

```
<?php

  use Illuminate\Database\Migrations\Migration;
  use Illuminate\Database\Schema\Blueprint;
  use Illuminate\Support\Facades\Schema;

  class CreateContingentsTable extends Migration
  {
      /**
       * Run the migrations.
       *
       * @return void
       */
      public function up()
      {
          Schema::create('contingents', function (Blueprint $table) {
              $table->id();
              $table->integer('id_categ1')->default(0);
              $table->integer('id_specialty')->nullable();
              $table->integer('id_group')->nullable();
              $table->integer('id_course')->nullable();
              $table->integer('id_degree')->nullable();
              $table->integer('id_categ2')->default(0);
              $table->integer('id_div')->nullable();
              $table->foreignId('id_survey')->constrained('surveys')-
  >onDelete('cascade');
              $table->index('id_survey');

          });
      }

      /**
```

```
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('contingents');
    }
}
```

Для тестування та заповнення бази даних тестовими даними Laravel пропонує механізм Seed. Він дозволяє створювати класи в яких буде відбуватися заповнення бази даних.

В кожному seed описуються правила заповнення бази даних потрібними сутностями та можна створити зв’язки між ними.

SurveySeeder.php

```
<?php

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use App\Models\Survey;

class SurveySeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        Survey::insert([
            [
                'title' => 'Опитування щодо задоволення дистанційним навчальним
процесом під час карантину',
                'from' => '2020.10.01',
                'to' => '2020.11.30',
                'status'=> 2,
                'created_user'=>'44dde7f4-83c3-e511-867d-001a4be6d04a',
                'updated_user'=>'44dde7f4-83c3-e511-867d-001a4be6d04a'
            ],
            [
                'title' => 'Опитування щодо проблем першокурсників',
                'from' => '2020.10.01',
                'to' => '2020.11.21',
                'status'=> 1,
                'created_user'=>'8248eef4-83c3-e511-867d-001a4be6d04a',
                'updated_user'=>'8248eef4-83c3-e511-867d-001a4be6d04a'
            ]
        ]);}
}
```

QuestionSeeder.php

```
<?php
use Illuminate\Database\Seeder;
```

```
  use App\Models\Question;
  class QuestionSeeder extends Seeder
  {
      /**
       * Run the database seeds.
       *
       * @return void
       */
      public function run()
      {
          Question::insert([
              [
                  'text' => 'Викладачі використовували інформаційні сервіси (mix, cabinet
  тощо) для дистнаційного навчання?',
                  'type' => 'single',
                  'id_survey' => 1
              ],
              [
                  'text' => 'Які сторонні сервіси використовувались для спілкування з
  викладачем?',
                  'type' => 'multi',
                  'id_survey' => 1
              ],
]);
      }
  }
```

UserSeeder.php

```
<?php
  use Illuminate\Database\Seeder;
  use App\Models\User;
  class UserSeeder extends Seeder
  {
         public function run()
      {
          User::insert([[
                  'guid' => '44dde7f4-83c3-e511-867d-001a4be6d04a',
                  'role' => 'admin'
              ],[
                  'guid' => '8248eef4-83c3-e511-867d-001a4be6d04a',
                  'role' => 'admin'
              ],[
                  'guid' => 'ac31345f-c580-e711-8194-001a4be6d04a',
                  'role' => 'user'
              ],[
                  'guid' => '9263185e-c580-e711-8194-001a4be6d04a',
                  'role' => 'user'
              ]]);
      }
  }
```

Роутинг фреймворку знаходиться в файлах php які знаходяться в директорії routes. Файли автоматично загружаються фреймворком Laravel. У файлі routes/web.php наведено маршрути для користувачів веб-інтерфейсу, також цими маршрутам користуються користувачі, що потрапили до сервісу з пошукових систем та перейшли за посиланням. Маршрути web

забезпечують стейт сессії та CSFR захист, але маршрути routes/api.php не підтримують стейт при запитах й входять до посердників api.

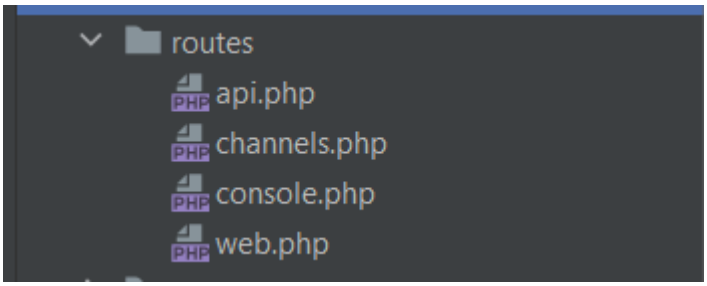

Рисунок 3.2 - Структура в папці routes

Параметри маршрутів повинні бути заключені в фігурні дужки та бути із буквених символів. Вони не можуть мати в собі символ “-” та “_”. Маршрути можна об’єднувати в групи. Групи маршрутів можуть використовувати спільні атрибути: простір імен, посередників та інше без необхідності вказувати для кожного маршруту окремо. Спільні атрибути вказуються через синтаксис `Route::group()`.

В проекті використовується роутинг з файлу web.php. Користувачі переходять з сайту cabinet.sumdu.edu.ua, а потім всі їх дії оброблює контролер з index методом за кореневим маршрутом “/”. Також для того щоб забезпечити коректне відображення малюнка в кабінеті задамо роут для повернення зображення сервісу на кабінет “/photo”.

```
<?php

use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', 'IndexController@index');
Route::get('/cache/clear/{key}/{tags}', 'IndexController@cacheClear');
Route::middleware('auth.cabinet')->get('/photo', 'IndexController@photo');
Route::middleware('auth.cabinet')->get('/logout', 'IndexController@logout');
Route::middleware(['auth.cabinet', 'role:admin'])->any('adminer',
'\Aranyasen\LaravelAdminer\AdminerController@index');
```

```
Route::get('/{path}', 'IndexController@index')->where('path','(surveys(.*)?)');
```

Наведений роутинг не може використовуватися для створення api, так як він повинен оброблювати переходи на сервіс. Для обробки даних SPA потрібно створити маршрути в файлі api.php. Для маршрутів не має збереженого стану, а отже дані зможе отримати будь-хто. Для вирішення цієї проблеми було створено посередника, який відповідає за авторизацію користувачів у сервісі. Користувачів api ми будемо авторизувати за допомогою запиту до кабінету. У випадку помилки перенаправляємо користувачів до кабінету з помилкою.

```
<?php
  namespace App\Http\Middleware;
  use Closure;
  use Illuminate\Support\Facades\Auth;
  class CabinetAuthenticate
  {
      public function handle($request, Closure $next)
      {
          if ( !Auth::check() ) {
              if ( $request->ajax() ) {
                  return response()->error(trans('auth.failed'), 401);
              }
              return redirect()-
  >away(config('cabinet.url').'?error='.cabinet_message(trans('auth.failed')));
          }
          return $next($request);
      }
  }
```

Після написання посередника ми повинні зареєструвати його в класі Kernel.php

```
protected $routeMiddleware = [
      'auth.cabinet' => \App\Http\Middleware\CabinetAuthenticate::class,
      'role' => \App\Http\Middleware\CheckRoleUser::class,
];
```

Для користувачів зі статусом адміністратора ми перевіряємо в користувачі перевіряємо поле role. У випадку коли роль не співпадає користувачі будуть перенаправлені зі статусом 403.

```
<?php

namespace App\Http\Middleware;
```

```
use Closure;
use Illuminate\Support\Facades\Auth;

class CheckRoleUser
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @param array $roles
     * @return mixed
     */
    public function handle($request, Closure $next, ...$roles)
    {
        if  ( !in_array(Auth::user()['role'],  $roles) ) {
            if ( $request->ajax() ) {
                return response()->error(trans('auth.forbidden'), 403);
            }
            return redirect()-
>away(config('cabinet.url').'?error='.cabinet_message(trans('auth.forbidden')));
        }

        return $next($request);
    }
}
```

В маршрутизації наведено маршрути для створення опитувань, питань та відповідей. Також наведено маршрути для створення копій опитувань та експорту результатів опитувань. Для автокомплітів потрібен доступ до даних з серверу, для цієї цілі наведено роут для доступу до контингенту, щоб повернути дані контингенту для інтерфейсів.

```
<?php

use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

Route::middleware(['auth.cabinet'])->group(function() {
    Route::get('/user', 'UserController@show');
    Route::post('/answers/survey/{idSurvey}', 'AnswerController@store');

    Route::apiResource('surveys', 'SurveyController', ['parameters' => ['surveys' =>
'id']])->only(['index','show']);

    Route::middleware('role:admin')->group(function(){
        Route::apiResource('surveys', 'SurveyController', ['parameters' => ['surveys' =>
'id']])->except(['index','show']);
```

```
        Route::post('/surveys/copy/{id}', 'SurveyController@copy');

        Route::get('/surveys/result/export/{id}', 'SurveyController@export');

        Route::get('/cabinet/autocomplete/{contingentName}',
'CabinetController@autocomplete');

        Route::apiResource('contingents', 'ContingentController', ['parameters' =>
['contingents' => 'id']])->except(['index','show']);

        Route::apiResource('questions', 'QuestionController', ['parameters' =>
['questions' => 'id']])->except(['index', 'show']);

    });
});
```

Для обробки запитів потрібно додати сервіс до системи кабінет. Це можна зробити за допомогою системи “Кабінет СумДУ”. Для цього перейдемо за роутом (https://cabinet.sumdu.edu.ua/settings) та в секції інформаційні сервіси потрібно додати сервіс розміщений на віддаленому сервісу.

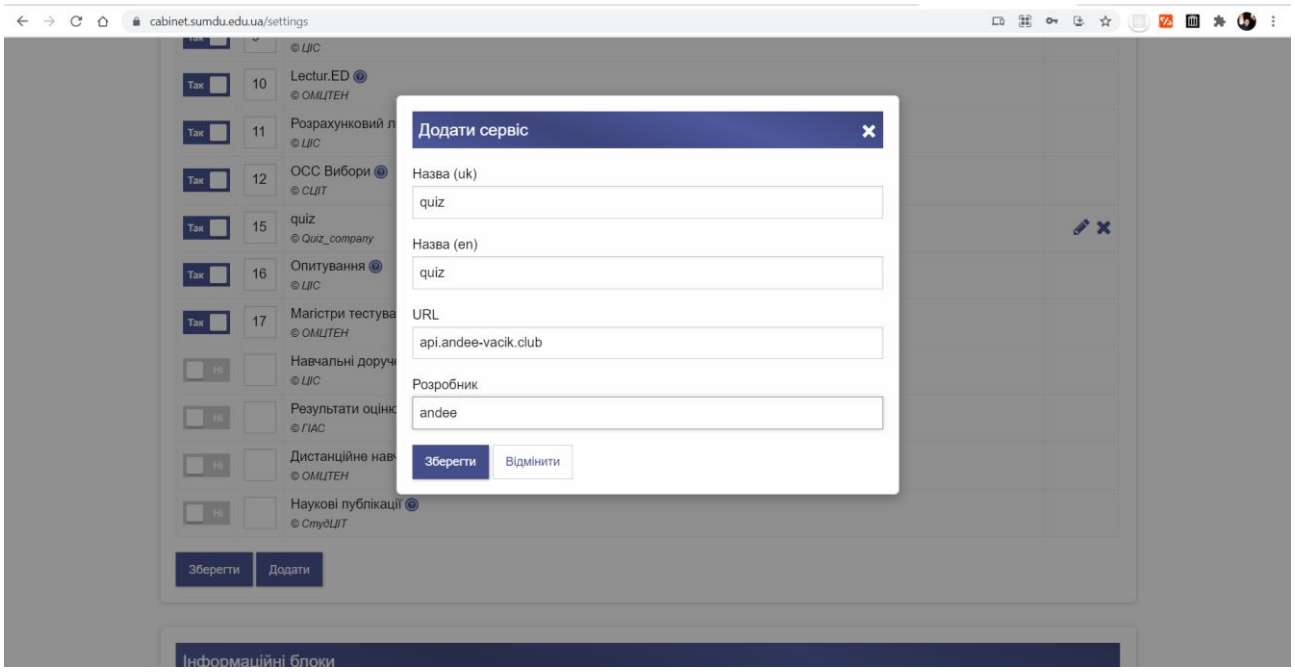

Рисунок 3.3 - Додання проекту, як сервісу кабінету

За авторизацію користувачів відповідає IndexController.php та CabinetGuard.php.

В класі CabinetGuard.php метод validate відповідає за валідацію користувача та перевіряє його доступ за допомогою key який знаходиться у користувача.

```
public function validate(array $credentials = [])
{
    $key = $credentials['key'];
    if ( $key ) {
        if ( !($session_key = session('key')) || $session_key != $key ) {
            session()->invalidate();
            $credentials['sid'] = session()->getId();
            $person = $this->provider->retrieveByCredentials($credentials);
            session(['key' => $key]);
            session(['person' => $person]);
            $user = User::firstOrCreate(['guid' => $person['guid']]);
            $user->person = $person;
            session(['user' => $user]);
        }
    } else if ( !session('key') ) {
        throw new CabinetAuthKeyException( trans('auth.key_absent') );
    }

    $user = session('user');
    $this->setUser($user);
    return true;
}
```

Кожен запит що надходить до нашого сервісу оброблюється і проходить валідацію з IndexController.php в якому відбувається перевірка ключа та mode кабінету. Mode представляє цифру за якою кожний сервіс з яким відбувається інтеграція повинен забезпечити відповідну відповідь з даними.

```
public function index(Request $request, Guard $cabinetGuard, SurveyService
$surveyService)
{
    $mode = $request->query('mode', 0);

    $modes = config('cabinet.modes');
    if ( !in_array($mode, array_filter($modes, fn($value) => $value !== 4)) ) {
        try {
            $cabinetGuard->validate(['key' => $request->query('key', []), 'sid' =>
session()->getId()]);
        } catch (CabinetResponseException | CabinetAuthKeyException $e) {
            if ( $mode == 4 ) return icon_service_response();
            return redirect()->away(config('cabinet.url').'?error='.cabinet_message($e-
>getMessage()));
        }
    }

    switch($mode) {
        case $modes[3]:
            return response(trans('labels.head.description_shot'),200)->header('Content-
type', 'text/plain');
        case $modes[4]:
            return icon_service_response($surveyService->countAll());
        case $modes[100]:
            return response('', 200)->header('X-Cabinet-Support',25);
        case $modes[101]:
            $sid = $request->query('sid');
            if ( $sid ) {
                session()->setId($sid);
                session()->start();
                session()->invalidate();
```

```
                exit();
            }
    }

    return view('layout');
}
```

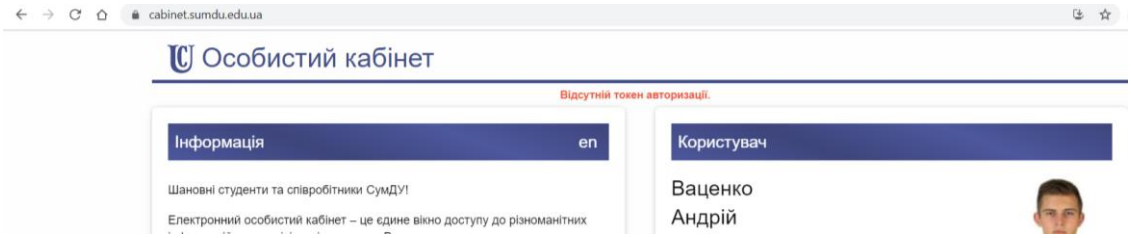

Рисунок 3.4 - Вивід помилки за допомогою інтеграції з кабінетом

Щоб забезпечити приватність користувачів та не допускати витоку даних потрібно розробити метод виходу для користувачів сервісу. Метод видаляє сесію з сервісу та вилогінює користувача.

```
public function logout(CabinetRepository $cabinetRepository)
{
    $key = session('key');
    session()->invalidate();
    $response = $cabinetRepository->logout($key);

    if ( isset($response['status']) && $response['status'] === 'OK' ) {
        return redirect()->away(config('cabinet.url'));
    }  else {
        return redirect()-
>away(config('cabinet.url').'?error='.cabinet_message(config('messages.cabinet.error_logo
ut')));
    }
}
```

В методі вище викликається функція CabinetGuard.php. Функція представляє з себе відправку http запиту до кабінету в якому відбудеться виклик api методів кабінету. В такий спосіб відбувається logout користувачів в сервісі, але не в кабінеті СумДУ.

```
public function logout($key) {
    return HTTP::asForm()->get(config('cabinet.logout_api'), [
        'key' => $key
    ])->throw()->json();

}
```

Наступна функція представляє з себе метод в якому відбувається робота

```
public function retrieveByCredentials(array $credentials = [])
{
    $auth_response = $this->cabinetRepository->gerPersonInfo($credentials['key'],
$credentials['sid']);
    $this->validateCredentials($this->user, is_array($auth_response) ? $auth_response :
[]);
    return $auth_response['result'];
}
```

Наступний метод відправляє http запит з метою отримати валідну відповідь від кабінету та авторизувати користувача в нашому сервісі.

```
public function gerPersonInfo($key, $sid) {
    return HTTP::asForm()->get(config('cabinet.get_person_info'), [
        'key' => $key,
        'token' => config('cabinet.service_token'),
        'sid' => $sid
    ])->throw()->json();
}
```

Для створення опитувань та оновлення опитувань фреймворк використовує контролер який вказано в маршрутизації. Фреймворк має особливі контролери, що дозволяють створювати CRUD методи в новому класі. Такі контролери називають ресурсними. В контролері з опитуваннями відбувається створення опитування в методі store. Дані що надходять до методу уже пройшли валідацію.

Наведено методи створення та оновлення опитувань.

SurveyController.php

```
public function store(SurveyStoreRequest $request)
{
    return response()->success($this->service->create($request), null, 201);
}
```

```
public function update($id, SurveyStoreRequest $request)
{
    return response()->success($this->service->update($id, $request));
}
```

Валідація винесена до спеціальних класів фреймворку під назвою validation requests. В реквестах в спеціальному методі rules задаємо правила валідації відповідно до встановлених шаблонів. Існує досить багато правил валідації одні з популярних правил:

- required – запит повинен мати поле та не бути пустим

- nullable – поле може бути пустим
- string- в полі повинен бути рядок
- max – максимальне значення

У випадку не відповідності валідації до правил фреймворк повертає помилку валідації зі статусом 422.

SurveyStoreRequest.php

```
public function rules()
  {
      return [
          'survey' => 'required|array',
          'survey.title' => 'required|string|max:400',
          'survey.from' =>
  ['required','date_format:d.m.Y'/*,'after_or_equal:'.date('d.m.Y')*/],
          'survey.to' => 'required|date_format:d.m.Y|after_or_equal:survey.from',
          'survey.status' => 'required|integer'
      ];
  }
```

Логіка функціоналу винесена до сервісу. Таким чином логіка інкапсульована в сервісах – це спрощує розробку та підтримування коду в довгостроковій перспективі. Код наведений нижче створює модель опитування з полями, які отримуються з запиту та додається guid користувача.

```
public function create(FormRequest $request)
  {
      return Survey::create(
          $request->only($this->fields)['survey'] + [
              'created_user' => Auth::user()->guid,
              'updated_user' => Auth::user()->guid
          ]
      );
  }

  public function update($id, FormRequest $request)
  {
      $survey = Survey::findOrFail($id);
      $survey->update(
          $request->only($this->fields)['survey'] +
          ['updated_user' => Auth::user()->guid]
      );
      return $survey;
  }
```

Для експорту даних результатів опитувань ми використовуємо бібліотеку Laravel Excel. Бібліотека дозволяє експортувати моделі фреймворку та оброблювати дані при експорті. Пакет дозволяє обробити та

агрегувати дані перед експортом. Важливо відмітити, що він підтримує розширення файлів csv, xlsx та інші.

```
public function export($id) {
    return Excel::download(new SurveyResultExport($id), 'result.xlsx',
\Maatwebsite\Excel\Excel::XLSX);
}
```

3.3 Створення фронтенд частини на базі фреймворку React app

Однією з головних частин будь-якої фронтенд програми є функціонал спілкування з сервером. За допомогою цієї функції відбувається запит даних frontend react для отримання даних. Запити використовують пакет axios, що був імпортований раніше. Цей пакет спрощую запити до бекенду та дозволяє встановлювати interceptors до запиту. Також пакет дозволяє передавати додаткові параметри та конфігурацію до запиту. В такий спосіб відбувається спрощення і уніфікування процесу запиту даних до backend.

Метод повертає проміс що буде в одному із статусів або resolved або rejected. В методі під назвою fetchData відбувається запит по базовому url з додаванням префіксу. Такий спосіб має великі преваги адже дозволяє додавати для нових версій префікси в залежності від додавання нових роутів на бекенді. Прикладом можна назвати префікси “api”, “api/v2”, “api/v1”, “api/v2/config”. В такий спосіб ми залишаємо простір для розвитку і розширення backend та робимо клієнтський код відкритий до розширення але закритий до модифікації [11].

```
import axios from 'axios';

import { API_PREFIX } from '../../constants/dataFetch';

const fetchData = ({ url, method = 'GET', data = {}, params = {}, configs = {} }) => {
    return new Promise(async (resolve, reject) => {
        try {
            const response = await axios({
                url: `${API_PREFIX}${url}`,
                method,
                data,
                params,
                headers: {
                    'X-Requested-With': 'XMLHttpRequest',
                },
                ...configs
            });
```

```
            resolve(response)
        } catch (error) {
            reject(error)
        }
    });
};

export default fetchData
```

Основна логіка відображення даних відбувається у функції App.js. Вона є точкою запуску react програми. Для побудови компонетів та винесення логіки вищого рівня, в класі App потрібно створити стейт в якому ми збережемо користувача.

```
const [ userState, setUserState ] = useState(useContext(UserContext));
```

Для перевірки що користувач має доступ до деяких компонентів в дереві є потреба в передачі даних з стейту до компонентів нижчого рівня. Цей процес буде реалізовано через провайдер react до якого передається стейт та функція зміни стейту [12].

```
<UserContext.Provider value={{userState, setUserState}}>
    <StickyMessageSurveyContext.Provider value={{stickySurveyState,
setStickySurveyState}}>
        <BrowserRouter>
            <Layout />
        </BrowserRouter>
    </StickyMessageSurveyContext.Provider>
</UserContext.Provider>
```

Для збереження даних про користувачів в реакт можна використати контекст. Він дозволяє зберігати дані які будуть потрібні в компонентах нижчого рівня.

Таким чином потік даних йде від батьківських компонентів до компонентів нижчого рівня. Заданий спосіб передачи даних ідеально підходить, якщо потрібно передавати Ui тему чи мову. В сервісі в контексті зберігається користувач, якого ми отримуємо з бекенду та його права. Створимо об’єкт, який буде мати в собі guid користувача, role, та прапорець чи є користувач адміністратором.

userContext.js

```
import { createContext } from 'react';
```

```
const userState = {
    guid: '',
    role: 'guest',
    isAdmin: false
};

const UserContext = createContext(userState);

export default UserContext;
```

Будь-яка складна програма потребує навігації. Для поліпшення та розбиття коду по класам ми використали офіційний пакет react router. Пакет дозволяє сворювати роутинг та не заважає потоку даних. В ньому є широкий функціонал за допомогою якого можна налаштувати маршрутизацію для фреймворку.

```
<Switch location={location}>
    <Route exact path="/">
        <Surveys/>
    </Route>
    <Route exact path="/surveys/:id([0-9]+)" >
        <Suspense fallback={<ThreeCircleLoader/>}>
            <Survey />
        </Suspense>
    </Route>
    <PrivateRoute exact path="/surveys/edit/:id([0-9]+)" role={"admin"} user={user}>
        <Suspense fallback={<ThreeCircleLoader/>}>
            <SurveyUpdate />
        </Suspense>
    </PrivateRoute>
</Switch>
```

Отримаємо дані при відображені маршрутизації на сторінці. Запит повинен повернути користувача та його права на користування інтерфейсом. Таким чином ми виконуємо запити до серверу та синхронізумо стейт програми.

```
const { rawData: userRawData, isLoading: isUserLoading, isError: isUserError } =
useDataApi({
    url: '/user'
});
```

Після виконання запитів користувачу відображується сторінка з опитуваннями в яких він може прийняти участь. Для користувачів типу “admin” сервіс відображає розширений інтерфейс з датами проведення опитувань. Таким чином користувачі з розширеними правами можуть

створювати опитування та отримувати список створених опитувань в сервісі, а також відслідковувати час проведення опитування.

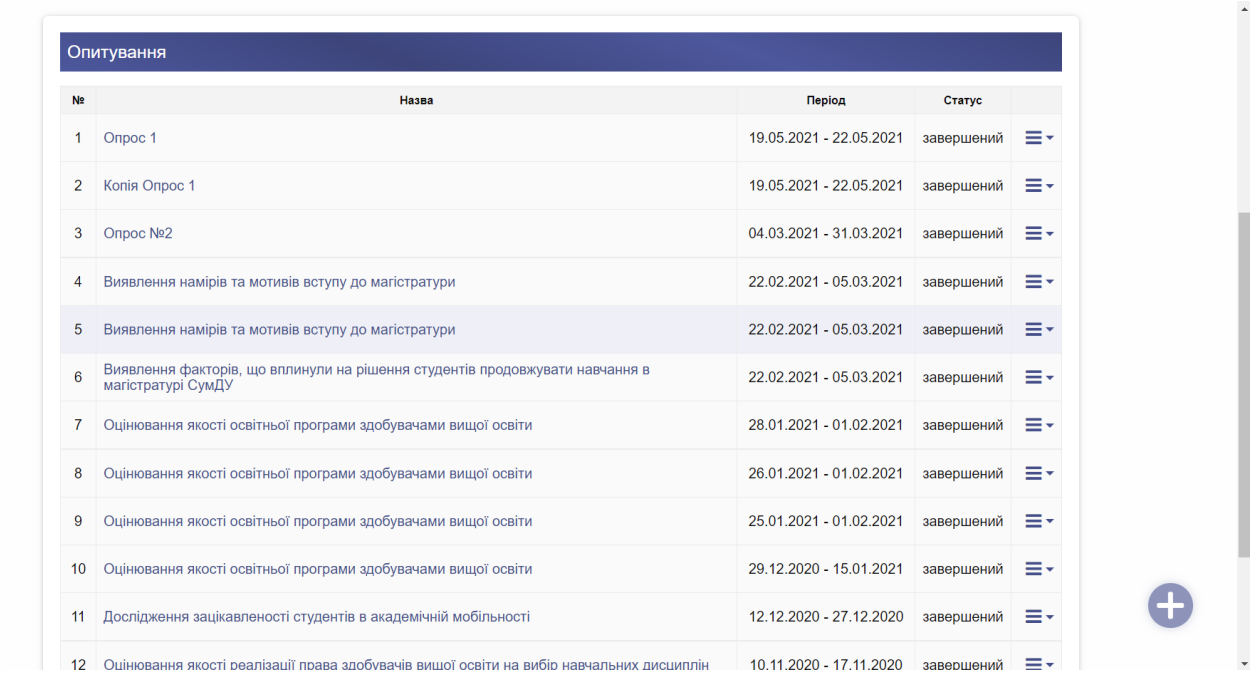

Рисунок 3.5 - Інтерфейс адміністратора сервісу опитувань

Для переходу до одного з опитувань користувач повинен натиснути на тексті(назві), що підсвічений синім кольором. Користувачу інтуітивно буде зрозуміло, що текст є клікабельний, а курсор при наведенні змінить своє відображення.

Для відображення отриманих даних про всі опитування ми використовуємо функцію Javascript під назвою map [14].

```
{surveysState.map((survey, index) => (
        <CSSTransition key={survey.id} classNames="item" timeout={500}>
            <SurveyRow
                key={survey.id}
                survey={survey}
                index={index}
                isAdmin={isAdmin}
                removeSurveyClick={openRemoveReveal}
                copySurveyClick={copySurveyHandler}
            />
        </CSSTransition>
    )
)}
```

Перехід до опитування відбувається після того як користувач натиснув посилання. Відображення посилання потрібно змінити маршрут, за

яким новий компонент відобразить нову логіку та інтерфейс. В проекті для цієї цілі ми використали компонент react router під назвою Link. Цей клас має в собі функціональність посилання html і відображується, як звичайний тег <a>. Але так як це компонент реакт він впливає на логіку фреймворку та змінює його маршрутизацію.

```
<Link to={`/surveys/edit/${survey.id}`} className="tab-switcher">
    <span className="font-awesome va-bottom">&#xf040;</span> <FormattedMessage id="edit"/>
</Link>
```

Для отримання швидкого доступу до потрібного функціоналу адміністраторам в правій частині таблиці відображується dropdown в якому є функціонал редагування опитування, копіювання, видалення та отримання результатів.

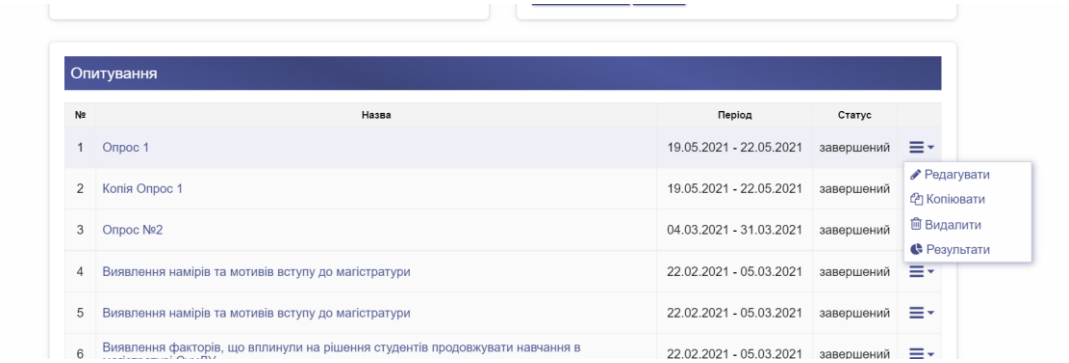

Рисунок 3.6 - Відображення dropdown опитування

Для створення опитування користувач натискає на знак “+” в правому нижньому кутку. Після натискання відображається форма в якій є тема в яку потрібно ввести назву опитування. Адміністратору відображується термін статусу закінчення опитування. Також є можливість відразу встановити статус опитування.

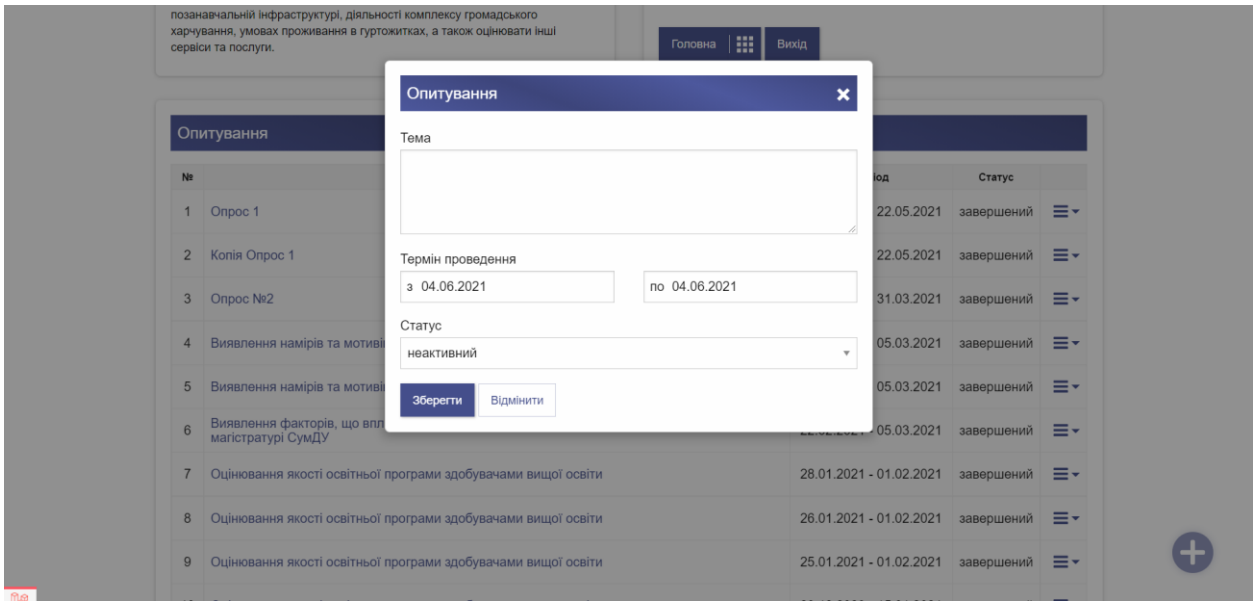

Рисунок 3.7 - Інтерфейс створення опитування

Для відображення в проекті констант наприклад для кнопок та посилань в проекті використовується react-intl версії 4.8.5. Бібліотека дозволяє створити файли перекладів і в залежності від мови сервісу відображати потрібний текст для користувачів. В компонентах можна отримати переклад за допомогою такого звернення до бібліотеки [15].

```
{this.props.intl.formatMessage({id: 'save'})}
```

В файлі в папці i18n знаходяться файли з перекладом на мови які підтримує сервіс. Приклад створення константи в файлі перекладу.

```
"save": "Зберегти",
```

В проекті можливо налаштувати як опитування так і відредагувати або видалити питання. Це досягається за допомогою кнопок в правому верхньому кутку біля кожного питання. Для додавання дій при кліку на кнопку потрібно створити функції обробки даних.

```
<button className="small edit-button primary font-awesome edit" type="button"
onClick={editQuestionClick} style={{right: "25px"}}>&#xf040;</button>
<button className="small edit-button primary font-awesome remove" type="button"
onClick={removeQuestionClick}>&#xf014;</button>
```

Для відкриття нового вікна в програмі потрібно змінити стейт. Досягається передачею функції до компоненту функції та створення виклик її на подію onclick. Функція змінює стейт одного з компонентів та відкриває форму програми.

```
 const removeQuestionClick = () => {
    openRemoveReveal({data: question, type:'question'});
};
const openEditQuestionReveal = (data) => {
    setEditQuestionRevealState( {open: true, data});
};
```

Після відкриття опитування для редагування форма виглядає таким чином.

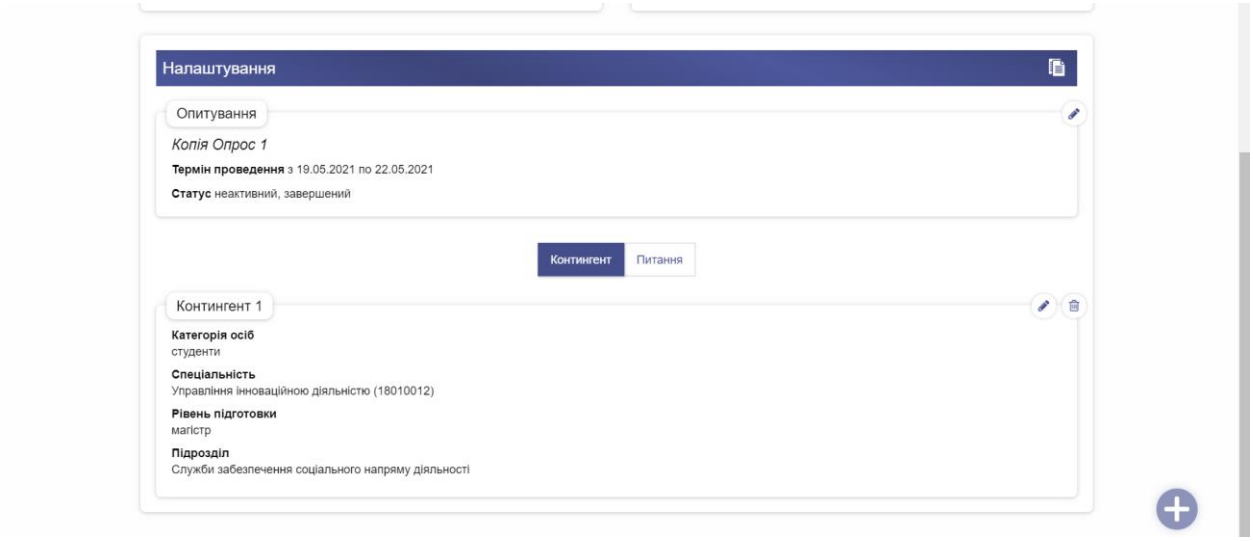

Рисунок 3.8 Інтерфейс налаштування контингенту

Користувач може видали питання не бажаючи того. Для виключення такої ситуації було розроблено форму з підтвердженням результатів.

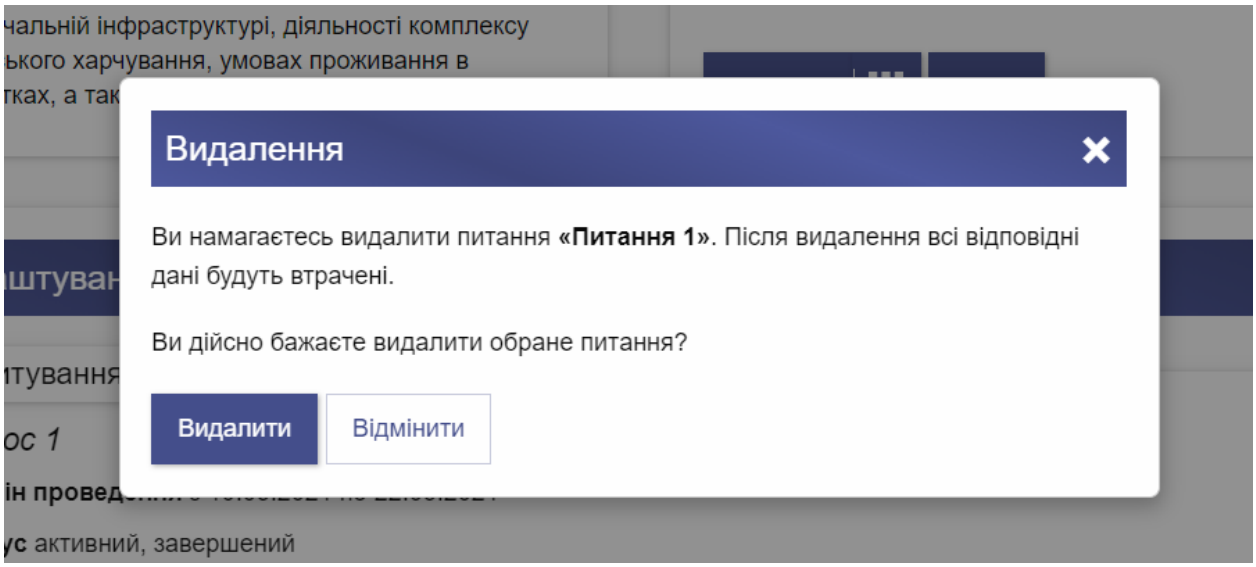

Рисунок 3.9 - При видаленні питання відображується форма з підтвердженням

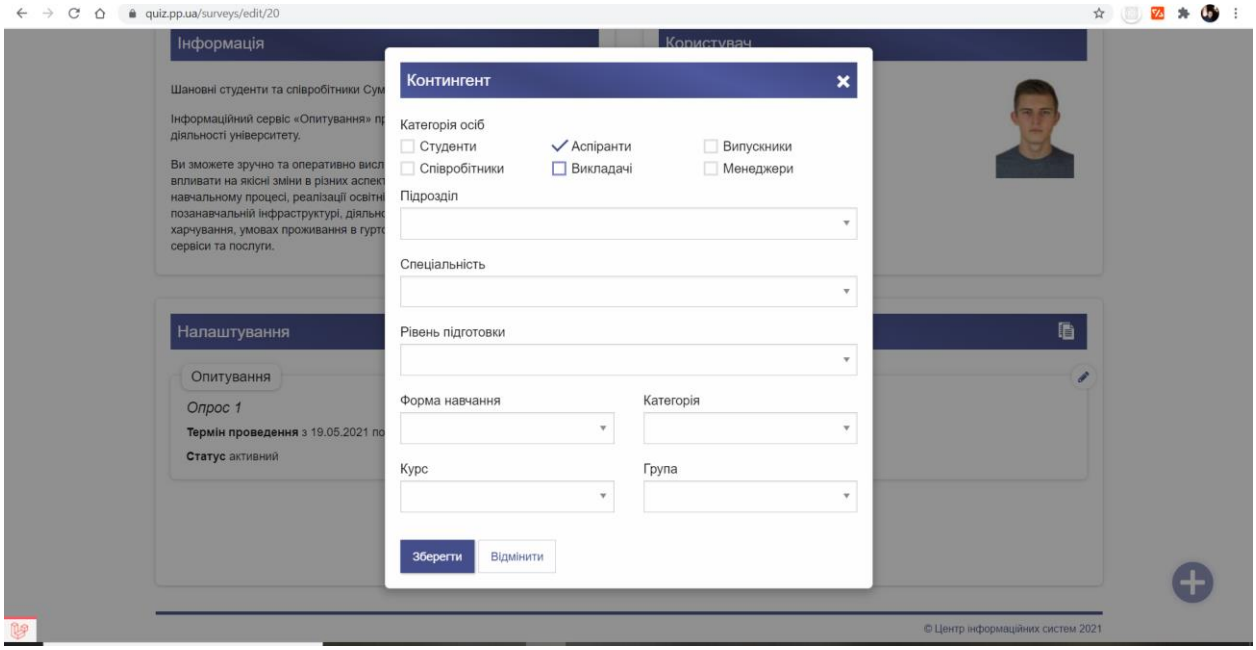

Рисунок 3.10- Інтерфейс форми створення створення контингенту

Для вибору дати вирішено було додати jquery datapicker. Він добре виглядає в дизайні та спрошує користувачу вибір дати.

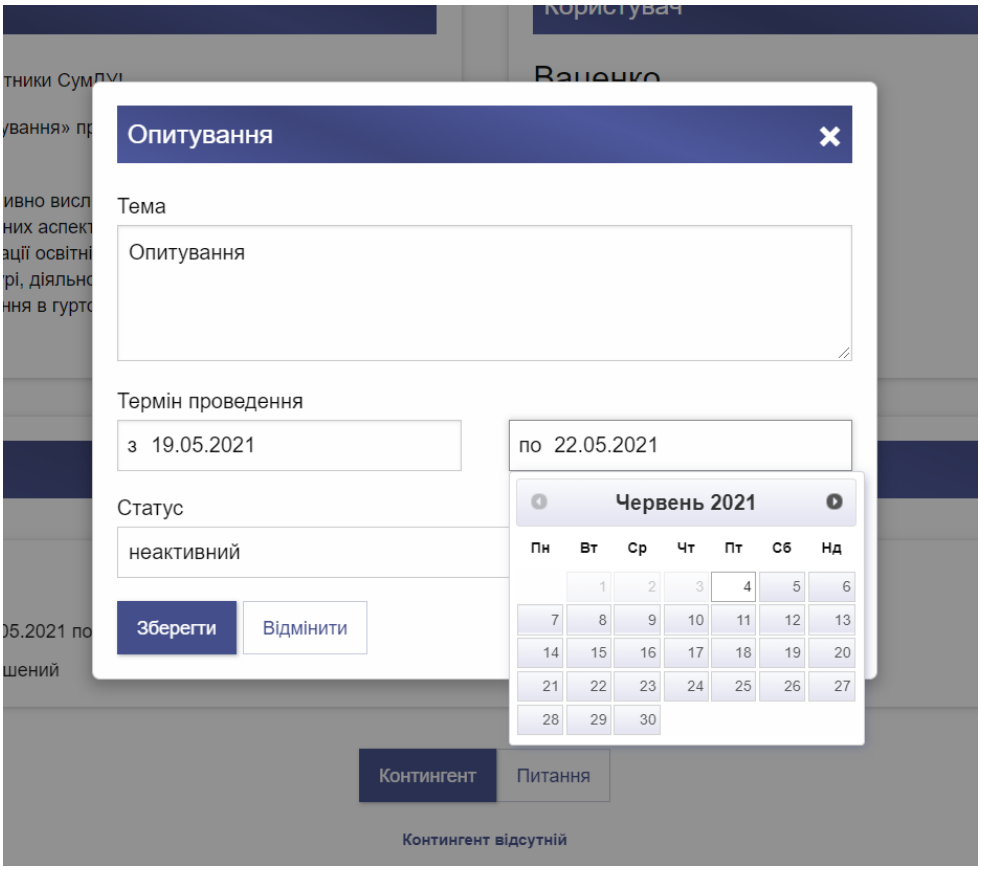

Рисунок 3.11 - Інтетерфейс вибору дати

Для додавання питання до опитування було розроблено форму створення питання. В ній наявне текстове поле для вводу необхідного значення. Також є ввід типу питання та додання нових відповідей.

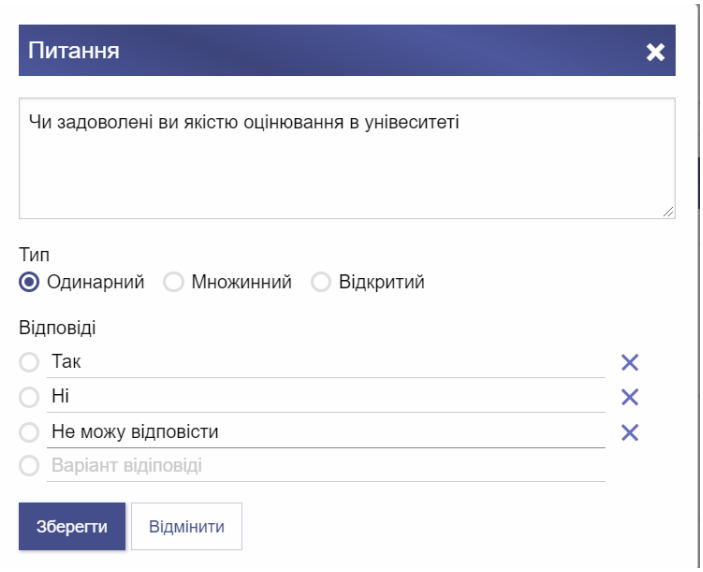

Рисунок 3.12 - Форма додання питання до опитування

Функції додавання питання та видалення приведені нижче. Функція додавання та видалення нижче.

```
addVariant(event) {
    if (event.type === 'keydown' && ( event.which || event.keyCode ) === 9) {
        return true; //tabulation not add new input for variant
    }
    if ( this.state.variants.length < this.maxCountVariants ) {
        this.setState(state => ({
            variants: [...state.variants, {text: '', id: '', id_ques:
this.props.editQuestionState.id, key: -this.state.variants.length * Math.random()}]
        }));
    }
}

removeVariant(event) {
    const target = event.target;
    this.setState(state => ({
        variants: state.variants.filter((variant, index) => +target.dataset.index !== index
)
        })
    );
}
```

У сервісі наявний функціонал збереження результатів опитування за кожним опитуванням. Після натискання на кнопку експорту в правому верхньому куті для клієнту відбувається завантаження файлу з результатами проходження опитування. Також можливо переглянути результати опитувань.

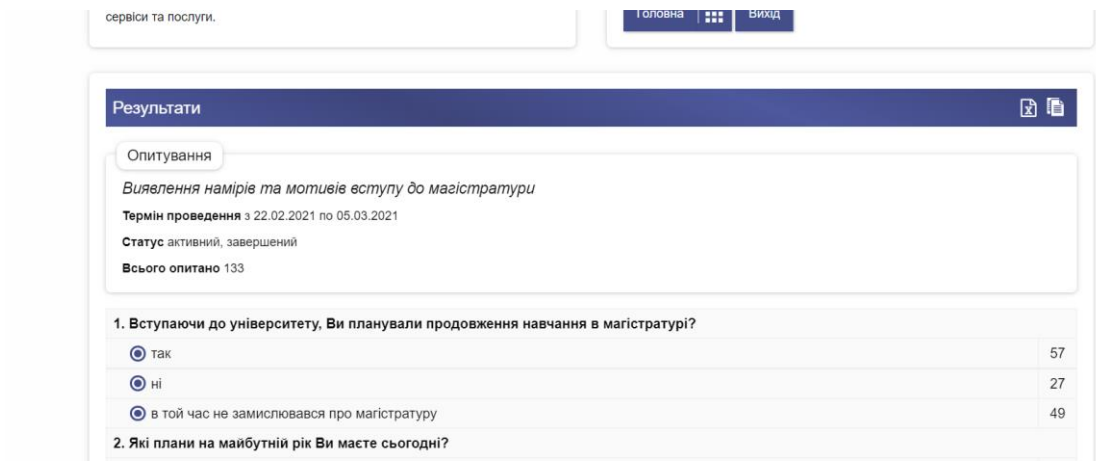

Рисунок 3.13 - Інтерфейс отримання результатів за пройденим опитуванням

У файлі будуть збережені результати проходження опитування категоріями користувачів. На кожне питання створена колонка в якій першим рядком йде назва питання, а іншим відповіді на питання. Також в документі є колонки, які ідентифікують користувача за курсом та групою, що дозволяє аналізувати результати та приймати потрібні дії.

Рисунок 3.14 - Результат експорту даних до CSV файлу

ВИСНОВКИ

Під час виконання випускної роботи було проведено аналіз літератури, методів та інструментів, які забезпечують клієнт-серверну взаємодію, вивчені особливості їх застосування під час створення сервісу опитування.

Після ознайомлення з існуючими рішеннями було розроблена логіка роботи та її реалізація у вигляді сервісу на базі Laravel та React. Дана розробка дозволяє створювати опитування та питання, а іншим користувачам проходити опитування. Додаток був реалізований за допомогою мови програмування Javascript і фремворку React та мови php з фремворком Laravel. Розробка програми відбувались за допомогою середовища розробки PhpStorm і пакетних менеджерів npm та composer .

Сервіс отримав інтуїтивно зрозумілий інтерфейс з анімаціями, що дає змогу адміністраторам створювати опитування, додавати питання відкритих та закритих типів, редагувати та видаляти опитування та питання, налаштовувати контингент осіб, що будуть проходити опитування. Також сервіс дозволяє експортувати результати в вигляді csv файлів та отримувати результати опитувань в реальному часі в веб застосунку. За допомогою таких засобів адміністратори сервісу мають можливість в реальному часі аналізувати результати опитувань та отримувати їх в зручному форматі.

Під розробки сервісу опитування були використанні різноманітні технології та знання, зокрема

- фреймворк React
- мова Javascript;
- мова Php;
- команди Unix-подібних систем;
- веб-технології: JSON-формат, HTML, CSS, Chrome console;
- підтримка програмного забезпечення: використання системи контролю версій, системи автоматичної збірки, уникнення антипатернів;

- фреймворк Laravel;
- пакетні менеджери npm та composer;
- та інші.

Створений сервіс успішно працює і виконує свою функцію дозволяючи автоматизувати опитування. Також дозволяє автоматизувати процес опитування студентів та працівників СумДУ та забезпечує віддалене проведення опитування в зручний користувачам час. Зручний та мінімалістичний інтерфейс не створює проблем користувачам, а широкий функціонал дає можливість адміністраторам сервісу швидко проводити, створювати і редагувати опитування. Більш того результати за пройденими опитуванням можуть бути експортовані до excel файлу, що дозволяє ранній аналіз проблем та гарантує швидке вирішення.

СПИСОК ЛІТЕРАТУРИ

1. Таненбаум Э. Компьютерные сети. 4-е изд. / Э. Таненбаум. – СПб.: Питер, 2007. – 992 с

2. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 4-е изд. / В.Г. Олифер , Н.А. Олифер. – СПб.: Питер, 2010. – 944 с.

3. Бертсекас Д., Галлагер Р. Сети передачи данных / Д. Бертсекас. – М.: Мир, 1989. – 544 с.

4. Кульгин М. Технологии корпоративных сетей. Энциклопедия / М. Кульгин. – СПб: Питер, 2000. – 704 с.

5. Крейг Хант TCP/IP network administration – O'Reilly Media, Inc. – 2020 – 221 с.

6. Дронов В. Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS – БХВ-Петербург, 2016 – 13 – 65 с.

7. Laravel: Up & Running: A Framework for Building Modern PHP Apps 2nd Edition by Matt Stauffer – 2021 – Volume 13. – 181 – 278 с.

8. Jesse Griffin Domain-Driven Laravel. Learn to Implement Domain-Driven Design Using Laravel Springer – Nature Customer Service Center LLC 2020 – 109 – 177 с.

9. Мэтт Стаффер Laravel. Полное руководство – Питер 2021 – 56 – 145с.

10. Kelt Dockins Design Patterns in PHP and Laravel – Springer Nature Customer Service Center LLC 2016 – 105 – 203 с.

11. Хавербеке Марейн Выразительный JavaScript. Современное веб-программирование – Питер 2019 – 104 – 234 с.

12. Learning React: Modern Patterns for Developing React Apps 2nd Edition by Alex Banks, Eve Porcello – 2020 – 345 – 378 с.

13. Флэнаган Дэвид JavaScript. Полное руководство – Диалектика-Вильямс 2021 – 74 – 156 с.

14. Фаулер Мартин Рефакторинг кода на JavaScript. Улучшение проекта существующего кода – Вильямс 2019 – 9 – 204 с.

15. Pro React 16 1st ed. Edition by Adam Freeman – 2020 – Volume 13. – 345 – 378 с.

ДОДАТОК. ВИХІДНИЙ КОД

App.js

```
import React, {useContext, useState} from 'react'
import { BrowserRouter } from 'react-router-dom';
import { IntlProvider } from 'react-intl';

import Layout from '../Layout/Layout';
import i18n_uk  from '../../i18n/uk';
import i18n_uk_datepicker  from '../../i18n/uk_datepicker';
import ErrorBoundary from "../../components/ErrorBoundary/ErrorBoundary";
import StickyMessageSurveyContext from "../../contexts/stickyMessageSurveyContext";
import UserContext from "../../contexts/userContext";
import BigCircleLoader from "../../components/Loader/BigCircleLoader";

const App = () => {
    const messages = { 'uk': i18n_uk };
    $.datepicker.regional['uk'] = i18n_uk_datepicker;

    const language = document.querySelector('html').getAttribute('lang');

    $.datepicker.setDefaults($.datepicker.regional[language]);

    const [ stickySurveyState, setStickySurveyState ] =
useState(useContext(StickyMessageSurveyContext));
    const [ userState, setUserState ] = useState(useContext(UserContext));

    return (
        <IntlProvider locale={language} messages={messages[language]}>
            <ErrorBoundary>
                <BigCircleLoader>
                    <UserContext.Provider value={{userState, setUserState}}>
                        <StickyMessageSurveyContext.Provider value={{stickySurveyState,
setStickySurveyState}}>
                            <BrowserRouter>
                                <Layout />
                            </BrowserRouter>
                        </StickyMessageSurveyContext.Provider>
                    </UserContext.Provider>
                </BigCircleLoader>
            </ErrorBoundary>
        </IntlProvider>
    )
};

export default App;
```

Layout.js

```
import React, { useContext, useEffect, lazy, Suspense } from 'react';
import { Switch, Route } from 'react-router-dom'
import { TransitionGroup, CSSTransition } from "react-transition-group";

import ThreeCircleLoader from "../../components/Loader/ThreeCircleLoader";
import Surveys from "../../components/Survey/Surveys";
import StickyMessage from "../../components/Sticky/StickyMessage";
import useDataApi from "../../utils/hooks/useDataApi";
import userContext from "../../contexts/userContext";
import PrivateRoute from "../../components/PrivateRoute/PrivateRoute";
const Survey = lazy(() => import("../../components/Survey/Survey"));
const SurveyUpdate = lazy(() => import( "../../components/Survey/SurveyUpdate"));
const SurveyResult = lazy(() => import("../../components/Survey/SurveyResult/SurveyResult"));
```

```
const Layout = () => {

    const { rawData: userRawData, isLoading: isUserLoading, isError: isUserError } =
useDataApi({
        url: '/user'
    });

    const { userState, setUserState } = useContext(userContext);
    const user = userRawData && !isUserError ? userRawData : userState;

    useEffect(() => {
        setUserState({...user, isAdmin: user.role === 'admin'});
    }, [isUserLoading]);

    return (
        <>
            <StickyMessage/>
            {isUserLoading ?
                <ThreeCircleLoader/>
                :
                <main className="row">
                    <Route render={({ location }) => (
                        <TransitionGroup>
                            <CSSTransition
                                key={location.pathname}
                                classNames="tab"
                                in={true}
                                mountOnEnter={true}
                                appear={false}
                                enter={false}
                                timeout={300}
                            >
                                <div className="tab">
                                    <Switch location={location}>
                                        <Route exact path="/">
                                            <Surveys/>
                                        </Route>
                                        <Route exact path="/surveys/:id([0-9]+)" >
                                            <Suspense fallback={<ThreeCircleLoader/>}>
                                                <Survey />
                                            </Suspense>
                                        </Route>
                                        <PrivateRoute exact path="/surveys/edit/:id([0-9]+)"
role={"admin"} user={user}>
                                            <Suspense fallback={<ThreeCircleLoader/>}>
                                                <SurveyUpdate />
                                            </Suspense>
                                        </PrivateRoute>
                                        <PrivateRoute exact path="/surveys/result/:id([0-9]+)"
role={"admin"} user={user}>
                                            <Suspense fallback={<ThreeCircleLoader/>}>
                                                <SurveyResult />
                                            </Suspense>
                                        </PrivateRoute>
                                        <Route exact path="" render={() => (window.location =
"/404")}/>
                                    </Switch>
                                </div>
                            </CSSTransition>
                        </TransitionGroup>
                    )}/>
                </main>
            }
        </>
    )
};
```

```
export default Layout;
```

Surveys.js

```
import React, {useContext, useEffect, useState} from 'react';
import {injectIntl, IntlProvider} from "react-intl";
import PropTypes from "prop-types";
import {withRouter} from 'react-router-dom';
import {CSSTransition} from "react-transition-group";

import useDataApi from "../../utils/hooks/useDataApi";
import ThreeCircleLoader from "../Loader/ThreeCircleLoader";
import userContext from "../../contexts/userContext";
//import StickyMessageSurveyContext from "../../contexts/stickyMessageSurveyContext";
import SurveysTable from "./SurveysTable/SurveysTable";
import SurveyMainSettingsForm from "./SurveyEdit/SurveyMainSettings/SurveyMainSettingsForm";

import {surveyDefault} from '../../types/survey';

const initialEditReveal = {
    open: false,
    data: surveyDefault
};

const Surveys = ({ intl, location }) => {
    //const { setStickySurveyState } = useContext(StickyMessageSurveyContext);

    const { rawData: surveysRawData, isLoading: isSurveyLoading, isError: isSurveyError } =
useDataApi({
        url: '/surveys'
    });
    const surveys = surveysRawData && !isSurveyError ? typeof surveysRawData === 'object' ?
Object.values(surveysRawData) : surveysRawData  : [];

    const { userState: {isAdmin}  } = useContext(userContext);


    useEffect(() => {
        if (typeof location.state !== 'undefined') {
            //setStickySurveyState(location.state.setStickySurveyState);
            history.replaceState(undefined, '');
        }
    });

    const [ editRevealState, setEditRevealState ] = useState({...initialEditReveal});

    const openEditReveal = () => {
        setEditRevealState({open: true, data: surveyDefault});
    };

    const closeEditReveal = () => {
        setEditRevealState(initialEditReveal);
    };

    return (
        <>
            {isSurveyLoading && <ThreeCircleLoader />}
            <CSSTransition in={!isSurveyLoading} timeout={300} appear={true} exit={false}
classNames="tab">
                <div className={`clearfix ${isSurveyLoading ? 'tab-exit-done' : ''}`}>
                    <div className="small-12 column">
                        <div className="callout clearfix paddings">
                            <SurveysTable
                                surveys={surveys}
```

```
                                isAdmin={isAdmin}
                            />
                            {isAdmin &&
                                <>
                                    <SurveyMainSettingsForm
                                        {...editRevealState.data}
                                        open={editRevealState.open}
                                        closeEditReveal={closeEditReveal}
                                    />
                                    <button
                                        className="plus-button big add-question primary pulse"
                                        type="button"
                                        title={intl.formatMessage({id:'survey.add'})}
                                        onClick={openEditReveal}
                                    />
                                </>
                            }
                        </div>
                    </div>
                </div>
            </CSSTransition>
        </>
    )
};

Surveys.propTypes = {
    intl: PropTypes.shape({intl: IntlProvider.propTypes}).isRequired
};

export default injectIntl(withRouter(Surveys));
```

SurveysTable.js

```
import React, {useContext, useEffect, useState} from 'react';
import {FormattedMessage, injectIntl, IntlProvider} from 'react-intl';
import {TransitionGroup, CSSTransition} from "react-transition-group";
import PropTypes from "prop-types";

import SurveyRow from "../SurveyRow/SurveyRow";
import StickyMessageSurveyContext from "../../../contexts/stickyMessageSurveyContext";
import errorBoundaryContext from "../../../contexts/errorBoundaryContext";
import foundationReinit from "../../../utils/hooks/foundationReinit";
import fetchData from "../../../utils/helper/fetchData";
import RevealRemove from "../../Reveal/RevealRemove/RevealRemove";
import { survey } from "../../../types/survey";
import BigCircleLoaderContext from "../../../contexts/bigCircleLoaderContext";

const initialRemoveReveal = {
    open: false,
    type: '',
    data: {
        title: ''
    }
};

const SurveysTable = ({intl, surveys, isAdmin}) => {
    const [ surveysState, setSurveysState ] = useState(surveys);
    const { setStickySurveyState } = useContext(StickyMessageSurveyContext);
    const { setBigCircleLoaderState } = useContext(BigCircleLoaderContext);
    const { setErrorState } = useContext(errorBoundaryContext);
    const [ removeRevealState, removeRevealSwitch ] = useState(initialRemoveReveal);

    foundationReinit();

    useEffect(() => {
```

```
        setSurveysState(surveys);
    }, [surveys.length]);

    const openRemoveReveal = ({data}) => {
        removeRevealSwitch({data, open: true, type: 'survey'});
    };

    const closeRemoveReveal = () => {
        removeRevealSwitch(initialRemoveReveal);
    };

    const removeSurveyHandler = async (e) => {
        e.preventDefault();
        try {
            const url = `/surveys/${removeRevealState.data.id}?_method=DELETE`;
            const { data: {message, status, data} } = await fetchData({ url, method:'POST'});
            if (status === 'success' && data === true) {
                setSurveysState(surveysState.filter(survey => survey.id !==
removeRevealState.data.id));
                setStickySurveyState({type:'primary', message});
            } else {
                setStickySurveyState({type:'alert', message: intl.formatMessage({id:
'survey.error.remove'})});
            }
        } catch (error) {
            const errorState = {type: 'alert', message: intl.formatMessage({id:
'errors.ajax'})};
            if (error.response) {
                if ([401, 404].includes(error.response.status)) {
                    setErrorState({hasError: true, error: error.response});
                } else {
                    if (error.response) {
                        errorState.message = error.response.data.message;
                    }
                }
            }
            setStickySurveyState(errorState);
        } finally {
            closeRemoveReveal();
        }
    };

    const copySurveyHandler = async id => {
        try {
            setBigCircleLoaderState(true);
            const { data: {message, data} } = await fetchData({ url: `/surveys/copy/${id}`,
method: 'POST'});
            setStickySurveyState({type:'primary', message});
            setSurveysState([...surveysState, data]);
        } catch (error) {
            const errorState = {type: 'alert', message: intl.formatMessage({id:
'errors.ajax'})};
            if (error.response) {
                if ([401, 404].includes(error.response.status)) {
                    setErrorState({hasError: true, error: error.response});
                } else {
                    if (error.response) {
                        errorState.message = error.response.data.message;
                    }
                }
            }
            setStickySurveyState(errorState);
        } finally {
            setBigCircleLoaderState(false);
        }
    };
```

```
    return (
        <>
            <h5 className={`callout cabinet small clearfix one-icon ${surveysState.length ? ''
: 'margin-none'}`}>
                <FormattedMessage id={`surveys.title${surveysState.length ? '' : '.empty'}`}/>
            </h5>
            <div className="table-survey">
                {!!surveysState.length &&
                <div className="table-scroll">
                        <table className="hover custom size-1em m-b-n l-h-1-1 p05">
                            <thead>
                            <tr>
                                <th className="text-center"><FormattedMessage
id="surveys.table.number.symbol"/></th>
                                <th className="text-center"><FormattedMessage
id="surveys.table.title"/></th>
                                <th className="text-center"><FormattedMessage
id="surveys.table.period"/></th>
                                <th className="text-center"><FormattedMessage
id="surveys.table.status"/></th>
                                {isAdmin && <th className="text-center"/>}
                            </tr>
                            </thead>
                            <TransitionGroup component="tbody">
                                {surveysState.map((survey, index) => (
                                        <CSSTransition key={survey.id} classNames="item"
timeout={500}>
                                            <SurveyRow
                                                key={survey.id}
                                                survey={survey}
                                                index={index}
                                                isAdmin={isAdmin}
                                                removeSurveyClick={openRemoveReveal}
                                                copySurveyClick={copySurveyHandler}
                                            />
                                        </CSSTransition>
                                    )
                                )}
                            </TransitionGroup>
                        </table>
                    </div>
                }
            </div>
            {isAdmin &&
                <RevealRemove
                    open={removeRevealState.open}
                    type={removeRevealState.type}
                    title={removeRevealState.data.title}
                    onClose={closeRemoveReveal}
                    onRemove={removeSurveyHandler}
                />
            }
        </>
    );
};

SurveysTable.propTypes = {
    intl: PropTypes.shape({intl: IntlProvider.propTypes}).isRequired,
    surveys: PropTypes.arrayOf(survey).isRequired,
    isAdmin: PropTypes.bool.isRequired
};

export default injectIntl(SurveysTable);
```

SurveyRow.js

```
import React from 'react';
import { Link } from 'react-router-dom';
import {FormattedDate, FormattedMessage} from 'react-intl';
import PropTypes from "prop-types";
import {injectIntl, IntlProvider} from "react-intl";


import { survey } from "../../../types/survey";

const SurveyRow = ({intl, survey, index, isAdmin, removeSurveyClick, copySurveyClick}) => {

    const removeSurvey = e => {
        e.preventDefault();
        removeSurveyClick({data: survey});
    };

    const copySurvey = e => {
        e.preventDefault();
        copySurveyClick(survey.id);
    };

    const from = new Date(survey.from);
    const to = new Date(survey.to);

    let className = survey.status === 1 ? '' : ' success';
    let status = survey.status_name;

    if ( survey.answers_exist ) {
        className = '';
        status = intl.formatMessage({id: "survey.status.3"});
    }
    if ( to < new Date().setHours(0,0,0, 0) ) {
        className = '';
        status = intl.formatMessage({id: "survey.status.4"})
    }


    return (
        <tr id={`survey-${survey.id}`}>
            <td className="text-center">{index+1}</td>
            <td data-period={`${intl.formatDate(from)} - ${intl.formatDate(to)}`}>
                <Link to={`/surveys/${survey.id}`} className="underline">{survey.title}</Link>
            </td>
            <td className="text-center">
                <FormattedDate value={from}/> - <FormattedDate value={to}/></td>
            <td className={`text-center${className}`}>{status}</td>
            {isAdmin &&
                <td className="text-center">
                    <ul className="dropdown menu" data-dropdown-menu="true" data-disable-
hover="true" data-click-open="true"
                        data-force-follow="false" data-alignment="left"
                        style={{verticalAlign: 'text-bottom', maxWidth: '40px',
float:'right'}}>
                        <li>
                            <a className="font-awesome size-21"
onClick={(e)=>{e.preventDefault()}} >&#xf0c9;</a>
                            <ul className="menu cabinet text-left" style={{minWidth:'165px'}}>
                                <li>
                                    <Link to={`/surveys/edit/${survey.id}`} className="tab-
switcher">
                                        <span className="font-awesome va-
bottom">&#xf040;</span> <FormattedMessage id="edit"/>
                                    </Link>
                                </li>
```

```
                                <li>
                                    <a className="tab-switcher" href="#" onClick={copySurvey}>
                                        <span className="font-awesome va-
bottom">&#xf0c5;</span> <FormattedMessage id="copy"/>
                                    </a>
                                </li>
                                <li>
                                    <a className="tab-switcher" href="#"
onClick={removeSurvey}>
                                        <span className="font-awesome va-bottom size-
18">&#xf014;</span> <FormattedMessage id="remove"/>
                                    </a>
                                </li>
                                <li>
                                    <Link to={`/surveys/result/${survey.id}`} className="tab-
switcher">
                                        <span className="font-awesome va-
bottom">&#xf200;</span> <FormattedMessage id="result.result"/>
                                    </Link>
                                </li>

                            </ul>
                        </li>
                    </ul>
                </td>
            }
        </tr>
    );
};

SurveyRow.propTypes = {
    intl: PropTypes.shape({intl: IntlProvider.propTypes}),
    survey: survey.isRequired,
    index: PropTypes.number.isRequired,
    isAdmin: PropTypes.bool.isRequired,
    removeSurveyClick: PropTypes.func,
    copySurveyClick: PropTypes.func
};


export default injectIntl(SurveyRow);
```

IndexController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Survey;
use App\Repositories\CabinetRepository;
use App\Services\SurveyService;
use Carbon\Carbon;
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Cache;
use Illuminate\View\View;


use App\Exceptions\CabinetAuthKeyException;
use App\Exceptions\CabinetResponseException;

class IndexController extends Controller
{

    /**
```

```
     * Cabinet authentication, cabinet api response and main page.
     *
     * @param Request $request
     * @param Guard $cabinetGuard
     * @param SurveyService $surveyService
     * @return View
     */
    public function index(Request $request, Guard $cabinetGuard, SurveyService $surveyService)
    {
        $mode = $request->query('mode', 0);

        $modes = config('cabinet.modes');
        if ( !in_array($mode, array_filter($modes, fn($value) => $value !== 4)) ) {
            try {
                $cabinetGuard->validate(['key' => $request->query('key', []), 'sid' =>
session()->getId()]);
            } catch (CabinetResponseException | CabinetAuthKeyException $e) {
                if ( $mode == 4 ) return icon_service_response();
                return redirect()->away(config('cabinet.url').'?error='.cabinet_message($e-
>getMessage()));
            }
        }

        switch($mode) {
            case $modes[3]:
                return response(trans('labels.head.description_shot'),200)->header('Content-
type', 'text/plain');
            case $modes[4]:
                return icon_service_response($surveyService->countAll());
            case $modes[100]:
                return response('', 200)->header('X-Cabinet-Support',25);
            case $modes[101]:
                $sid = $request->query('sid');
                if ( $sid ) {
                    session()->setId($sid);
                    session()->start();
                    session()->invalidate();
                    exit();
                }
        }

        return view('layout');
    }

    public function photo(CabinetRepository $cabinetRepository)
    {
        $response = $cabinetRepository->getPersonPhoto(session('key'));

        if ( $response->header('Content-Type') !== 'application/json' ) {
            return response($response->body())->withHeaders([
                'Content-type' => $response->header('Content-Type'),
                'Content-length' => $response->header('Content-Length')
            ]);
        }
    }

    public function logout(CabinetRepository $cabinetRepository)
    {
        $key = session('key');
        session()->invalidate();
        $response = $cabinetRepository->logout($key);

        if ( isset($response['status']) && $response['status'] === 'OK' ) {
            return redirect()->away(config('cabinet.url'));
        }  else {
            return redirect()-
>away(config('cabinet.url').'?error='.cabinet_message(config('messages.cabinet.error_logout'))
```

```
);
        }
    }

    public function cacheClear($key, $tags) {
        if ( $key === config('cache.key') ) {
            $count = Survey::query()->
                where('updated_at', '>=', Carbon::now()->subHours(24)->toDateTimeString())->
                count();
            if ( $count > 0 ) return Cache::store('apc')->tags(explode(',', $tags))->clear();
            return false;
        }
    }
}
```

SurveyController.php

```
<?php

namespace App\Http\Controllers\Api;

use App\Exceptions\NoCredentialsException;
use App\Exports\SurveyResultExport;
use App\Http\Requests\SurveyRequest;
use App\Http\Requests\SurveyStoreRequest;
use App\Models\Survey;
use App\Services\SurveyService;
use App\Http\Controllers\Controller;
use Illuminate\Http\Response;
use Maatwebsite\Excel\Facades\Excel;
use Symfony\Component\HttpFoundation\BinaryFileResponse;

class SurveyController extends Controller
{
    private SurveyService $service;

    public function __construct(SurveyService $surveyService)
    {
        $this->service = $surveyService;
    }

    /**
     * Display a listing of the survey.
     *
     * @return Response
     */
    public function index()
    {
        return response()->success($this->service->all());
    }

    /**
     * Display the specified survey.
     *
     * @param int $id
     * @param SurveyRequest $request
     * @return Response
     * @throws NoCredentialsException
     */
    public function show($id, SurveyRequest $request)
    {
        return response()->success($this->service->getById($id, $request));
    }

    /**
```

```
     * Store a newly created survey in storage.
     *
     * @param SurveyStoreRequest $request
     * @return Response
     */
    public function store(SurveyStoreRequest $request)
    {
        return response()->success($this->service->create($request), null, 201);
    }


    /**
     * Update the survey in storage.
     *
     * @param int $id
     * @param SurveyStoreRequest $request
     * @return Response
     */
    public function update($id, SurveyStoreRequest $request)
    {
        return response()->success($this->service->update($id, $request));
    }


    /**
     * Remove the survey from storage.
     *
     * @param  int $id
     * @return Response
     */
    public function destroy($id)
    {
        $result = Survey::findOrFail($id)->delete();
        return response()->success($result, trans('messages.survey.success_remove'));
    }

    /**
     * @param $id
     * @return BinaryFileResponse
     */
    public function export($id) {
        return Excel::download(new SurveyResultExport($id), 'result.xlsx',
\Maatwebsite\Excel\Excel::XLSX);
    }

    public function copy($id) {
        return response()->success($this->service->copy($id),
trans('messages.survey.copied'));
    }
}


<?php

namespace App\Http\Controllers\Api;

use App\Http\Requests\AnswersRequest;
use App\Services\AnswerService;
use App\Http\Controllers\Controller;


class AnswerController extends Controller
{
    protected $answerService;

    public function __construct(AnswerService $answerService)
    {
```

```
        $this->answerService = $answerService;
    }

    public function store(AnswersRequest $request, $idSurvey)
    {
        $this->answerService->insertAnswers($request->input('questions'), $idSurvey);
        return response()->success(null, trans('messages.surveys.success'), 201);
    }
}
```

CabinetGuard.php

```
<?php


namespace App\Services\Auth;


use App\Exceptions\CabinetAuthKeyException;
use App\Models\User;
use Illuminate\Contracts\Auth\Authenticatable;
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Contracts\Auth\UserProvider;
use Illuminate\Http\Request;

class CabinetGuard implements Guard
{
    private $request;
    private $provider;
    private $user;

    public function __construct(UserProvider $provider, Request $request)
    {
        $this->request = $request;
        $this->provider = $provider;
        $this->user = NULL;
    }

    /**
     * Determine if the current user is authenticated.
     *
     * @return bool
     */
    public function check()
    {
        return !is_null($this->user());
    }

    /**
     * Determine if the current user is a guest.
     *
     * @return bool
     */
    public function guest()
    {
        return !$this->check();
    }

    /**
     * Get the currently authenticated user.
     *
     * @return \Illuminate\Contracts\Auth\Authenticatable|null
     */
    public function user()
    {
```

```
        if (!is_null($this->user) ) {
            return $this->user;
        }

        $user = session('user', null);
        if ( isset($user->person['guid']) ) {
            $this->setUser($user);
            return $this->user;
        }
        return null;
    }

    /**
     * Get the ID for the currently authenticated user.
     *
     * @return int|string|null
     */
    public function id()
    {
        if ($user = $this->user()) {
            return $this->user()->getAuthIdentifier();
        }
    }

    /**
     * Validate a user's credentials.
     *
     * @param array $credentials
     * @return bool
     * @throws CabinetAuthKeyException
     */
    public function validate(array $credentials = [])
    {
        $key = $credentials['key'];
        if ( $key ) {
            if ( !($session_key = session('key')) || $session_key != $key ) {
                session()->invalidate();
                $credentials['sid'] = session()->getId();
                $person = $this->provider->retrieveByCredentials($credentials);
                session(['key' => $key]);
                session(['person' => $person]);
                $user = User::firstOrCreate(['guid' => $person['guid']]);
                $user->person = $person;
                session(['user' => $user]);
            }
        } else if ( !session('key') ) {
            throw new CabinetAuthKeyException( trans('auth.key_absent') );
        }

        $user = session('user');
        $this->setUser($user);
        return true;
    }

    /**
     * Set the current user.
     *
     * @param \Illuminate\Contracts\Auth\Authenticatable $user
     * @return void
     */
    public function setUser(Authenticatable $user)
    {
        $this->user = $user;
    }
}
```