МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

ПРОЕКТУВАННЯ ТА РОЗРОБЛЕННЯ СЕРВІСНОЇ АРХІТЕКТУРИ УПРАВЛІННЯ БІЗНЕС-ПРОЦЕСАМИ УНІВЕРСИТЕТУ. МОДУЛЬ ІСТОРІЯ ЗМІН

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти «бакалавр»

Виконав: студент 4 курсу, 441 групи
Спеціальності: 121 Інженерія програмного забезпечення
Освітньо-професійної програми: Інженерія програмного забезпечення першого (бакалаврського) рівня освіти
Букацель Дмитро Олександрович
Керівник: професор, кандидат фізико-математичних наук, доктор педагогічних наук Співаковський О. В.
Рецензент: Кузьмич В.І., професор кафедри алгебри, геометрії та математичного аналізу, ХДУ

Івано-Франківськ – 2022

ЗМІСТ

ВСТУП

Актуальність теми: При розробці будь-якого веб-додатку слід приділяти увагу логуванню історії змін, для того щоб мати чітке представлення що саме було змінено, для їх швидкого відстеження у разі виникнення помилки. Компонент історії змін у веб-додатку являє собою програмний модуль, що дозволяє відстежувати зміни, що вносяться у базу даних, та надає інформацію про останні оновлення користувачам. Можна навести приклади переваг використання компонента історії змін:

1. Історія змін дає змогу відстежити внесені зміни до бази даних. Це може допомогти в ідентифікації та вирішенні проблем, що виникають на сайті.
2. Історія змін дозволяє користувачам отримувати інформацію про останні оновлення, які, в свою чергу, допомогають користувачам залишатися в курсі останніх новин.
3. Інформація про останні зміни у базі даних може покращити якість роботи користувачів на сайті, так як оновлення можна буде дивитися безпосередньо у веб-додатку, без необхідності контактування з розробниками, аби дізнатися потрібну інформацію.
4. Якщо дані були випадково видалені або змінені, історія змін дозволяє повернути їх до попереднього стану.

Таким чином, компонент історії змін для веб-додатку є корисним інструментом, який може покращити процес відстеження останніх оновлень у разі виникнення проблем та допомогти користувачам залишатися в курсі останніх змін у базі даних. Використання компонента історії змін дозволяє покращити керування веб-додатком.

Об’єкт дослідження: Компонент історії змін у веб-додатку ХДУ24.

Предмет дослідження: розроблення модуля історії змін для веб-додатку ХДУ24, який дозволяє відображати інформацію про зміни, що відбуваються в системі, а також надаває користувачеві можливість перегляду історії змін.

Мета дослідження: проаналізувати технології, які використовуються для розробки компоненту історії змін даних, розглянути аналоги. На основі отриманої інформації реалізувати компонент логування у проекті ХДУ24, щоб надати користувачам можливість відстежувати зміни, внесені в дані, і зберігати попередні версії. Це дозволяє відновлювати дані у разі їх пошкодження або непердбачуваної зміни, відстежувати всі зміни в програмі. Загалом, мета створення компонента історії змін полягає в тому, щоб покращити якість та надійність веб-додатку, забезпечуючи можливість відстежування та відновлювання даних у разі потреби.

Завдання дослідження:

- Проаналізувати різні методи логування та історії дій користувача, які можуть використовуватись у веб-додатку
- Провести аналіз готових рішень систем з компонентом історії змін
- Розробити технічне завдання для компонента історії змін, визначивши функціональні та нефункціональні вимоги
- Огляд ендпоінтів API, які відповідають за логування
- Реалізувати компонент історії змін у веб-додатку, забезпечивши його роботу згідно з технічним завданням

РОЗДІЛ 1.
ТЕОРЕТИЧНИЙ АНАЛІЗ КОМПОНЕНТУ ЛОГУВАННЯ

1.1. Логування як важлива частина будь-якого додатку.

Лог (англ. – log) – це місце(файл), куди автоматично записується важлива інформація про роботу системи чи програми. Їх записує програмне забезпечення, яке управляє внутрішньою, або бекенд частиною сайту чи онлайн-системи. Лог – своєрідний журнал подій.[1]

У лог дій записуються відомості про помилки, дії користувачів та інші події, які відбуваються на сервері або в системі. Розробники користуються ними під час налагодження або перевірки, як працює програмне забезпечення.

Майже кожна сутність у системі ХДУ24 має свою історію змін, за допомогою якої можно відстежити усі зміни, які були виконані, і у разі потреби, швидко відновити попереднє значення.

Логування – це процес формування логів, структурування та переміщення їх в окремі файли для швидкого доступу у разі потреби. Правильна організація зберігання та виведення логів є надзвичайно важливою, оскільки це забезпечує швидкий та своєчасний доступ до них. Журнал подій містить інформацію про все, що відбулося з системою, як вона реагувала на події, які дії виконували конкретні користувачі, коли це відбувалося, та інше. Тому логування вважається ефективним способом моніторингу роботи певного ресурсу.[2]

Логування також корисне для виявлення типових помилок користувачів, а також для цілей безпеки. Створення гарної системи логування про діяльність користувача може попередити нас про зловмисну діяльність або випадкові помилки.

Логи також має багато типів, найпоширенішеми є:

1) Лог подій — це високорівневий журнал, який записує дані про діяльність системи, щоб забезпечити контрольний журнал

для виявлення та вирішення проблем. Журнали подій мають важливе значення для розуміння поведінки складних систем, особливо у випадку програм із невеликою кількістю взаємодії з користувачем.[3]

2) Системний журнал записує події операційної системи, такі як системні зміни, повідомлення про запуск, помилки, виявлення вірусів, попередження та несподівані завершення роботи.
3) Журнал доступу записує список усіх запитів на окремі файли, які люди або програми запитують із системи. Він містить інформацію про аутентифікацію користувача, хто запитував певний системний файл, коли він запитував його та іншу пов’язану інформацію.[3]
4) Лог сервера — це файл журналу, який сервер автоматично створює та підтримує його. Він містить список дій, які виконує сервер, наприклад, кількість запитів сторінок, IP-адреси клієнтів, типи запитів тощо.
5) Лог історії змін — це дані, які містять хронологічний запис змін, внесених у програмне забезпечення, або змін, які були внесені до бази даних. Наприклад, ці дані можуть відображати усі дії, які були виконані над якоюсь сутністю у базі даних, зміна, додаваня полів тощо.

Важливо, щоб логи могли надати точний контекст про те, що робив користувач, коли сталася певна помилка. Таким чином, наявність принаймні одного запису журналу для кожного запиту/результату на кожному рівні програми є обов’язковою.[4]

Нижче наведено декілька прикладів того, які є переваги використовувати логування.

Виявлення та усунення помилок: журнали подій є важливими для моніторингу програм і серверів і можуть запускати автоматичні

сповіщення, коли виникають помилки та неочікувані події. Дані журналу можуть бути корисними для точного визначення першопричини проблем, які виникають у програмі. Ці дані дозволяють розробникам відстежувати виклики методів, які призвели до проблеми, і визначати точний рядок коду, який викликав проблемну ситуацію, полегшуючи дослідження проблеми та відтворення проблемного сценарію. Розробники можуть використовувати інформацію, надану в файлах логу, для швидкого виявлення виникаючих проблем задля налагодження та вдосконалення системи.

Оптимізація роботи: оскільки програми та системи стають складнішими, керування ними стає все складнішим. Системні журнали містять цінну інформацію, яка може допомогти IT-командам визначити тенденції та оптимізувати роботу додатку. Ефективний аналіз логів є критично важливим компонентом ефективної роботи. Отже, служачи цінним ресурсом для визначення першопричини, дані журналу допомагають мінімізувати інший ключовий показник реагування на проблеми, що виникають: середній час до вирішення (MTTR).[6] Подібним чином зменшення MTTR додатково допомагає обмежити вплив проблем на кінцевих користувачів.

Покращення розуміння як користувачі використовують програму: хоча дані журналу є цінним активом для вирішення проблем у програмах, вони також можуть бути корисними з інших причин. Наприклад, розробники можуть використовувати дані журналу, щоб отримати повне розуміння того, як користувачі використовують програму. У веб-додатку розробники можуть аналізувати журнали запитів, щоб виявити ключові тенденції та закономірності, важливі для організації, наприклад, визначити час, коли веб-додаток має більший трафік. Крім того, цей аналіз може виявити інформацію про те, до якого вмісту ваші користувачі

мають доступ найчастіше та які браузери вони найчастіше використовують для доступу до нього.

Посилення безпеки: аналіз даних журналу аудиту може допомогти фахівцям з кібербезпеки швидко реагувати на незвичайні події програми та зменшити ризик неавторизованого доступу. Співвідносячи системні чи мережеві події з діяльністю користувача, можливо налаштувати сповіщення про незвичайну активність і забезпечити безпеку ресурсу. Наприклад, кілька невдалих спроб входу можуть викликати попередження. Журнал аудиту дозволяє оцінити дії користувача в програмі в контексті безпеки. Ці журнали зазвичай записують дії входу та виходу, а також відомості про те, коли та як користувач маніпулює даними в системі. Ця інформація дає організаціям значну перевагу, оскільки вони тепер мають механізм ідентифікації, відстеження та, у разі потреби, скасування неавторизованих змін даних. Іншими словами, організації можуть використовувати цей механізм для підвищення безпеки та обмеження збитків у разі виникнення інцидентів.[6]

Розшифровка логів має свої особливості і варто дотримуватися рекомендацій розробника під час аналізу логів. Програмісти використовують різні механізми запису логів:

1) Найбільш зрозумілий та поширений механізм - запис логів до текстового файлу. Кожна подія записується окремим рядком. Цей метод легко реалізувати та його можна прочитати будь-яким текстовим редактором.
2) Поглиблений лог - кожна подія записується декількома рядками, тобто одна подія розбивається на кілька менших. Читання таких логів потребує спеціальних програм.
3) Бінарний лог - це найскладніший тип файлів. Бінарний журнал використовується для реплікації даних між кількома серверами баз даних[5], резервного копіювання даних і

відновлення даних після збоїв. Також він може використовуватися для аналізу продуктивності та виявлення помилок у додатках.

4) Програми, що використовують БД або самі СУБД (система управління базами даних). Запис логів в БД може сповільнити їхню роботу через інтенсивний запис логів.

1.2. Методи логування історії дій користувача.

Методи логування історії дій - це процес реєстрації та збереження інформації про дії, що здійснюються в системі чи додатку. Це може допомогти при аналізі та налагодженні програмного забезпечення, а також при пошуку та усуненні помилок. Існує безліч методів логування історії дій користувачів. Розглянемо деякі з них:

1) Журнали аудиту - це програмне забезпечення, яке записує дії користувачів в системі, такі як вхід до системи, зміна файлів, створення нових користувачів і т.д. Лог аудиту фіксує виникнення події, час, коли вона сталася, користувача чи службу та сутність, на який вона впливає. Переглядаючи журнали аудиту, системні адміністратори можуть відстежувати дії користувачів, а групи безпеки можуть розслідувати порушення та забезпечувати дотримання нормативних вимог.[7]
2) Логування у файли – це поширений метод реєстрації активності користувача. Програма або система може записувати дії користувача у файл журналу, який можна переглянути пізніше. Найчастіше активність логується у файли формату JSON, XML, TXT.
3) Журнали баз даних – деякі системи можуть реєструвати дії користувача в базі даних, що полегшує запити й аналіз даних. Логування до бази даних є важливою частиною

розробки високодоступного рішення, оскільки журнали бази даних дають змогу відновлюватися після збою, а також синхронізувати первинну та вторинну бази даних. Ці журнали зберігають записи про зміни бази даних. Якщо базу даних потрібно відновити до точки після останньої повної резервної копії в автономному режимі, логи необхідні для того щоб відновити дані до помилки.[8]

4) Журнали подій операційної системи - операційна система може зберігати записи про дії користувачів, наприклад встановлення програм, видалення файлів тощо.
5) Логи моніторингу мережі – системи моніторингу мережі можуть записувати, які сайти користувачі відвідують, які дані передають тощо. Інструменти моніторингу мережі можуть фіксувати трафік у мережі та реєструвати дії користувачів.
6) Моніторинг активності співробітників – існують спеціальні програми, що відстежують активність користувачів на комп'ютерах в офісі, або локально на домашньому комп’ютері. Наприклад, ці програмі фіксують скріншоти екрану, запис клавіатури, відстеження миші тощо.

Конкретний метод логування вибирається в залежності від конкретних потреб і вимог до системи. Вибір потрібного методу також може залежити від вартості зберігання даних і складності аналізу логів. Деякі методи, такі як логування в базу даних, можуть вимагати більшого обсягу ресурсів, але можуть забезпечити більш простий доступ до даних та більш точний аналіз. Інші методи, такі як запис логів до окремих файлів, можуть бути простіші у реалізації, але можуть вимагати більших зусиль для обробки та аналізу даних.

У цілому вибір методу логування повинен базуватися на конкретних потребах системи, включаючи її ціль, розмір і складність, а також обмеженості ресурсів і вимог до безпеки.

1.3. Аналіз готових рішень у системах з модулем історії змін.

У багатьох сучасних системах важливо використовувати компонент історії змін (або журнал аудиту) в інформаційних системах – це модуль, який записує всі події та дії, що відбуваються в системі, з метою забезпечення безпеки, контролю та аналізу виконаних змін. Компонент історії змін відстежує всі операції, які здійснюються в системі, включаючи створення, зміну та видалення даних.

Компонент історії дій представлений у різних форматах, наприклад у вигляді системних журналів, лог-файлів або баз даних. Він потрібен у багатьох індустріях, де потрібні високі стандарти безпеки, таких як комп'ютерні мережі, банківська справа, державна та приватна сфери.

Зазвичай у багатьох системах доступ до історії змін є тільки у адміністраторів, та розглянути їх звичайні користувачі не мають можливості. Але також зустрічаються ці компоненти і з доступом для звичайних користувачів. Наприклад, можна виділити програму **Discord**.

Discord — це додаток для голосового, відео- та текстового чату, який десятки мільйонів людей використовують для спілкування з друзями та спільнотами. Люди щодня використовують Discord, щоб поговорити про багато речей, починаючи від мистецьких проектів і сімейних поїздок до домашніх завдань і підтримки психічного здоров’я. Це дім для спільнот будь-якого розміру, але найбільш широко ним користуються невеликі та активні групи людей, які регулярно спілкуються. [11]

У програмі Discord є спільні групи, які можуть налічувати сотні тисяч користувачів, так звані сервери. У кожного серверу є журнал аудиту, до якого мають доступ лише адміністрація цього серверу.

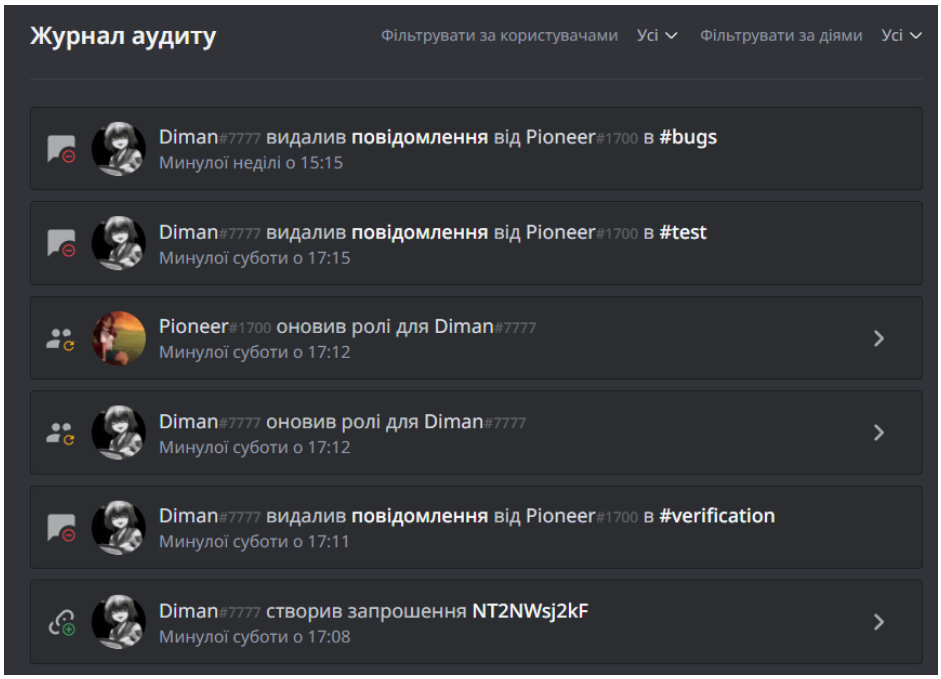


Рис. 1.3.1

На скріншоті вище(рис. 1.3.1) зображено приклад того як виглядає журнал аудиту у Discord.

Переваги журналу аудиту у Discord:

1. Надає повну інформацію про дії на сервері, що дозволяє з легкістю контролювати роботу модераторів та адміністраторів.
2. Допомагає швидко виявити та вирішувати виникаючі проблеми.
3. Забезпечує додаткову безпеку та захист для серверу шляхом відстеження небажаних дій та запобігання їх у майбутньому.

Недоліки журналу аудиту у Discord:

1. У деяких випадках може бути занадто багато інформації, яку важко та/або не потрібно одразу обробити.
2. Іноді може містити конфіденційну інформацію, яку не слід відкривати для інших учасників сервера, тому варто уважно стежити за тим у кого є права на перегляд.

Наступним прикладом системи з компонентом відстеження змін можна взяти онлайн ресурс **Google Spreadsheets**.

Google Sheets – це програма для роботи з електронними таблицями, яка входить до складу безкоштовного веб-пакета Google Docs Editors, який пропонує Google. Google Таблиці доступні як веб-програма, мобільна програма для: Android, iOS, Microsoft Windows, BlackBerry OS і як комп'ютерна програма на ChromeOS від Google. Програма сумісна з форматами файлів Microsoft Excel. Додаток дозволяє користувачам створювати та редагувати файли онлайн, співпрацюючи з іншими користувачами в режимі реального часу.[13] Редагування відстежуються користувачем за допомогою компоненту історії редагувань, що містить зміни. Також можна вносити правила, за якими система регулює правами які є у користувачів та що вони можуть з ними робити.

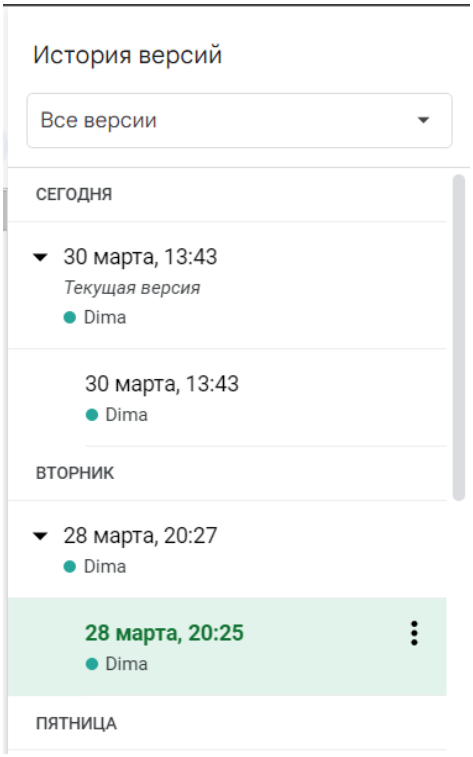
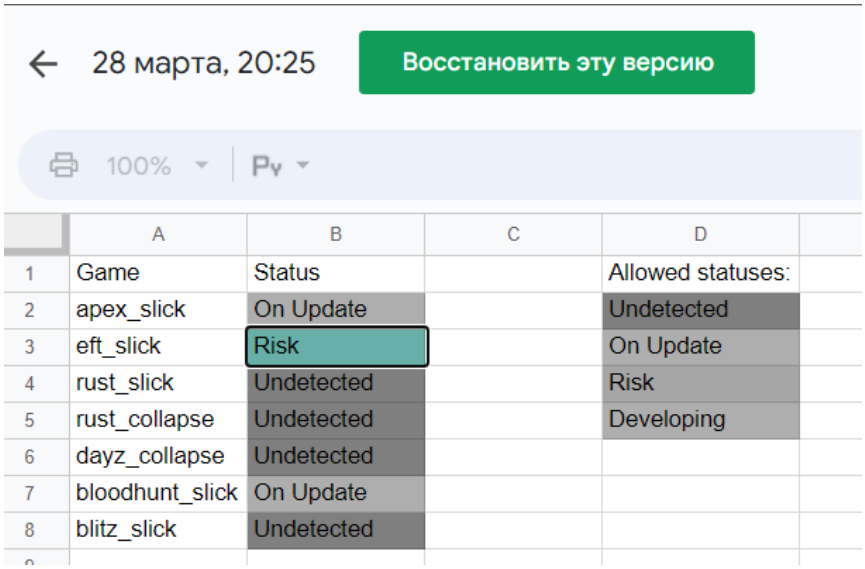


Рис. 1.3.2

На скріншоті вище(рис. 1.3.2) зображено приклад того як виглядає список усіх змін, які були зроблені та збережені з моменту створення

документу. В цілому цей компонент зображений досить ясно та функціонально зрозумілий.



	A	B	C	D
1	Game	Status		Allowed statuses:
2	apex_slick	On Update		Undetected
3	eft_slick	Risk		On Update
4	rust_slick	Undetected		Risk
5	rust_collapse	Undetected		Developing
6	dayz_collapse	Undetected		
7	bloodhunt_slick	On Update		
8	blitz_slick	Undetected		

Рис. 1.3.3

На рисунку 1.3.3 зображено те як саме відображаються виконані зміни. Зручно те, що по натисканню однієї кнопки можна відновити усі зроблеі зміни до цієї версії, але є також нюанси, які були досить незрозумілі. Наприклад виділення клітинок кольорами не збігається з тим що є у оригінальному документі, це може викликати додаткове осмислення того, за що відповідають ці комірки. Також можна виділити клітинки, у яких відбулися зміни, але показується тільки поточне значення на момент збереження, і для того аби подивитися яке значення було до зміни потрібно переходити на версію нижче і вже шукати там потрібну комірку. Але й не факт що саме у попередній версії було зроблено зміну саме у цій комірці.

Тому процес відстеження виконаних змін у комірках документу Google Spreadsheets може бути досить складним і затратним у часі.

РОЗДІЛ 2.
ПРОЕКТУВАННЯ ТА РОЗРОБКА МОДУЛЮ ІСТОРІЇ ЗМІН

2.1. Опис технічного завдання до компоненту історії змін.

Технічне завдання (ТЗ) – це документ, який є описом вимог до продукції або послуги, що розробляється і здатний залагодити всі можливі конфлікти та заощадити час. Безумовно, саме технічне завдання визначає коло завдань, які необхідно виконати при створенні певного продукту.[20] ТЗ виступає як основа для виконання робіт з проектування та розробки, і в ньому докладно описуються вимоги та параметри, якими необхідно керуватися при розробці продукту або послуги.

Технічне завдання містить різні відомості, включаючи цілі та завдання проекту, вимоги до характеристик продукту, засоби контролю якості, критерії здачі проекту тощо. Технічне завдання розробляється на початковому етапі проекту та є важливим етапом у процесі розробки, оскільки від його змісту залежить успішність та ефективність проекту.

1. Короткий опис цілей та задач компоненту історії змін.

Компонент історії змін у веб-додатку призначений для відстеження змін, що відбуваються з даними, він повинен зберігати інформацію про зміни, які були зроблені, і дозволяє адміністраторам та користувачам переглядати історію змін, включаючи дати, авторів та описи змін. Цей компонент призначений для того, щоб мати можливість відслідковувати зміни, які відбуваються у веб-додатку.

2. Функціональні вимоги: опис функціональності, яку має забезпечувати компонент історії змін.

- Компонент повинен виводити інформацію про зміни, які вносяться до бази даних(у API для факультетів і кафедр).
- Вивід даних у вигляді таблицю з певними колонками.
- Можливість гортати сторінки у таблиці.

3. Нефункціональні вимоги до характеристик компонента.

- Безпека: компонент повинен забезпечувати безпечне зберігання та обробку даних, включаючи перевірку користувачів, які наділені правами для перегляду історії змін.
- Компонент повинен легко масштабуватись у разі потреби, наприклад якщо будуть створені інші логування даних у API.

4. Опис вимог до користувальницького інтерфейсу компонента.

- Компонент повинен мати привабливий та сучасний дизайн, що відповідає всьому стилю веб-додатку(бібліотека Ant Design).
- Компонент повинен забезпечувати високу продуктивність під час обробки великого обсягу даних (вивід n-кількості рядків, та пагінація таблиці).
- Дизайн компоненту історії змін:

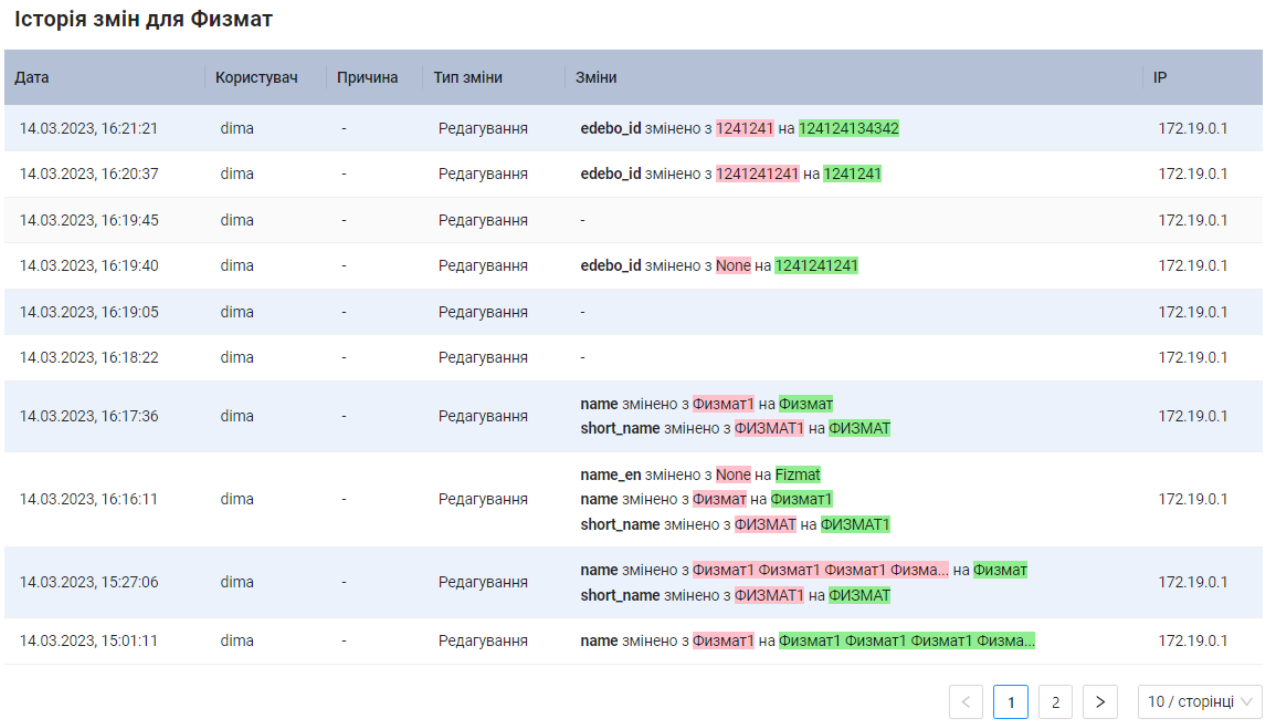


Рис. 2.1.1

Компонент має дизайн у вигляді таблиці з наведеними на скриншоті стовпцями:

1. Дата: точна дата виконаної зміни.
2. Користувач: ім’я користувача, який зробив зміну.
3. Причина, через яку була виконана зміна.

4. Тип зміни: Додавання/Редагування.
5. Зміни: опис змін, який складається з поля, у якому було виконано зміну, старе значення (виділено рожевим), нове значення (виділено світло-зеленим).
6. IP: IP-адреса користувача, який виконав зміну.

5. Опис вимог до тестування компонента.

- Фукнціональні вимоги: компонент має бути протестовано на відповідність функціональним вимогам.
- Користувацький інтерфейс: компонент має бути протестований на відсутність візуальних багів, наприклад проблеми з користувацьким інтерфейсом при зміні розміру вікна, або відображення достовірної таблиці.
- Безпека: протестувати компонент на відсутність проблем з доступом до даних користувачам, які мают права на перегляд, та виключити можливість перегляду тим користувачам, які не мають на це права.

2.2. Огляд ендпоінтів API, що відповідають за логування.

API (інтерфейс прикладного програмування) — це механізми, які дозволяють двом програмним компонентам спілкуватися один з одним за допомогою набору визначень і протоколів. У контексті API слово «Програма» стосується будь-якого програмного забезпечення з окремою функцією. Інтерфейс можна розглядати як контракт на обслуговування між двома програмами. Цей контракт визначає, як обидва спілкуються один з одним за допомогою запитів і відповідей. Їх API документація містить інформацію про те, як розробники структурують ці запити та відповіді.[12]

Майже кожна сутність у базі даних має свою історію змін, яка логується та зберігається. Дістати потрібні дані можна надіславши **GET** запит за такою адресою: **/api/{entity}/{id}/histoty**.

Обов’язкові параметри при запиті:

id – унікальний індентифікатор кожної сутності, за допомогою якого можно отримати потрібні дані.

entity – сутність, для якої потрібно отримати історію змін.

Наприклад, розглянемо запит історії змін факультету:

GET - /api/faculty/{id}/history/

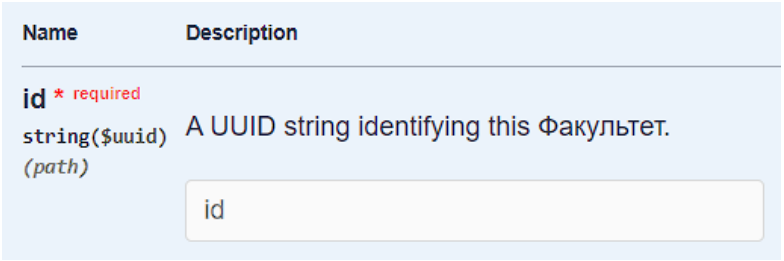


Рис. 2.2.1

Приклад відповіді від сервера у форматі JSON:

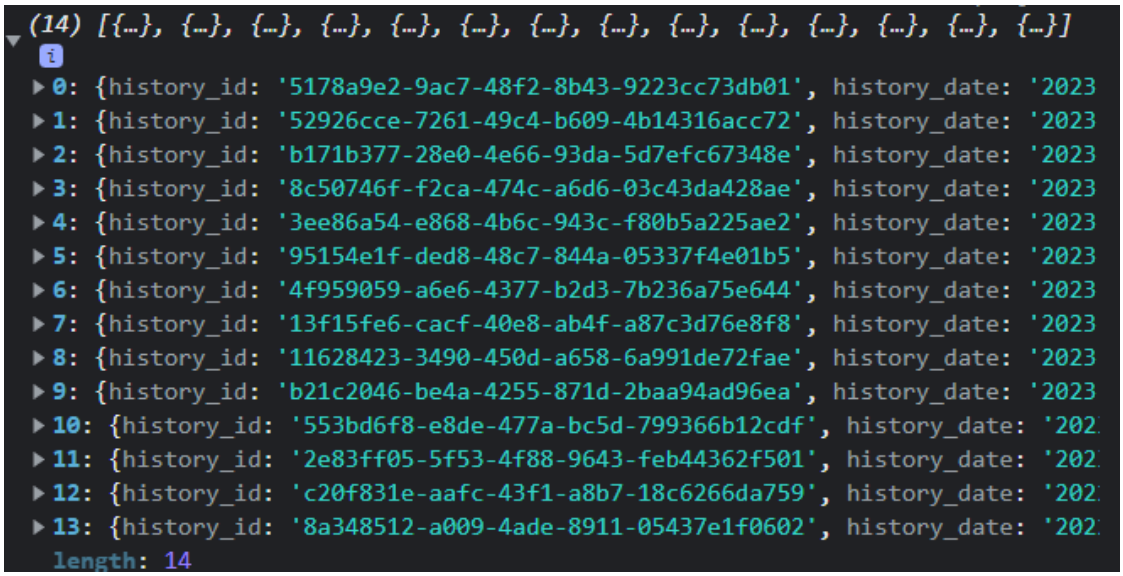


Рис. 2.2.2

На скріншоті вище(рис. 2.2.2) можна побачити, що API відправляє у якості відповіді від серверу масив з об’єктами, де кожен елемент масиву це якась зареєстрована зміна.

Розглянемо детальніше яку інформацію стосовно зміни даних сутності містить в собі кожен об’єкт.

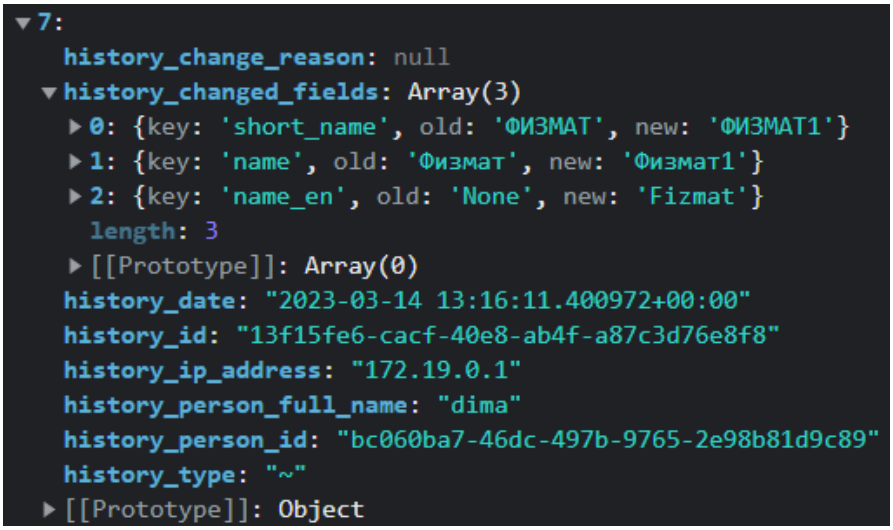


Рис. 2.2.3

На скріншоті вище(рис. 2.2.3) відображено бачимо об’єкт, який описує певну зміну конкретної сутності, та містить такі властивості:

- history_change_reason: опис причини, для чого була зроблена зміна.
- history_changed_field: масив, який містить в собі об’єкти, кожен з яких має такі властивості:
    1) key: ключ, або поле сутності, в якому було зроблено зміни;
    2) old: старе значення;
    3) new: нове значення.
- history_date: точна дата виконаної зміни.
- history_id: унікальний ідентифікатор кожної зареєстрованої зміни.
- history_ip_adress: IP-адреса, з якої була виконана зміна.

- history_person_full_name: ім’я людини, яка виконала зміну.
- history_person_id: унікальний ідентифікатор цієї людини.
- history_type: позначення якого саме типу була виконана зміна, може бути таких видів:
    1) “~” – Редагування;
    2) “+” – Додавання.

2.3. Реалізація та тестування компоненту історії змін.

Для розробки компонента історії змін застосовані можливості бібліотеки React для побудови інтерфейсу користувача, а також бібліотека Ant Design для використання готових компонентів дизайну і створення кастомних компонентів.

React - це бібліотека для мови програмування JavaScript, яка призначена для створення інтерфейсів користувача. React використовує компонентний підхід, що дозволяє розбити інтерфейс на невеликі незалежні компоненти, які можна використовувати повторно. React працює з використанням JSX (розширений синтаксис JavaScript), який дозволяє описувати структуру інтерфейсу користувача у вигляді компонентів і HTML-подібного синтаксису. Для управління станом компонентів React використовується принцип односпрямованого потоку даних. [9]

React також надає безліч функцій та можливостей, таких як робота з подіями, життєвий цикл компонентів, робота зі станом компонента тощо. Завдяки цим можливостям React дозволяє створювати швидкі, масштабовані з можливістю легко підтримувані існуючі користувацькі інтерфейси.

Ant Design - це бібліотека готових до використання компонентів інтерфейсу для React, яка надає багато різноманітних елементів, таких як кнопки, таблиці, форми, навігаційні елементи, модальні вікна і багато

інших. Компоненти Ant Design мають простий API і дозволяють швидко створювати стильні і функціональні інтерфейси користувача, які також мають адаптивний дизайн для мобільних пристроїв.[10] Бібліотека має можливості для перевизначення стандартного дизайну компонентів, таких як налаштування колірної схеми, стилів, типографіки та інших параметрів. Також Ant Design активно підтримується спільнотою розробників та оновлюється регулярно, що гарантує стабільну роботу та актуальність бібліотеки.

Під час розробки компоненту історії змін були використані такі компоненти бібліотеки Ant Design, як: Col, Row, Typography, Tooltip. Але найважливіший це – Table.

Компонент Table із бібліотеки Ant Design для React – це універсальний компонент, призначений для створення таблиць. Цей компонент приймає дві важливі властивості: columns та dataSource[15]:

- columns – це масив об'єктів, що описують колонки таблиці.
- dataSource - це масив даних, які будуть використовуватись для заповнення таблиці. Кожен елемент масиву повинен містити поля, імена яких повинні збігатися з dataIndex у відповідних колонках.

Компонент Table також підтримує безліч інших властивостей, наприклад, loading, pagination, scroll, rowSelection та ін.

```
<Table
  title={() => (
    <Title
      style={{ margin: 0 }}
      level={4}
    >{`Історія змін для ${historyName}`}</Title>
  )}
  columns={columns}
  dataSource={data}
  loading={loading}
  scroll={{ x: true }}
  rowKey="id"
  pagination={{
    showSizeChanger: true,
  }}
/>
```

Рис. 2.3.1

Код, що зображений вище на скріншоті(рис. 2.3.1) відповідає за сам компонент таблиці, яка приймає дані, та має свої налаштування, такі як: відображення спінеру під час завантаження даних, гортання сторінок, окремі колонки з даними, тощо.

```
{
  title: 'Зміни',
  dataIndex: 'history_changed_fields',
  render: changes => {
    if (!changes || Object.keys(changes).length === 0) return '-';

    const rowOutput = changes.map(({ key, old, new: newv }) => {
      return (
        <p>
          <b className="bold">{key}</b> змінено з{' '}
          <Tooltip placement="top" title={old}>
            <span className="pink">{truncString(old, 30)}</span>
          </Tooltip>{' '}
          на{' '}
          <Tooltip placement="top" title={newv}>
            <span className="lightgreen">{truncString(newv, 30)}</span>
          </Tooltip>
        </p>
      );
    });

    return rowOutput;
  },
},
```

Рис. 2.3.2

На рисунку 2.3.2 зображений код, що відповідає за форматування та виведення колонки зі змінами. Властивості об’єкту що відповідає за колонку змін:

- title – назва колонки, яка буде виведена у першому рядку таблиці,
- dataIndex – це те поле об’єкту з історією змін, який відправляє сервер, яке потрібно індексувати та обробляти при виводі у рядках таблиці. На рисунку 2.2.3 це поле “history_changed_fields”,
- render – це функція яка рендерить(повертає) вже сформатоване значення у рядок.

Додана перевірка на те, якщо змін немає то виводити знак «-». Також додано обрізання рядку, якщо він містить більше ніж 30 символів, а для того щоб побачити рядок повністю є елемент Tooltip.

Компонент Tooltip з бібліотеки Ant Design призначений для додавання підказки на елементи інтерфейсу. Він має дві обов'язкові властивості: title та children. Властивість title задає текст підказки, а властивість children задає елемент, на якому відображатиметься підказка.[16] Тобто, з прикладу вище, це буде обрізана строка, яка що відповідає за старе або нове значення, відповідно до кольору її підкреслення. Також застосовано іншу властивість компоненту Tooltip, яка має назву placement, тобто з якої частини від компоненту відображати підказку. У наведеному прикладі placement має значення «top» (зверху).

Усі інші стовпці у таблиці мають досить просту та зрозумілу логіку, найбільша увага при розробці була приділена стовпцю з виведенням змін, саме тому алгоритм роботи цього стовця і описаний вище.

У якості тестування роботи компоненту було вирішено додати компонент, який відповідає за виведення історії змін, у окремий розділ «Історія змін» (див. рис. 2.3.3) на сторінці management (управління веб-додатком та ресурсами).

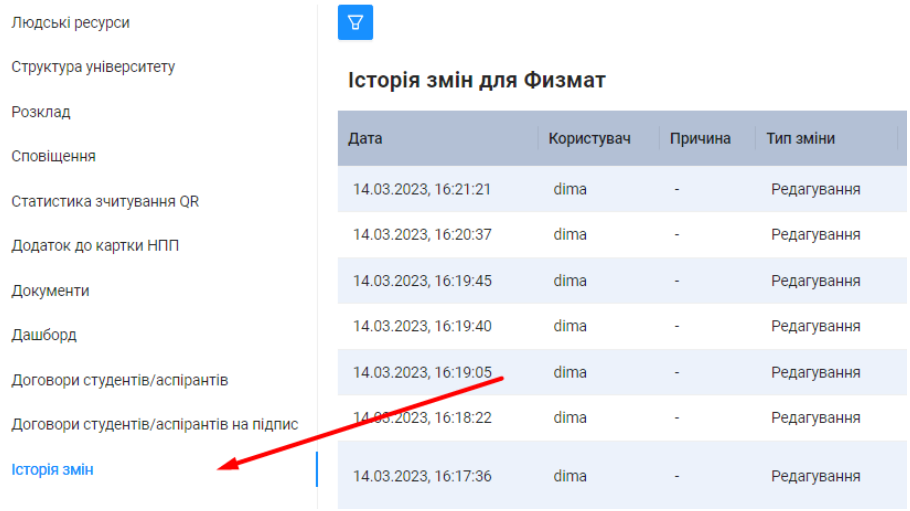


Рис. 2.3.3

Додана можливість запитувати потрібну історію змін деякої сутності за допомогою значка фільтру, який знаходится вище таблиці.

Тестування компоненту історії змін виявило те, що він відповідає технічному завданню стосовно дизайну та потрібних даних, але запитувати потрібні дані не зовсім зручно за допомогою фільтрів. Тому, було вирішено виокремити цей компонент як модальне вікно за допомогою компоненту Modal з бібліотеки Ant Desig, та додати окрему іконку змін, яка показує модальне вікно при кліканні на неї.

Modal – компонент бібліотеки Ant Design для створення модальних вікон, які можуть містити різноманітні повідомлення, повідомлення, діалогові вікна тощо.[17] Він використовується для того, аби поліпшити користувацький інтерфейс та забезпечити більш зручну і зрозумілу інтерактивність на сторінці.

Модальне вікно з'являється поверх вмісту сторінки, блокуючи решту вікна та надаючи користувачеві додаткові можливості. Також можуть бути додані різні кнопки та елементи керування для зручнішої роботи з модальним вікном. Компонент Modal є дуже важливим елементом інтерфейсу веб-додатків, оскільки дозволяє взаємодіяти з користувачем без перенавантаження сторінок і зручно обмінюватися інформацією.

Для тестування результатів та відображення, логіка роботи модального вікна з історією змін була реалізована у підрозділі «Студенти», з розділу «Людські ресурси» на сторінці менеджменту.

У підрозділі «Студенти», а саме у таблиці зі стовпчиком «ПІБ» була додана кнопка, яка містить іконку історії змін.

```
<ProtectedRender allowedUserRoles={[ADMIN]}>
  <Button
    type="text"
    style={{ paddingLeft: 5 }}
    onClick={e => {
      e.stopPropagation();
      setIsHistoryModalOpen(true);
      setHistoryModalProps({
        name: `${row.person.surname} ${row.person.name}`,
        endpoint: `student/${row.id}`,
      });
    }}
  >
    <HistoryOutlined />
  </Button>
</ProtectedRender>
```

Рис. 2.3.4

На рисунку вище зображений фрагмент коду, що відповідає за цю кнопку. На ньому можна побачити, що цей міні-компонент повність обернутий у ProtectedRender, таким чином, що доступ мають лише користувачі з роллю адміністратора, а для всіх інших ця кнопка не буде відображатись.

Для роботи з кнопкою використовується комопонент Button з бібліотеки Ant Design.

Компонент Button – це елемент інтерфейсу, який дозволяє користувачеві при натисканні виконати певну команду або викликати певну функцію. У бібліотеці Ant Design компонент Button є елементом, що широко настроюється, він має властивості, такі як type, який визначає колір фону, size, який керує розміром кнопки, та disabled, який блокує кнопку від натискання.[18]

Модифікації кнопки можуть бути додані з різних бібліотек, наприклад іконки, готові шаблони і т.д. Це скорочує час на створення дизайну та фокусує увагу на архітектурному рівні та бізнес-логіці веб-додатків, замість витрачання часу на створення елементів з нуля.

При натисканні кнопки відбувається назначення стану модального вікна за допомогою фукнції setIsHistoryModalOpen() як true, тобто відкрите, а також передаються необхідні властивості до модального вікна за допомогою функції setHistoryModalProps(), такі як ім’я того об’єкту, історія змін якого виведется там сам запит до API з конкретним ID об’єкта.

```
const [isHistoryModalOpen, setIsHistoryModalOpen] = useState(false);
const [historyModalProps, setHistoryModalProps] = useState({});
```

Рис. 2.3.5

Ці функції керують станом компонента, завдяки хуку useState із бібліотеки React, які присвоюють значення змінним, за якими вони зафіксовані, а вже потім ці змінні з поточними даними передаються до модального вікна.

Хук useState - це один з основних хуків React, який використовується для додавання змінних, що відповідають за стан компонентів функціонального типу. Щоразу, коли змінна стану переназначається, компонент ре-рендериться, що дозволяє відобразити оновлений стан у інтерфейсі користувача.[19]

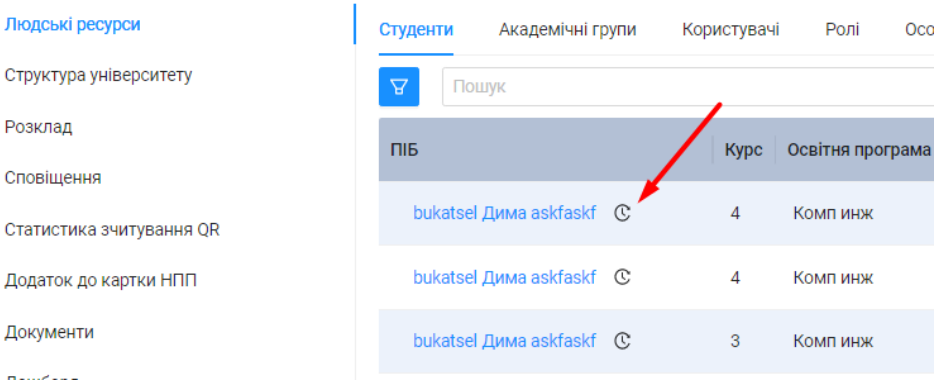


Рис. 2.3.6

На скріншоті вище (рис. 2.3.6) зображена іконка, що відповідає за відображення модального вікна з історією змін потрібного об’єкту. Цю іконку можуть побачити лише користувачі у яких наявна роль адміністратора, тому побачити історію змін зможуть лише користувачі з відповідною роллю.

Розроблений компонент модального вікна з таблицею історії змін має такі властивості:

- open – властивість, що відповідає за те, який стан у модального вікна, тобто відкрите/закрите.
- onCancel – властивість, що приймає фукнцію, яка буде виконана після того, коли користувач зробить певну дію для того аби закрити модальне вікно.
- footer – приймає значення React-компоненту, який буде відображатися внизу модального вікна.
- width – ширина, яку буде займати модальне вікно відносно усього розміру екрану.

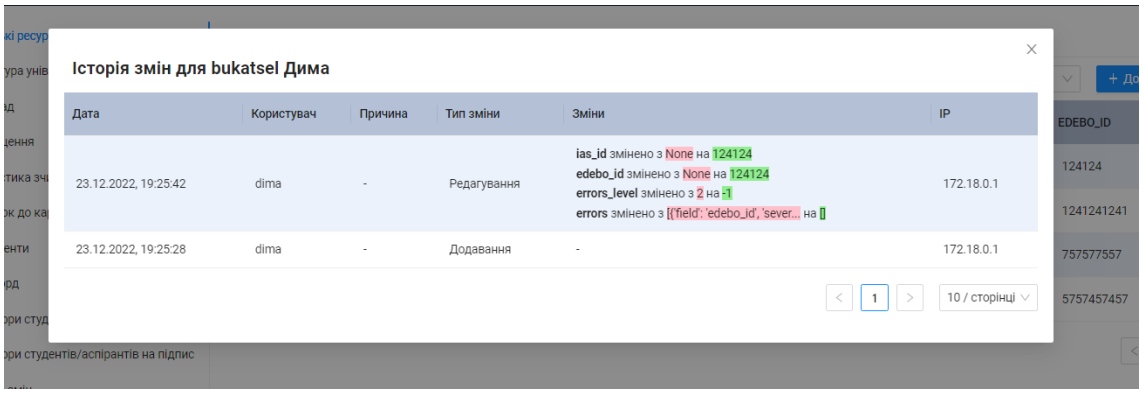


Рис. 2.3.7

На рисунку 2.3.7 зображено приклад вигляду модального вікна, яке містить у собі таблицю з усіма змінами, які було виконано з поточним об’єктом.

Таким чином, цей спосіб відображення є набагато зручнішим, ніж попередній, тому що можна одразу переглянути необхідні зміні, а не переходити у іншу вкладинку та робити пошук за допомогою фільтрів.

Кожен раз при відкритті модального вікна туди передається необхідна інформація щодо запиту та назва таблиці і робиться відповідний запит на сервер за допомогою React хуку useEffect.

Хук useEffect в React надає можливість виконати сторонні-ефекти у функціональних компонентах, який викликається під час різних стадій життєвого циклу компонента. Наприклад, створення компоненту у DOM-дереві, оновлення його або видалення. Хук useEffect викликається після кожного відтворення компонента і виконує передану йому функцію.[14]

Функція, передана в хук useEffect, може також повертати функцію, яка буде виконана при видаленні компонента з DOM-дерева. Це дуже корисно для відписки від подій чи очищення ресурсів тощо.

```
useEffect(() => {
  updateData(null, `api/${endpoint}/history`);
}, []);
```

Рис. 2.3.8

На рисунку 2.3.8 зображено фрагмент коду, який відповідає за запит на сервер, та оновлює дані, які відображаються у модальному вікні у таблиці історії змін.

У змінну **endpoint** приходить значення у форматі **“entity/id”**. Тому, якщо розглядати приклад з історією змін кожного студента, тоді endpoint матиме вигляд student/student-id. Кінцевий запит до серверу буде – api/student/student-id/history, який поверне необхідні дані для відображення їх у таблиці історії змін.

Компонент історії змін створений окремо від сторінок, тому він може легко бути перевикористаний у інших місцях веб-додатку ХДУ24, де потрібно дізнатися історію змін певної сутності. Це може значно скоротити час при розробці нових компонентів, замість того аби створювати модальне вікно на кожній сторінці, є універсальний компонент, який відповідає за ці дії. Також якщо потрібно внести якісь корегування у компонент історії змін достатньо лише змінити один файл, замість того аби змінювати у всіх місцях де він використовується.

Компонент працює коректно та відповідає очікуваному функціоналу, тобто він відображає історію змін даних у модальному вікні у будь-якому місці веб-додатку. Компонент відповідає вимогам якості програмного забезпечення та рекомендаціям щодо написання коду, таким як: відповідність стандартам розробки React-компонентів, відсутність помилок та неефективних частин коду. Компонент легко піддається супроводу та доопрацюванню у разі зміни вимог до функціональності чи дизайну, готовий до інтеграції з іншими компонентами, що дозволяє створити складніші веб-додатки.

Отже, успішне тестування компонента гарантує його роботу відповідно до технічного завдання, що гарантує користувачеві зручність використання продукту та зменшує ризик виникнення помилок, пов'язаних з роботою даного функціоналу.

ВИСНОВОК

У кваліфікаційній роботі була розглянута актуальна тема розробки компонента історії змін для веб-додатку ХДУ24, який є важливим інструментом для відстеження та відновлення даних, а також забезпечує інформацією про останні зміни.

Метою дослідження було вивчення способів розробки компонента історії змін та розгляд аналогів. На основі отриманої інформації було реалізовано компонент логування у проекті ХДУ24, що дозволяє відстежувати зміни в даних, зберігати попередні версії та відновлювати дані у разі їх пошкодження або непередбаченої зміни.

У теоретичній частині кваліфікаційної роботи було наведено переваги використання компонента історії змін у веб-додатку для можливість відстеження змін у базі даних, отримання інформації про останні оновлення тощо. Також було розглянуто методи, які зазвичай використовуются для запису логів історії дій користувача.

В результаті проведеного дослідження було з'ясовано, що компонент історії змін є важливим інструментом для покращення процесу управління веб-додатком, а також для забезпечення надійності та збереження даних. Для досягнення мети дослідження було виконано такі завдання:

- У ході дослідження було вивчено різні методи запису історії змін, у тому числі призначення журналу аудиту та створення версій даних, проаналізовано різні методи логування історії дій користувача, які можуть використовуватися у веб-додатку. Виявлено, що використання компонента історії змін може значно підвищити ефективність управління веб-додатком, адже завдяки цьому компоненту можна відстежувати зміни, і, у разі потреби відновлювати непередбачувані дії.

- Проведено аналіз готових рішень у системах із компонентом історії змін. Було проведено аналіз функціональності, переваг та недоліків кожного з інструментів.
- Було розроблено технічне завдання для компонента історії змін у додатку ХДУ24, в якому визначено необхідні функціональні та нефункціональні вимоги, а також технічні особливості, такі як вимоги до відображення користувацького інтерфейсу та дизайн.
- Було зроблено огляд ендпоінтів API, що відповідають за логування, щоб визначити, які запити мають бути зроблені для отримання даних історії змін, обов’язкові параметри. Також було проаналізовано відповідь від сервера, який містить об’єкт з усіма зробленими змінами, відповідно до сутності, яку було запитано. Проаналізовано та описано поля, що містить цей об’єкт.
- Реалізувано компонент історії змін у веб-додатку ХДУ24, забезпечивши його роботу відповідно до технічного завдання. Цей компонент призначений для відображення історії змін відповідно до заданих параметрів.

В результаті виконаних завдань було досягнуто поставленої мети, створення компоненту історії змін для веб-додатку ХДУ24.

Отже, курсова робота є актуальною та практично значущою, оскільки вона розглядає важливий аспект розробки веб-додатків. Результати дослідження можуть бути використані для покращення якості та надійності веб-застосунків, а також для підвищення зручності використання їх користувачами.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Лог-файл [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://blog.skillfactory.ru/glossary/log-fajl/.
2. Логирование: понятие, механизмы и уровни [Електронний ресурс] – Режим доступу до ресурсу: https://www.datahata.by/info/articles/logirovanie.html.
3. What Are Log Files? [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/what-is/log-files/?nc1=h_ls.
4. THE IMPORTANCE OF LOGGING [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.hexacta.com/importance-logging-introducing-elastic-stack/.
5. Overview of the Binary Log [Електронний ресурс]. – 2004. – Режим доступу до ресурсу: https://mariadb.com/kb/en/overview-of-the-binary-log/.
6. THE KEY BENEFITS OF LOG DATA [Електронний ресурс] – Режим доступу до ресурсу: https://www.mezmo.com/learn-log-management/the-key-benefits-of-log-data.
7. Audit Logging Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.datadoghq.com/knowledge-center/audit-logging/.
8. Database logging [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.ibm.com/docs/en/db2/11.5?topic=strategies-database-logging.
9. What is React? [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/learn.
10. Ant Design of React [Електронний ресурс] – Режим доступу до ресурсу: https://ant.design/docs/react/introduce/.

11. WHAT IS DISCORD? [Електронний ресурс] – Режим доступу до ресурсу: https://discord.com/safety/360044149331-what-is-discord.
12. What Is An API (Application Programming Interface)? [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/what-is/api/.
13. Google Sheets [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Google_Sheets.
14. useEffect – React [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://react.dev/reference/react/useEffect.
15. Table - Ant Design [Електронний ресурс] – Режим доступу до ресурсу: https://ant.design/components/table.
16. Tooltip - Ant Design [Електронний ресурс] – Режим доступу до ресурсу: https://ant.design/components/tooltip.
17. Modal - Ant Design [Електронний ресурс] – Режим доступу до ресурсу: https://ant.design/components/modal.
18. Button - Ant Design [Електронний ресурс] – Режим доступу до ресурсу: https://ant.design/components/button.
19. useState - React [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/reference/react/useState.
20. How to write technical task? [Електронний ресурс] – Режим доступу до ресурсу: https://encomage.com/2020/04/27/how-to-write-technical-task.

ДОДАТКИ

ДОДАТОК А. КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Букацель Дмитро Олександрович, учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

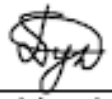
ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:
- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:
- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;
– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;
– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;
– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;
– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;
– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;
– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;
– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;
– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науководослідницькі завдання;
– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;
– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;
– не підроблювати документи;
– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;
– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;
– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;
– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;
– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;
– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;
– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

12.09.2019		Букацель Дмитро
(дата)	(підпис)	(ім'я, прізвище)