ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА
(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій
(повне найменування факультету)

Кафедра інформаційних технологій
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка чат-боту з посиленим захистом персональних даних для взаємодії учасників освітнього процесу»
(тема кваліфікаційної роботи українською мовою)

«Development of a chatbot with personal data enhanced protection for the educational process participants interaction»
(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач заочної форми навчання
спеціальності 122 Комп'ютерні науки
(код, назва спеціальності)
Освітня програма Комп'ютерні науки
(назва)
Зябухін Тарас Вікторович
(прізвище, ім'я, по-батькові здобувача)

Керівник ст. викладач Вохменцева Т.Б.
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент Корчемний П.А.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .
№ 1 від 09 червня 2024 р.

Завідувачка кафедри
(підпис) КАЗАКОВА Надія .
(прізвище,ім'я)

Захищено на засіданні ЕК № 13 .
протокол № 2 від 19 червня 2024 р.

Оцінка відмінно / A / 95 .
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК
(підпис) КОПИЧЕНКО Іван .
(прізвище,ім'я)

Одеса 2024

ЗМІСТ

ТЕРМІНИ, СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Архітектура веб-застосунка – це структура та організація програмного забезпечення, яка визначає спосіб взаємодії різних компонентів веб-застосунка, включаючи клієнтську та серверну сторони. Вона описує, як застосунок розподіляється між різними середовищами та як вони спілкуються між собою для забезпечення функціональності та продуктивності.

База даних – це організована колекція даних, яка зберігається та управляється з допомогою комп'ютерної системи. База даних призначена для ефективного зберігання, організації і використання інформації, яка може бути доступна для багатьох користувачів та застосунків.

Бот – це автоматизований обчислювальний агент, який працює в інформаційній системі та взаємодіє з користувачами через текстові повідомлення, команди, кнопки та інші елементи інтерфейсу.

Веб-сервер – це програмне забезпечення, яке надає доступ до ІС та веб-застосунків через Інтернет. Веб-сервер обробляє запити користувачів та відправляє їм відповіді у вигляді веб-сторінок.

Інформаційна система – це система, яка збирає, зберігає, обробляє, аналізує та використовує дані з метою надання користувачам інформації та підтримки прийняття рішень. Вона може містити як апаратне, так і програмне забезпечення, а також людські ресурси.

Клієнт – це програмне забезпечення або пристрій, який взаємодіє з веб-сервером для отримання веб-сторінок або інших ресурсів.

ООП – це парадигма програмування, яка базується на використанні об'єктів та їх взаємодії.

Реляція – це взаємозв'язок між двома або більше сутностями в базі даних.

Сервер – це програмне забезпечення або комп'ютер, який надає веб-ресурси (наприклад, веб-сторінки) клієнтам через Інтернет.

СКБД – це програмне забезпечення, яке дозволяє створювати, управляти та оптимізувати бази даних.

API – це набір правил, протоколів та інструментів, що дозволяють різним програмам взаємодіяти одна з одною.

Apache – це один з найпопулярніших веб-серверів у світі. Apache є вільним програмним забезпеченням та відомий своєю гнучкістю та розширюваністю. Він широко використовується для розгортання ІС та веб-застосунків.

Artisan – це вбудована у Laravel консольна утиліта, яка надає широкий набір команд для роботи з застосунком. Artisan дозволяє виконувати різноманітні завдання, такі як створення контролерів, міграції баз даних, запуск тестів тощо.

Backend – це частина веб-застосунка, яка відповідає за обробку запитів користувачів, взаємодію з базою даних та виконання бізнес-логіки.

Blade – це шаблонний рушій, який використовується в Laravel для генерації HTML-коду.

Composer – це менеджер залежностей для PHP, який дозволяє легко встановлювати та керувати бібліотеками та пакетами для розробки програмного забезпечення на PHP.

Database Migration – це процес автоматизованого керування еволюцією схеми бази даних.

Laravel – це відкритий PHP-фреймворк для розробки веб-застосунків.

Middleware – Це програмне забезпечення або компонент, який знаходиться між різними системами чи програмами, що спілкуються між собою.

Nginx – це вільний веб-сервер та проксі-сервер, який відомий своєю високою продуктивністю, надійністю, простотою налаштування та низьким використанням ресурсів.

ORM – це технологія програмування, яка дозволяє зв'язувати об'єкти програмного коду з записами в реляційній базі даних.

PHP – це скриптова мова програмування загального призначення, яка часто використовується для розробки веб-застосунків та ІС [5].

Sail – це набір інструментів для розробки та розгортання Laravel-застосунків у середовищі Docker.

Webhook – це механізм, який дозволяє ІС та веб-застосункам автоматично отримувати оновлення або повідомлення від іншого сервісу чи застосунка, коли вони стають доступними, без необхідності постійного опитування сервера для отримання цих даних.

ВСТУП

У сучасному освітньому середовищі, де технології займають ключову роль, захист особистих даних учасників освітнього процесу стає вкрай важливим завданням. Використання електронних засобів комунікації в навчальних закладах з одного боку сприяє підвищенню ефективності навчального процесу, але з іншого боку, несе ризики втрати конфіденційності та безпеки даних.

Метою кваліфікаційної роботи бакалавра є розробка чат-бота для месенджера Telegram, який забезпечить безпечну та конфіденційну комунікацію між викладачами та студентами. Основна перевага такого чат-бота полягає в його здатності утримувати особисті дані користувачів у безпеці шляхом анонімізації спілкування. Викладачі та студенти можуть обмінюватися інформацією без ризику розголошення своїх особистих даних. Крім того, передбачена розробка адміністративної панелі, яка дозволить уповноваженим особам навчального закладу контролювати функціонування чат-бота та забезпечувати його відповідність законодавчим вимогам щодо захисту персональних даних.

Задачі, які ставляться перед розробленим чат-ботом, включають забезпечення безпечної комунікації між учасниками освітнього процесу та надання зручної можливості для спілкування через розповсюджений месенджер Telegram. Це означає, що чат-бот повинен забезпечити шифрування передачі даних та використовувати механізми аутентифікації, щоб уникнути несанкціонованого доступу до інформації. Крім того, важливо забезпечити зручний та інтуїтивно зрозумілий інтерфейс для користувачів, щоб забезпечити ефективну комунікацію без зайвих труднощів або перешкод.

Дана кваліфікаційна робота бакалавра складається з 52 сторінок, 15 рисунків, 2 таблицьта 9 джерел посилання.

1. РОЗГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУЛЮВАННЯ ЗАВДАННЯ

1.1 Дослідження предметної області

У сфері освіти важливим аспектом є забезпечення надійної та конфіденційної комунікації між учасниками навчального процесу. Викладачі та студенти регулярно обмінюються інформацією через різні канали зв’язку, включаючи особисті номери телефонів та соціальні мережі. Такий обмін інформацією, хоча і зручний, створює значні ризики для безпеки персональних даних. Відсутність належного захисту може призвести до витоку інформації та її використання в недобросовісних цілях, таких як спам, шахрайство або надмірне контактування.

Учасники освітнього процесу можуть використовувати особисті дані викладача для недобросовісних цілей, таких як надмірне контактування, надсилання спаму або навіть використання цих даних у шахрайських схемах.

Реалізація захищеного та безпечного засобу комунікації, такого як чат-бот у Telegram, стає ефективним рішенням для уникнення таких ризиків. Він дозволяє учасникам освітнього процесу спілкуватися анонімно та без використання особистих ідентифікаторів, забезпечуючи при цьому необхідний рівень конфіденційності. Однією з ключових переваг чат-бота є його здатність утримувати особисті дані у безпеці. Чат-бот функціонує за принципом анонімності, що не надає учасникам освітнього процесу доступ до особистої інформації один про одного. Крім того, наявність адміністративної панелі для уповноваженої особи в навчальному закладі робить можливим контроль за функціонуванням та управлінням чат-ботом, забезпечуючи відповідність законодавству про захист персональних даних. Системи управління навчальним процесом, такі як Moodle, хоча і забезпечують платформу для спілкування та роботи, мають свої обмеження у забезпе-

ченні ефективної та зручної комунікації між студентами та викладачами. У порівнянні з месенджерами, системи типу Moodle можуть відчувати недоліки у швидкості відповіді та розширених можливостях комунікації. Такі обмеження можуть вплинути на природу комунікації та змусити учасників освітнього процесу знаходити альтернативні шляхи зв'язку, де можливий обмін особистими даними через месенджери.

Проєктування та реалізація захищеного чат-бота для взаємодії учасників освітнього процесу відповідає наступним цілям: забезпечення конфіденційності особистих даних, зручності та ефективності комунікації, а також зменшення ризиків пов'язаних з використанням особистих ідентифікаторів у цілях, не передбачених освітнім процесом. Додатковим ризиком при обміні особистими даними через месенджери є можливість порушення конфіденційності в результаті несанкціонованого доступу до даних або використання недостатньо захищених платформ. Наприклад, якщо студент чи викладач використовує месенджери для обміну особистими даними, такими як номери телефонів або ідентифікатори в соціальних мережах, ця інформація може стати доступною зловмисникам у разі компрометації обраної платформи. Внаслідок цього може відбутися недозволене використання особистої інформації або її неправомірне розповсюдження.

Проте, чат-бот, спеціально розроблений для освітнього процесу у Telegram чи інших месенджерах, може стати ефективним рішенням для забезпечення конфіденційності та безпеки особистих даних учасників. Його можна налаштувати таким чином, що він не вимагатиме обміну особистими ідентифікаторами, а замість цього надаватиме можливість анонімного спілкування за допомогою групових чатів, шифрування повідомлень та інших заходів для захисту приватності.

Отже, розробка чат-бота як засобу комунікації в освітньому середовищі може розв'язувати проблему обміну особистими даними через месенджери шляхом забезпечення анонімності, безпеки та зручності спілкування. Дослідження

імплементації подібних рішень у різних інформаційних системах (ІС) дозволить забезпечити оптимальний вибір платформи для безпечного обміну інформацією та покращення комунікаційного процесу в освітньому середовищі.

1.2 Формулювання завдання

Метою роботи є розробка безпечного та зручного чат-бота для забезпечення ефективної комунікації між учасниками освітнього процесу Основними компонентами системи є студенти та викладачі, які зможуть зручно спілкуватися через популярний месенджер Telegram без необхідності встановлення додаткового програмного забезпечення або використання сторонніх веб-чатів та сервісів. Враховуючи все вищеописане завданнями проєкту є:

- створення інтерфейсу чат-бота, який дозволить студентам та викладачам легко та ефективно обмінюватися інформацією;
- впровадження механізмів шифрування даних для захисту інформації, що передається між користувачами;
- використання механізмів автентифікації для запобігання несанкціонованому доступу до інформації;
- розробка гнучкої системи налаштувань, яка дозволить адаптувати функціональність бота відповідно до потреб користувачів та вимог освітнього закладу;
- забезпечення можливості інтеграції додаткових модулів та сервісів для розширення функціональності чат-бота;
- створення адміністративної панелі для керування чат-ботом, що дозволить уповноваженим/відповідальним особам навчального закладу контролювати та адмініструвати роботу бота, забезпечуючи відповідність законодавчим вимогам щодо захисту персональних даних.

Реалізація цих завдань дозволить створити ефективний та зручний інструмент для взаємодії між учасниками освітнього процесу, сприяючи покращенню комунікації та підвищенню ефективності навчання та викладання.

1.3 Визначення вимог до розробки системи

Створення інформаційної системи (ІС) для взаємодії учасників освітнього процесу вимагає ретельного врахування ключових аспектів, що забезпечують безпеку, зручність та функціональність для ефективного обміну інформацією. Під час визначення вимог до системи необхідно зробити акцент на надійності та конфіденційності особистих даних, що є основою для забезпечення довіри користувачів. До основних вимог можна віднести:

а) вимоги до безпеки та конфіденційності (система повинна гарантувати захист особистих даних користувачів);
   1) шифрування особистих даних користувачів під час реєстрації за допомогою номеру залікової книжки або номеру телефона та під час обміну повідомленнями;
   2) механізми валідації даних та впровадження засобів перевірки та захисту від несанкціонованого доступу до інформації;

б) функціональність чат-бота та адміністративної панелі;
   1) реєстрація та аутентифікація студентів за допомогою номеру залікової книжки;
   2) реєстрація та аутентифікація викладачів за допомогою номеру телефону;
   3) можливість додавання адміністраторів кафедр та деканатів, додавання студентів та викладачів;

4) реалізація зручного обміну різними типами повідомлень: текстові, голосові, файлові тощо;
5) механізми адміністрування діалогів між учасниками освітнього процесу;

в) забезпечення зручності та доступності;
1) інтеграція системи з Telegram для зручності учасників освітнього процесу;
2) наявність всіх функціональних можливостей у звичному месенджері без встановлення додаткового ПЗ чи використання сторонніх сервісів;

г) вимоги до ефективності та швидкодії системи;
1) забезпечення швидкої обробки та передачі повідомлень;
2) максимальна реакція чат-бота на запити користувачів для забезпечення плавної та ефективної роботи системи;

д) додаткові функції адміністративної панелі;
1) можливість блокування діалогів за потреби адміністратором;
2) можливість редагування не активованих викладачів та студентів;
3) можливість блокування діалогів при необхідності адміністратором;
4) облік та керування активними користувачами системи.

Узагальнюючи вимоги до створення системи, важливо враховувати безпеку, функціональність, зручність, швидкодію та доступність для забезпечення найвищого рівня задоволення потреб учасників освітнього процесу. Ці вимоги є критично важливими для розробки ефективної та надійної ІС, яка буде використовуватися для комунікації в освітньому середовищі.

2. АНАЛІЗ ВИБОРУ АРХІТЕКТУРИ ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ЧАТ-БОТА

Для реалізації чат-бота та адміністративної панелі (в рамках даної ІС) було обрано технології, що відповідають вимогам до швидкості розробки та зручності підтримки. Оптимальний вибір компонентів та інструментів дозволяє зосередитися на функціональності системи, мінімізуючи час і зусилля, необхідні для розробки та підтримки. Використання вже готових компонентів сприяє ефективній імплементації необхідних функцій, знижуючи витрати часу на їх розробку. Для забезпечення гнучкості та можливості модифікації використовуються інструменти, що дозволяють розширювати функціональність системи за потреби. Розробка та тестування на початку створення чат-боту та його компонентів відбувається в локальному середовищі, що забезпечує ефективне відлагодження та реалізацію за допомогою спеціально підготовлених контейнерів.

2.1 Аналіз загальної архітектури чат-бота

Під час розробки чат-бота та адміністративної панелі, для забезпечення комунікації між учасниками освітнього процесу, обрано використання «Цибулевої архітектури». Цибулева архітектура – це структура програмного забезпечення, де логіка застосунку розділяється на п'ять рівнів: Presentation Layer, Application Layer, Domain Layer, Infrastructure Layer, та Persistance Layer (див. рис.1).

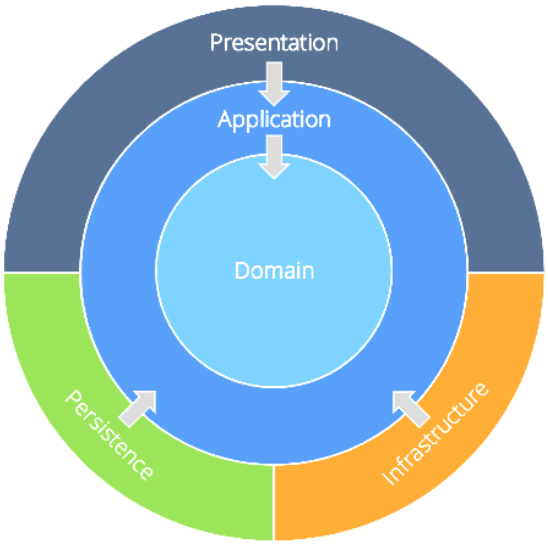

Рисунок 1 – Схема цибулевої архітектури

Presentation Layer – це рівень, який відповідає за відображення інтерфейсу для взаємодії користувача з системою. Даний рівень відповідає за взаємодію користувача з системою. У контексті розробки чат-бота для освітніх цілей, цей шар включає інтерфейс адміністративної панелі, де уповноважені особи можуть керувати функціонуванням чат-бота. Шар забезпечує зручний інтерфейс для управління користувачами та повідомленнями.

Рівень Application Layer відповідає за логіку застосунку. В цьому випадку, цей рівень інтегрує логіку взаємодії з ботом через Telegram API та обробку команд, що надходять через webhook.

Domain Layer в свою чергу відповідає за обробку основних даних, що використовуються в чат-боті, таких як інформація про користувачів, повідомлення та інші важливі дані. Він забезпечує обробку правил та алгоритмів, що визначають функціонування системи.

Рівень Infrastructure Layer створює зв’язок між даними та логікою застосунку. Використовуючи патерн репозиторію для роботи з базою даних (БД),

Infrastructure Layer забезпечує ефективну взаємодію з даними. Він відповідає за управління доступом до БД, забезпечуючи її безпеку та консистентність.

Persistance Layer відповідає за збереження та управління даними. Цей шар забезпечує незалежність від джерела даних та надає стандартизовані інтерфейси для доступу до даних з інших компонентів системи. Його основною функцією є забезпечення постійного зберігання даних і забезпечення їх доступності для інших частин програмного забезпечення.

Вибір «Цибулевої архітектури» для реалізації чат-бота дозволяє розподілити логіку системи на чітко визначені рівні, що сприяє покращенню структурованості та підтримці коду. Використання патерну репозиторію для доступу до бази даних полегшує взаємодію з інформацією та забезпечує її безпеку та консистентність. Система також розділена на дві основні частини: адміністративну панель та чат-бота для комунікації з учасниками освітнього процесу через Telegram (рис.2). Адміністративна частина використовує підходи, властиві Model-View-Controlle (MVC) патерну, для забезпечення управління системою через інтерфейс користувача. Це дозволяє відокремити логіку бізнес-процесів від представлення даних користувачам, сприяючи простоті розширення та підтримки коду.

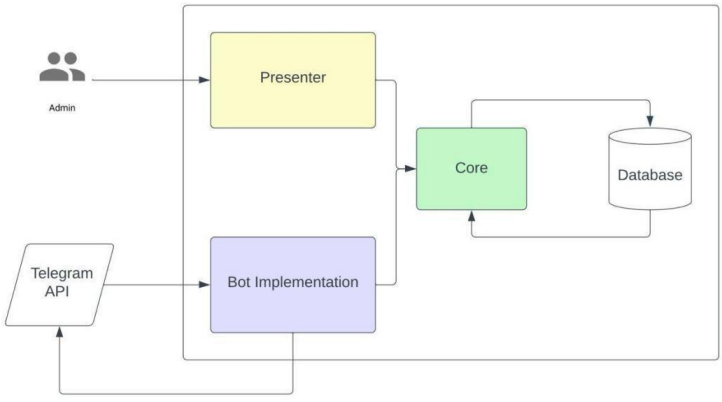

Рисунок 2 – Загальна схема чат-бота

У випадку чат-бота, взаємодія з Telegram клієнтом відбувається через webhook, що дозволяє отримувати повідомлення та команди від користувачів і обробляти їх за допомогою відповідної логіки застосунку (див. рис. 3).

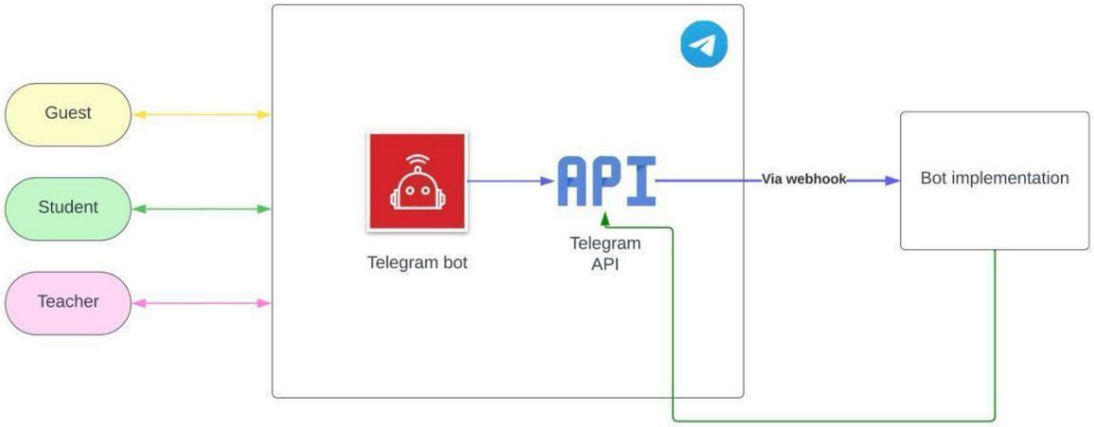

Рисунок 3 – Схема взаємодії бота та Telegram Api

Такий підхід відображає спрощення управління системою для користувачів через адміністративну частину та забезпечує зручну і безпечну комунікацію між учасниками освітнього процесу через бота у месенджері Telegram.

2.2 Порівняльний аналіз та обґрунтування вибору програмних засобів для реалізації

При обранні програмних засобів для розробки ІС чат-бота для комунікації в освітньому процесі, важливо враховувати кілька ключових аспектів: поширеність, продуктивність, витрати на інфраструктуру та підтримку. Для аналізу було

розглянуто кілька мов програмування та фреймворків, що відповідають цим критеріям [3] (див.рис.4).

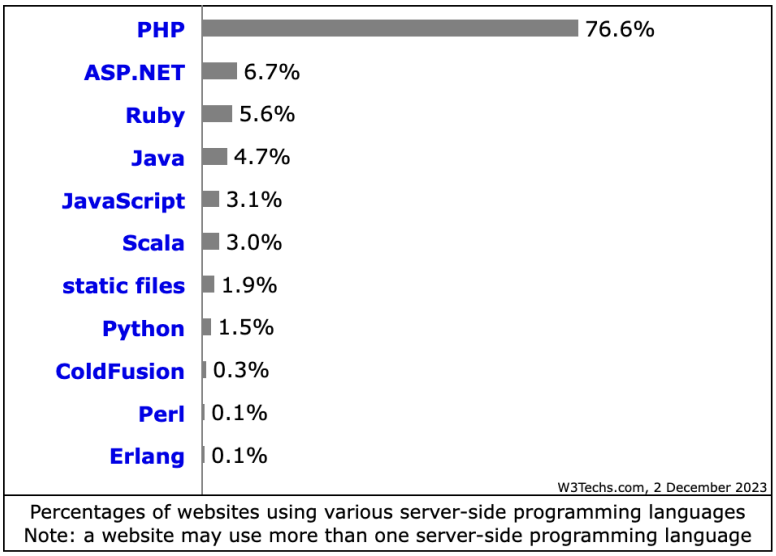

Рисунок 4 – Статистика використання мов програмування

PHP відомий своєю високою поширеністю в сфері веб-розробки. За даними статистики, понад 76% веб-застосунків використовують PHP. Однією з його переваг є широка підтримка, що означає, що багато хостинг-провайдерів мають підтримку PHP за замовчуванням, що полегшує розгортання ІС та застосунків, знижуючи витрати на інфраструктуру. Одним з ключових аргументів на користь PHP 8.2 є його нові можливості та поліпшена продуктивність. У порівнянні з попередніми версіями, PHP 8.2 пропонує значні вдосконалення, включаючи покращення у безпеці, нові функції та зменшення споживання ресурсів. Нижче наведені бенчмарки різних версій PHP [4] (див.рис.5).

PHP version	Math	String Manipulation	Loops	If / Else	Total Time (seconds)
8.2.12	4.30911	4.58159	4.76873	3.91580	17.57523
8.1.25	4.51500	5.25013	4.41980	3.72173	17.90665
8.0.30	4.29773	4.77057	4.42696	4.19197	17.68723
8.3.0RC5	4.67783	4.55844	4.77963	3.72777	17.74367
8.3.0RC4	4.25723	4.58364	4.76373	3.63546	17.24006
8.3.0RC3	4.22423	4.45863	4.73637	3.67527	17.09449
8.3.0RC2	4.24933	4.49621	4.74647	3.78586	17.27788
8.3.0RC1	5.07100	4.72439	4.73667	3.71169	18.24375
8.3.0beta2	4.39006	4.71944	4.75767	3.64146	17.50864
8.3.0beta1	4.26404	4.55889	4.73617	3.74243	17.30153
8.3.0alpha3	4.43824	4.62700	4.77983	3.69407	17.53914
8.3.0alpha2	4.55453	5.01261	5.44633	4.36264	19.37611
8.3.0alpha1	4.23871	4.50066	5.45287	3.80250	17.99474
7.4.33	4.21874	4.76173	4.60103	4.31053	17.89203
7.3.33	3.38197	4.69794	6.51654	2.25437	16.85081
7.2.34	3.76893	5.39654	6.81449	2.19043	18.17039
7.1.33	3.94130	6.72770	7.66927	2.98151	21.31978
7.0.33	3.87790	6.34081	7.50240	4.62316	22.34426
5.6.40	17.33584	16.18283	24.68934	9.62617	67.83417
5.5.38	18.81211	16.44000	33.37594	10.16464	78.79269
5.4.45	19.76127	17.15620	27.30743	10.39054	74.61544
5.3.29	22.25886	17.07307	36.07780	11.57677	86.98650

Рисунок 5 – Бенчмарки різних версій PHP

Попри це, PHP має свої обмеження. Наприклад, в порівнянні з іншими мовами, такими як Python чи Java, PHP може мати менші можливості управління пам'яттю, а також обмеженість у масштабуванні при створенні великих застосунків. Враховуючи багатофакторну природу вибору мови програмування, PHP 8.2 видається оптимальним вибором для реалізації цієї кваліфікаційної роботи, оскільки він поєднує в собі поширеність, нові можливості та низькі витрати на підтримку та інфраструктуру. Обираючи PHP для розробки ІС, що взаємодіє з Telegram Bot API для організації комунікації в освітньому процесі, важливо врахувати доступність готових рішень та компонентів, що спрощують розробку та підтримку системи. Використання PHP у поєднанні з Telegram Bot API надасть можливість використовувати готові компоненти для взаємодії з API Telegram. Наявність готових бібліотек та фреймворків для роботи з Telegram Bot API у середовищі PHP дозволяє зосередитися на функціоналі системи та швидше розробляти функції взаємодії з ботом у месенджері. Крім того, використання готового компонента для роботи з Telegram Bot API в PHP спрощує інтеграцію та забезпечує швидку та ефективну реалізацію функціональності бота. Це дозволить зосередитися на розробці унікальних функцій та взаємодії з учасниками освітнього

процесу, мінімізуючи час та ресурси, витрачені на основні рутинні завдання з роботою з API месенджера Telegram.

MySQL є однією з найпопулярніших систем управління базами даних (СКБД) і має ряд переваг, що роблять його відмінним вибором для реалізації ІС з безпечної комунікації між викладачами та студентами в освітньому процесі:

- надійність і стабільність (MySQL відомий своєю стабільністю та надійністю. Ця СКБД добре витримує навантаження і має високу продуктивність, що робить його чудовим вибором для проєктів з великим обсягом даних та великою кількістю запитів);
- ефективність (MySQL пропонує оптимізовані методи обробки даних, що дозволяє прискорити виконання запитів. Ця швидкодія дає змогу оптимізувати роботу системи та забезпечувати швидкий доступ до необхідної інформації);
- масштабованість (MySQL легко масштабується як вертикально (шляхом збільшення обсягу ресурсів на одній системі), так і горизонтально (шляхом розподілу даних на кілька серверів), що дозволяє пристосувати систему до потреб проєкту що зростають);
- спільнота та підтримка (багата спільнота користувачів та широкий ряд документації дозволяють швидко знаходити відповіді на питання та розв’язувати проблеми);
- безпека (MySQL надає різні механізми безпеки для захисту даних, включаючи можливість шифрування та управління доступом до бази даних).

Враховуючи ці фактори, MySQL обраний для розробки ІС, орієнтованої на забезпечення безпечної комунікації між викладачами та студентами. Його надійність, ефективність та можливість масштабування відповідають вимогам і забезпечують стабільну та швидку роботу системи у середовищі освітнього процесу.

Для реалізації чат-бота та адміністративної панелі обрано Laravel 10 версії, який використовує ORM Eloquent. Laravel – це потужний фреймворк PHP, який дозволяє швидко розробляти веб-застосунки, включаючи різні функції та компоненти, що полегшують роботу розробників. Нижче приведена порівняльна таблиця популярних фреймворків.

Таблиця 1 – Порівняльна таблиця фреймворків [8]

Framework	Laravel	CodeIgniter	Symfony	Laminas (Zend Framework)
Рік виходу	2011	2006	2005	2006
Архітектура	MVC	MVC	Full-stack framework	Component-based framework
Популярність	Дуже популярний	Популярний	Популярний	Помірно популярний
Навчальна крива	Для початківців	Для початківців	Складна	Помірна
Спільнота	Активна	Активна	Активна	Активна
Документація	Дуже добре	Дуже добре	Помірно	Помірно
ORM	Eloquent ORM	Built-in ORM (Active Record)	Doctrine ORM	Zend Db (Laminas ORM)
Маршрутизація	Система з широкими можливостями	Проста система	Гнучка система	Гнучка система
Шаблонізатор	Blade template engine	Custom templating engine	Twig template engine	Custom templating engine
Підтримка БД	MySQL, PostgreSQL, SQLite, etc.	Багато БД	Багато БД	Багато БД
Тестування	Вбудована підтримка	Обмежені можливості	Вбудована підтримка	Вбудована підтримка
Безпека	Комплексні функції безпеки	Базові функції безпеки	Комплексні функції безпеки	Комплексні функції безпеки
Розширення	Велика екосистема пакетів	Помірна кількість компонентів	Широкий асортимент пакетів і пакетів	Помірна кількість компонентів

Для управління залежностями використовується Composer – це менеджер залежностей у PHP. Він дозволяє встановлювати, оновлювати та керувати різними бібліотеками, що використовуються в проєкт. Щоб забезпечити функціональність бота, використовується готовий компонент від спільноти. Для дебагу та власної модифікації компонента використовується власний інструмент, розроблений окремо від цієї роботи. У процесі розробки на локальному середовищі використовується Sail, який забезпечує швидке налаштування локальної розробки за допомогою Docker контейнерів. Sail – це інструмент, який надає середовище для локальної розробки Laravel-застосунків, використовуючи Docker контейнери. Docker – це платформа для розробки, доставляння та запуску застосунків у контейнерах. Переваги використання Sail:

- простота налаштування (Sail забезпечує швидке налаштування локального середовища розробки. Завдяки заздалегідь сконфігурованим Docker контейнерам, розробники можуть швидко запустити проєкт на своєму комп'ютері без необхідності встановлення окремих сервісів (наприклад, вебсервер, СКБД тощо) вручну);
- ізоляція середовищ (Docker контейнери надають ізольоване середовище для розробки, що дозволяє запускати проєкти з різними конфігураціями середовища (наприклад, різні версії PHP або бази даних) без впливу на основну операційну систему);
- портативність (локальне середовище, налаштоване за допомогою Sail, може легко розгортатись на різних комп'ютерах розробників. Це дає можливість швидко розпочати роботу над проєкт на будь-якому комп'ютері без додаткового налаштування);
- універсальність (Docker забезпечує універсальність середовища, оскільки контейнери можна легко мігрувати між різними платформами та операційними системами, що полегшує роботу розробників);

– масштабованість (Docker контейнери можуть бути масштабовані для відповіді на змінні потреби проєкт, забезпечуючи гнучкість та пристосування до вимог розробки).

Sail забезпечує зручне середовище для локальної розробки Laravel застосунків, використовуючи потужні можливості Docker, що полегшує процес розробки та дозволяє швидко створювати, тестувати та розгортати проєкти без витрат часу на складні налаштування. Інфраструктурно проєкт складається зі СКБД MySQL для зберігання даних, вебсервер (nginx або apache) – для обробки та надання вебсторінок, а також інтерпретатора PHP (php-fpm) для виконання скриптів на сервері.

При виборі алгоритму шифрування для забезпечення безпеки даних у чат-боті слід розглянути кілька популярних варіантів, а саме AES (Advanced Encryption Standard), RSA (Rivest–Shamir–Adleman) та DES (Data Encryption Standard). Враховуючи інформацію з порівняльної таблиці найпопулярніших варіантів алгоритмів шифрування (див.табл.2), AES-256-CBC має суттєві переваги у безпеці, ефективності та сучасності.

Таблиця 2 – Порівняльна таблиця алгоритмів шифрування [9]

Параметр	AES-256-CBC	RSA	DES
Тип алгоритму	Симетричний	Асиметричний	Асиметричний
Довжина ключа	265 біт	1024-4096 біт	56 біт
Безпека	Висока	Висока	Низька
Швидкодія	Висока	Низька	Висока
Стійкість до атак	Висока	Висока	Низька
Стандарт	AES є стандартом NIST	Широко використовуваний	Застарілий
Застосування	Шифрування даних	Безпечна передача даних	Навчання та історичні цілі
Ефективність	Висока	Низька	Середня
Складність реалізації	Помірна	Висока	Низька

Advanced Encryption Standard (AES) – це симетричний алгоритм шифрування, який використовується для захисту конфіденційності даних. AES став стандартом у багатьох галузях через свою ефективність, безпеку та широке застосування. Основні особливості AES-256-CBC:

- ключ шифрування (AES-256 вказує на те, що використовується ключ завдовжки 256 бітів. Більша довжина ключа у порівнянні з меншими версіями AES (наприклад, AES-128) робить його більш стійким до криптоаналітичних атак);
- режим Cipher Block Chaining (CBC) (цей режим шифрування використовує попередній зашифрований блок для шифрування наступного блоку даних, що робить шифр більш стійким до атак на шифротекст. Він також дозволяє шифрувати однакові блоки даних по-різному, що ускладнює криптоаналіз);
- використання в сучасних системах (AES-256 є одним з найбільш популярних алгоритмів шифрування і використовується у багатьох сучасних системах для захисту конфіденційності даних, таких як збереження паролів, шифрування файлів та забезпечення безпеки під час передачі інформації);
- широкі можливості застосування (AES-256-CBC є стандартом безпеки у багатьох сферах, включаючи інформаційні технології, фінанси, медицину, військову та державну сфери, оскільки він забезпечує високий рівень захисту інформації).

Загалом, алгоритм шифрування AES-256-CBC є потужним та високонадійним алгоритмом шифрування, який забезпечує високий рівень безпеки для захисту даних. Також слід зазначити, що він є важким для розшифрування, як наслідок даний алгоритм вимагає великої кількості ресурсів для злому шифру. Використання цього алгоритму гарантує захист особистих даних учасників освітнього процесу в системі, тобто алгоритм сприятиме забезпеченню конфіденційності інформації та попередженню несанкціонованого доступу до даних користувачів.

3. ПРОЄКТУВАННЯ ЧАТ-БОТА

3.1 Проєктування backend частини чат-бота

Backend частина чат-бота відповідає за обробку запитів, зберігання даних та взаємодію з користувачами через Telegram. Основна мета полягає в забезпеченні надійної та безпечної обробки інформації, що надходить від користувачів.

Компонент Commander виступає основним контролером логіки взаємодії між користувачем та системою, що дозволяє боту розпізнавати отримані від користувача повідомлення та відповідно на них реагувати. Цей компонент відповідає за обробку вхідних даних, визначення потрібних дій та взаємодію з іншими складовими системи для ефективної роботи.

Використання фрагментів з Laravel у складі Commander дозволяє отримати доступ до функцій фреймворку для зберігання та обробки даних. Компонент використовує ORM Eloquent для роботи зі сховищем даних, що спрощує взаємодію з БД і дозволяє ефективно опрацьовувати запити, отримані від користувачів. Крім того, Commander відповідає за виконання різноманітних запитів та команд, які можуть включати створення, оновлення або видалення записів в БД, відправлення повідомлень користувачам, керування доступом та інші функції, необхідні для коректної роботи ІС.

Цей підхід дозволяє розділити логіку обробки запитів користувачів на модульні частини, які забезпечують більшу гнучкість та стабільність у розвитку системи. Компонент Commander виступає основним механізмом контролю та керування, об'єднуючи функціональні можливості Laravel для ефективного функціонування системи (див. рис. 6).

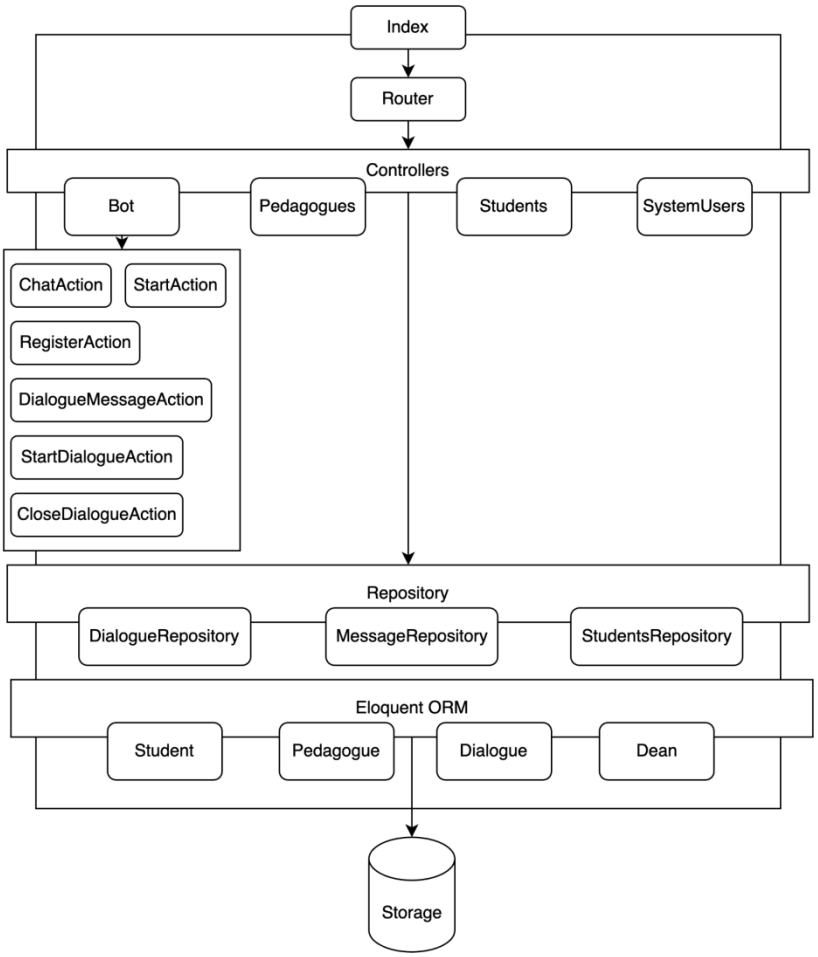

Рисунок 6 – Спрощена схема компонентів backend частини

Основна обробка запитів від бота здійснюється через Actions. Ці компоненти системи призначені для отримання та аналізу оновлень від Telegram API та виконання відповідних дій відповідно до потреб і можливостей ІС.

Коли Telegram API надсилає оновлення, Actions приймають ці дані, виконують їх перевірку на відповідність встановленим критеріям та визначають, які конкретно дії потрібно виконати. Якщо отримане оновлення не відповідає очікуванням системи або містить помилки, Actions здатні генерувати відповідну помилку для подальшої обробки (рис. 7).

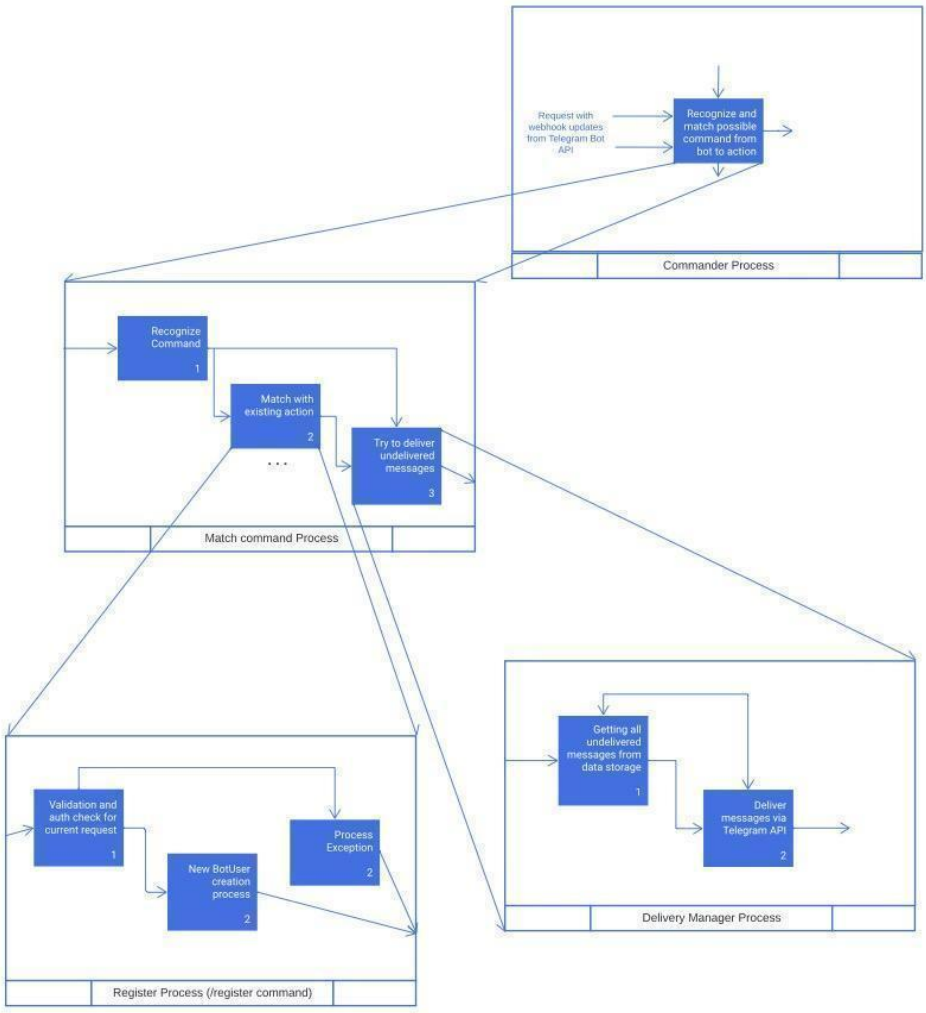

Рисунок 7 – IDEF0 діаграма компонента Commander

Коли Telegram API надсилає оновлення, Actions приймають ці дані, виконують їх перевірку на відповідність встановленим критеріям та визначають, які конкретно дії потрібно виконати. Якщо отримане оновлення не відповідає очікуванням системи або містить помилки, Actions здатні генерувати відповідну помилку для подальшої обробки. За допомогою Actions відбувається валідація та обробка отриманих від бота даних. Якщо вони відповідають умовам, встановле-

ним ІС, Actions виконують заплановану дію: це може бути створення нового повідомлення, збереження даних в БД, зміна статусу користувача чи інші визначені функції, які реалізовані в системі (див. рис. 8.)

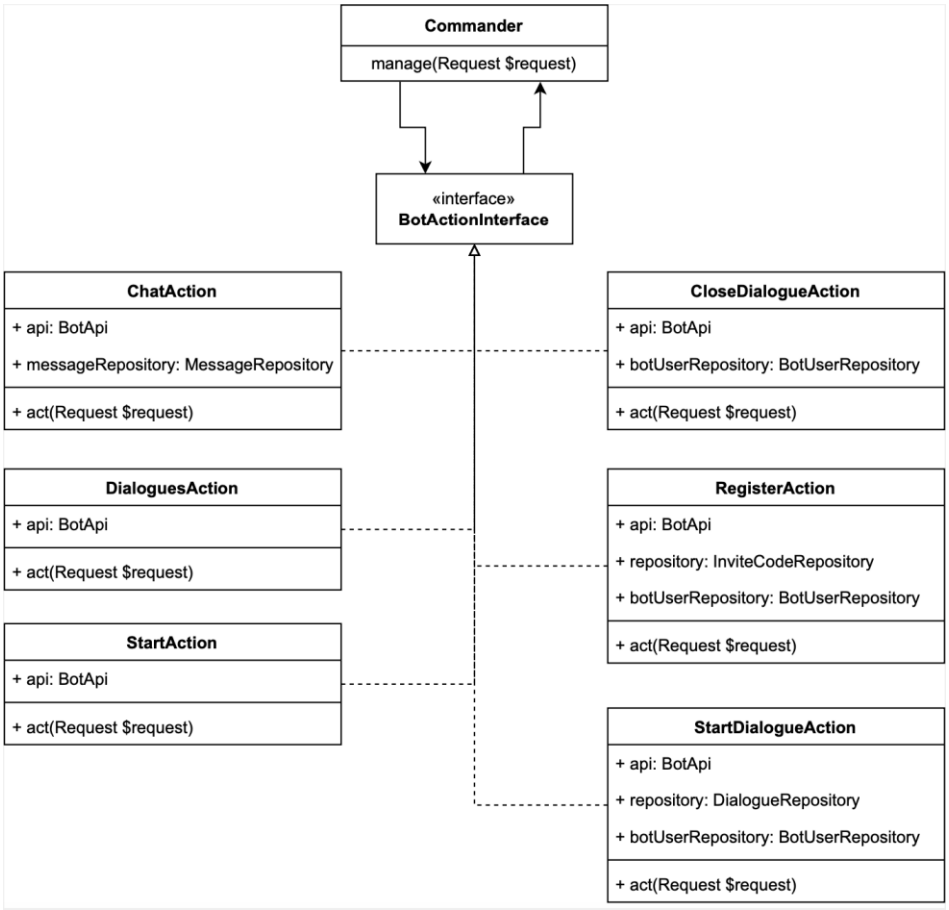

Рисунок 8 – Спрощена діаграма зв’язків компонента Commander

Такий підхід дозволяє розділити обробку запитів на окремі дії та забезпечити більшу гнучкість та чіткість в логікі програми, підвищуючи його ефективність та можливості управління. Actions виступають основними модулями, що забезпечують коректне та надійне виконання функцій боту в контексті ІС.

В рамкам реалізації адміністративної панелі (адмінпанель), у вигляді розробки окремої ІС, обрано використання таких технологій, як Laravel, Blade,

Sanctum та Tailwind CSS [6] для оформлення користувацького інтерфейсу. Основним архітектурним підходом є використання MVC. Дана модель представляє собою доступ до даних та виконує всі операції, пов'язані з БД. У контексті адміністративної панелі вона відповідає за взаємодію з БД, виконуючи завдання зберігання, оновлення та отримання необхідних даних.

Представлення (View) відповідає за відображення інформації користувачу через графічний інтерфейс. Використання Blade, яке є шаблонізатором Laravel, дозволяє створювати динамічні та ефективні шаблони для побудови інтерфейсу користувача адмінпанелі.

Контролер (Controller) відповідає за обробку вхідних запитів, взаємодію з моделлю для отримання необхідних даних та передачу цих даних у відповідні представлення для відображення користувачеві. Завдяки цій структурі, відбувається логічне розподілення функціонала та легше втручання в окремі компоненти системи для подальшого розширення чи вдосконалення.

Для забезпечення аутентифікації та авторизації використовуватиметься Sanctum, що в свою чергу забезпечить безпеку управління доступом до різних частин системи. Інтерфейс користувача адмінпанелі буде розроблений з використанням Tailwind CSS, це дозволить швидко та гнучко налаштовувати вигляд та стиль UI-компонентів.

Даний підхід до проєктування адміністративної панелі з використанням MVC, Laravel, Sanctum та Tailwind CSS забезпечує зручність, легкість розширення та високий рівень безпеки в контексті цієї ІС. Під час розробки backend частини цього проєкту, особлива увага приділяється дотриманню стандартів PHP Standards Recommendations (PSR). PSR [7] – це рекомендації, які визначають узгоджений стиль написання коду, структуру директорій, автозавантаження класів та інші аспекти програмування на мові PHP.

Дотримання стандартів PSR має декілька переваг:

- стандартизація коду (PSR визначає загальні правила та стилі коду для PHP-проєктів. Це полегшує співпрацю великих команд розробників, адже всі вони використовують однаковий стиль коду);
- зрозумілість та читабельність (стандарти PSR сприяють створенню коду, який є більш зрозумілим та легко читається для інших розробників. Це важливо для підтримки та розвитку системи в майбутньому);
- сумісність з бібліотеками та фреймворками (багато бібліотек та фреймворків у своєму функціоналі використовують стандарти PSR. Це дозволяє з легкістю інтегрувати ці рішення в проєкт та використовувати їх спільно з іншими компонентами системи);
- підтримка спільноти (стандарти PSR розроблені та підтримуються PHP-спільнотою. Їх використання сприяє покращенню якості коду та сприяє активному обміну досвідом між розробниками).

Взаємодія з кодом відбуватиметься через систему контролю версій Git, а також сервіс хостингу коду GitHub для зберігання кодової бази. Проєкт планується розмістити у приватному репозиторії, що забезпечить конфіденційність коду та контроль доступу до нього. Використання Git є критично важливим в контексті розробки програмного забезпечення з наступних причин:

- контроль версій (Git дозволяє зберігати історію змін в коді, що дозволяє відстежувати та відновлювати попередні версії коду в разі потреби);
- колаборація та командна робота (Git спрощує спільну роботу над проєктом. Він дозволяє кільком розробникам одночасно працювати над різними гілками коду та об'єднувати зміни);
- інтеграція та розгортання (використання GitHub дозволяє налаштовувати автоматизовані процеси розгортання за допомогою GitHub

Actions. Це дозволить автоматизувати процеси після злиття гілки в основну гілку (main), наприклад – автоматичний деплой коду на сервер);

- зберігання та резервне копіювання (GitHub забезпечує безпечне зберігання коду в хмарному сервісі, що гарантує його доступність та захищеність від втрати).

Використання Git та GitHub з приватним репозиторієм та налаштованими діями для автоматизації розгортання є важливим елементом у розробці ІС. Це забезпечує безпеку, ефективність та швидкість розробки, дозволяючи команді розробників працювати спільно та ефективно над проєктом.

3.2 Проєктування функціональних вимог та моделювання бази даних

Проєктування БД для цього проєкту є важливим етапом, оскільки це формує основу для зберігання та управління даними, необхідними для взаємодії учасників освітнього процесу. БД повинна ефективно забезпечити зберігання та обробку інформації про студентів, викладачів, групи, кафедри, повідомлення та інші аспекти взаємодії учасників освітнього процесу в рамках проєкту що імплементується. База даних повинна включати такі сутності:

- студенти та викладачі (зберігання основної інформації про учасників освітнього процесу, таку як ім'я, прізвище, електронна адреса, контактні дані, номер залікової книжки тощо);
- деканати та кафедри (інформація про деканати студентів та відповідно – кафедри з викладачами, для організації та категоризації учасників освітнього процесу);
- повідомлення (зберігання текстових повідомлень між учасниками освітнього процесу, які будуть взаємодіяти через чат-бот);

- адміністративна інформація (дані, необхідні для реалізації адміністративних функцій, такі як керування користувачами, створення нових діалогів тощо);
- сутності для аутентифікації та авторизації користувачів (забезпечення безпеки та захисту доступу до системи).

Проєктування БД має на меті створити структуру, яка враховує потреби у зберіганні та маніпулюванні даними в рамках проєкту, забезпечуючи ефективну та безпечну роботу із даними користувачів учасників освітнього процесу.

Таблиця dean_offices необхідна для зберігання інформації про деканати, до яких відносяться студенти (таблиця students), таблиця departments містить інформації про кафедри, до яких відносяться педагоги (таблиця pedagogues). В системі для адміністрування кафедрами та деканатами, а також студентами та викладачами є безпосередньо адміністратори кафедр та адміністратори деканатів (таблиця users) вони пов’язані зовнішнім ключем administrator_id (який є і в dean_offices, і в departments). Викладачі та студенти пов’язані додатковою таблицею, для надання можливості відправляти повідомлення, фактично це таблиця з діалогами яка має назву students_to_pedagogues, що містить foreign keys для зв’язку з таблицями students та pedagogues. Всі повідомлення між учасниками освітнього процесу зберігаються в зашифрованому вигляді у таблиці messages, яка пов’язана зі students_to_pedagogues.

Структура бази даних виглядає наступним чином:

– users (id (primary), name, email, email_verified_at, password, remember_token, created_at, updated_at, role);
– dean_offices (id (primary), name, administrator_id, created_at, updated_at);
– departments (id (primary), name, administrator_id, created_at, updated_at);

- students (uuid (primary), full_name, chat_id, username, gradebook_number, active_dialogue_id, is_blocked, dean_office_id, created_at, updated_at);
- pedagogues (uuid (primary), full_name, chat_id, username, active_dialogue_id, is_blocked, phone, department_id, created_at, updated_at);
- students_to_pedagogues (id (primary), pedagogue_uuid, student_uuid, blocked);
- messages (uuid (primary), data, is_delivered, created_at, updated_at, dialogue_id);
- migrations (id (primary), migration, batch).

Реляції між таблицями:

- dean_offices може мати декількох адміністраторів (users);
- departments може мати декількох адміністраторів (users);
- students може належати тільки до одного dean_office;
- pedagogue може належати тільки до одного deparments;
- один students може мати безліч діалогів з pedagogues і навпаки;
- кожен запис students_to_pedagogues має багато messages, але тільки 1 message може мати тільки 1 students_to_pedagoues.

Такий підхід дозволить забезпечити необхідну структуру для зберігання та організації даних, забезпечуючи функціональність системи та взаємодію між її основними складовими. На основі викладених таблиць та їх зв'язків, можна зазначити кілька переваг такої структури бази даних для ІС:

- ефективність зберігання даних: структура бази даних дозволяє ефективно зберігати інформацію про користувачів, діалоги, повідомлення та інші аспекти участі в освітньому процесі, забезпечуючи оптимальне використання простору пам'яті;

– гнучкість і зручність управління користувачами;
– забезпечення конфіденційності та безпеки: застосування шифрування даних у полях дозволяє зберігати чутливу інформацію в зашифрованому вигляді, що забезпечує більшу безпеку в системі;
– організація комунікації: структура таблиць students_to_pedagogues та messages дозволяє зберігати діалоги та повідомлення між викладачами та студентами, роблячи систему гнучкою та спрощеною для комунікації.

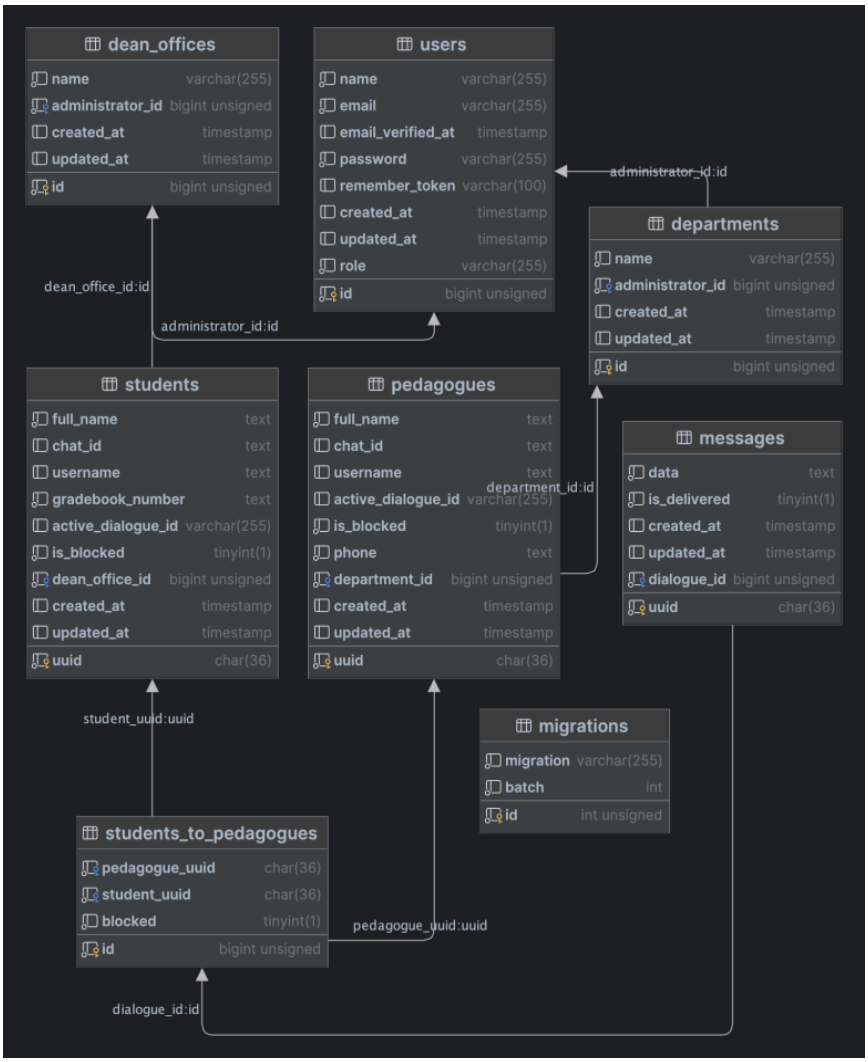

Рисунок 9 – Структура БД

Ця структура бази даних (рис. 9) відображає основні аспекти комунікації в освітньому процесі, забезпечуючи потрібний функціонал для взаємодії між учасниками та збереження необхідної інформації. Запропонована структура бази даних буде імплементована в MySQL, одній з найпоширеніших та надійних систем управління базами даних. Обрана система дозволить зберігати, організовувати та управляти інформацією, забезпечуючи надійність, ефективність та швидкість доступу до даних. MySQL відома своєю широкою підтримкою, гнучкістю у конфігурації та швидкими операціями з базою даних. Її можливості щодо оптимізації та прискорення роботи з даними зроблять ІС ефективною та стабільною у використанні. Крім того, обрана база даних відповідає сучасним стандартам і буде легко інтегрована з іншими компонентами системи.

4. РЕАЛІЗАЦІЯ ЧАТ-БОТА

Реалізація чат-бота включатиме кілька послідовних етапів, кожен з яких спрямований на розвиток певного функціонала та компонентів системи. Початковим кроком є створення адміністративної панелі для керування системою та взаємодії з ключовими сутностями, такими як викладачі, студенти, кафедри, деканати та діалоги. Наступним етапом є інтеграція бота в середовище Telegram, при цьому забезпечивши його функціонал безпечним для комунікації між учасниками освітнього процесу. Далі етап реалізації логіки реєстрації та автентифікації через чат-бот для студентів і викладачів, забезпечуючи захищений доступ до системи. Завершальним етапом є імплементація функціонала – надсилання повідомлень та управління доступними діалогами. Кожен з цих етапів має ключове значення для розвитку та функціональності чат-бота та ІС, забезпечуючи її повноцінне функціонування та безпеку у взаємодії з учасниками освітнього процесу.

4.1 Розробка панелі керування

Адміністративна панель є ключовим компонентом ІС, призначеною для управління та контролю за всіма аспектами системи. Розробка адмінпанелі базується на фреймворку Laravel з використанням шаблонів Blade для UI та Sanctum. Ця панель забезпечує управління користувачами – викладачами та студентами, керування кафедрами, деканатами, а також моніторинг активності та ведення журналу подій. Вона використовує стандартну архітектуру MVC, що надає структурованість коду, зручність у розробці та підтримці.

Sanctum використовується для забезпечення аутентифікації та авторизації користувачів, забезпечуючи захист доступу до конфіденційних даних та функцій системи (рис. 10).

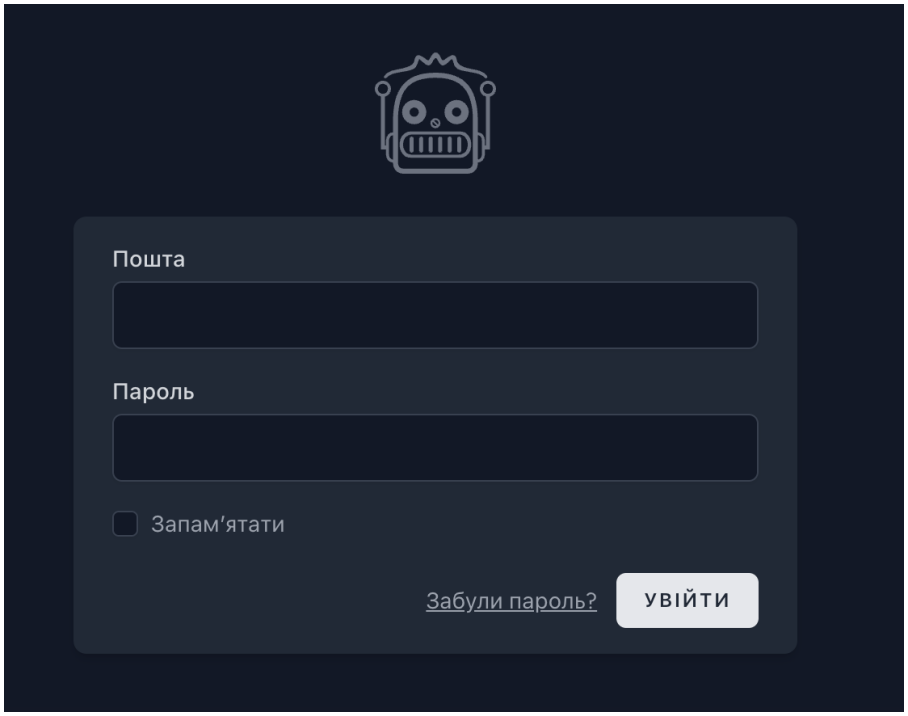

Рисунок 10 – Форма входу в адмінпанель

Розробка адміністративної панелі передбачає створення 5 контролерів, кожен з яких відповідає за різні аспекти управління та взаємодії з основними моделями системи. Нижче наведено короткий огляд цих контролерів та їх призначення:

- AdministratorController надає можливість створювати адміністраторів (див. рис. 11);
- DialoguesController керує діалогами між викладачами та студентами, забезпечуючи можливість переглядати, створювати, редагувати та видаляти діалоги;
- DeanOfficesController надає можливість створювати деканати, та прив'язувати до них адміністраторів;
- DepartmentsController надає можливість створювати кафедри, та прив'язувати до них адміністраторів;

– StudentsController надає можливість керувати студентами;
– PedagoguesController надає можливість керувати викладачами.

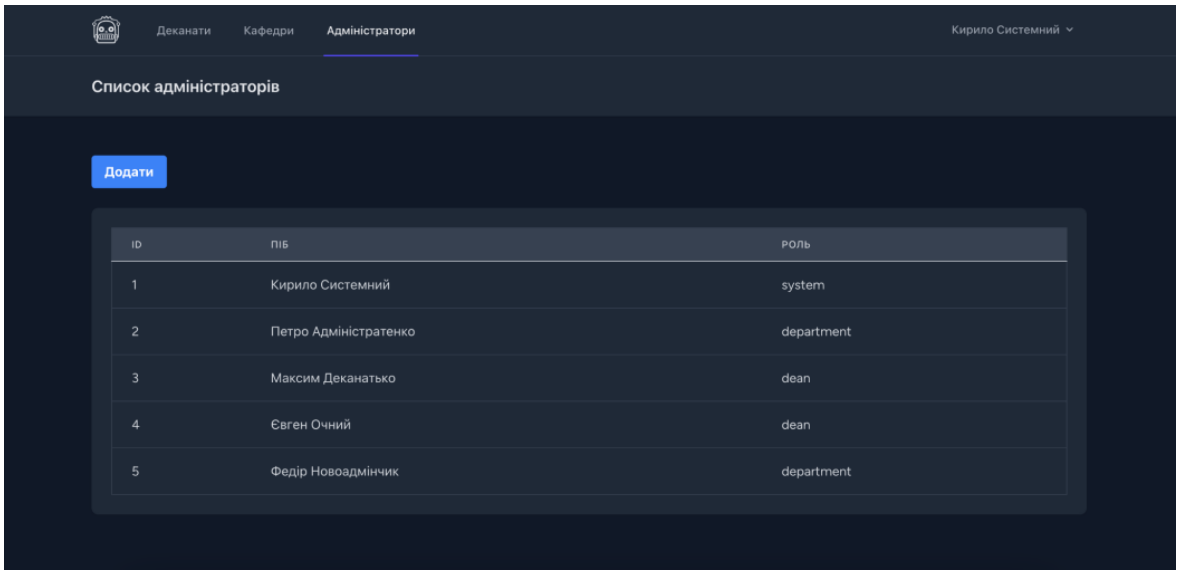

Рисунок 11 – Сторінка зі списком адміністраторів

Кожен з цих контролерів взаємодіє з відповідними моделями, які відображені в БД. Таким чином імплементовано моделі User, DeanOffice, Department, Student та Pedagogue, Message, Dialogue, які забезпечують основу для зберігання та операцій з даними. Робота з цими моделями відбувається завдяки міграціям, які визначають структуру БД. Для кожного з цих контролерів створено відповідні представлення (views), які дозволяють відобразити дані та взаємодіяти з ними через веб-інтерфейс. Ці представлення відповідають за відображення списків, форм редагування, перегляду та видалення об'єктів кожної моделі.

У файлі web.php налаштовано маршрути (routes) для кожного методу контролера, що визначить шлях доступу до кожного дії у системі. Це дозволяє користувачам здійснювати різноманітні операції через веб-інтерфейс за допомогою відповідних URL-адрес. Таким чином адміністративна панель забезпечує зручне

управління ключовими сутностями системи та їх взаємодію через інтуїтивно зрозумілий веб-інтерфейс.

В процесі розробки а саме, для генерації основних складових системи використовувався інтерфейс командного рядка Laravel – Artisan. Зокрема, використовувалося створення моделей та міграцій для БД, що дозволило швидко та зручно описати структуру даних і перенести її в БД.

Слід також зазначити, що нащадки класу Request в Laravel використовуються для валідації даних, які надходять від форм адмінпанелі. Це дає можливість перевірити вхідні дані ще до їх подальшої обробки, це в свою чергу забезпечує коректність та безпеку інформації.

Крім того варто відзначити, що у поточній версії імплементації ІС для цієї роботи не передбачено реєстрації в адмінпанелі для більшості користувачів. Це означає, що доступ до адміністративних можливостей обмежений кількістю користувачів (наприклад, тільки один адміністратор в поточній версії системи).

4.2 Розробка та налаштування чат-бота

Розробка чат-бота починається з етапу налаштування його основних параметрів через Telegram Bot API. Для цього потрібно створити чат-бота з використанням спеціального бота Telegram під назвою BotFather. Створення бота згідно з документацією [2] Telegram:

- першим кроком є створення бота через BotFather, відправивши команду `/newbot`;
- отриманий API-ключ (token) використовується для взаємодії з Telegram API;
- за допомогою BotFather також налаштовано привітання, опис, та аватарку для бота (див. рис. 12).

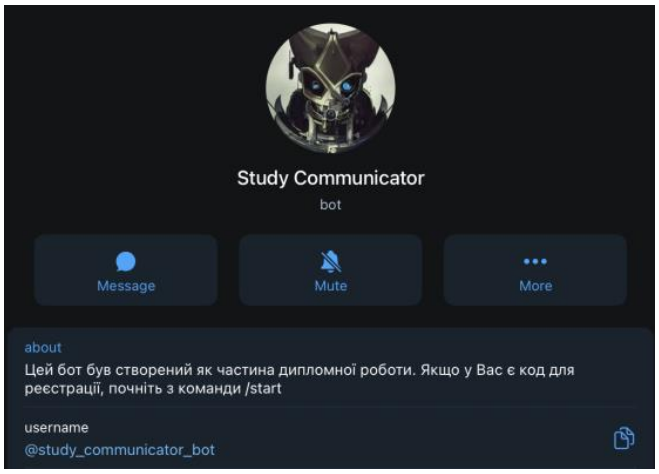

Рисунок 12 – Вигляд налаштованої інформації

Розробка бота на PHP з використанням компонента telegram-bot/api має свої особливості. Запит, що прийшов на визначений route обробляється відповідним контролером, після чого запит передається в службу Commander, де відбувається аналіз тексту повідомлення користувача. У цьому контексті розглядаються дві основні команди: /start та /gradebook <number>.

Команда /start надає користувачеві інформацію щодо можливостей бота. В контексті даних ця команда відповідає за відображення тексту з інформацією про реєстрацію або перелік доступних команд для користувача (див. рис. 13).

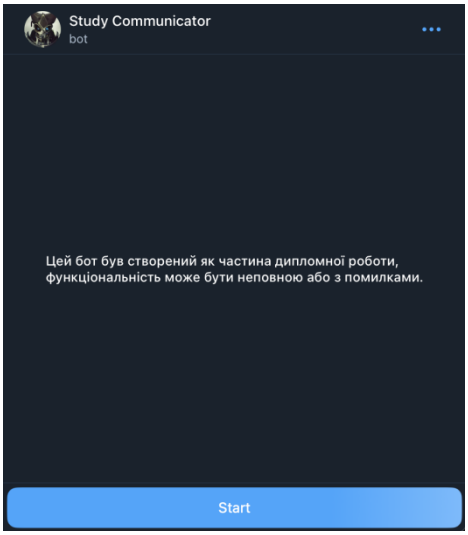

Рисунок 13 – Початкове повідомлення в чаті з ботом

Команда /start, перевіряє чи вже авторизувався користувач і якщо «ні» – просить користувача авторизуватись, в іншому випадку, – надсилає повідомлення про успішну авторизацію та виводить список доступних команд (див. рис. 14). Далі наведено код який відповідає за цю команду:

```
public function act(Request $request): void
{
      $message = "Доброго дня!\n\nПеред початком, " .
        "будь ласка, авторизуйтесь:\n\n- Для викладачів /phone\n- Для студентів /gradebook
<number>";
    if ($request->user !== null) {
       $username = Crypt::decrypt($request->user->full_name);
    $message = "Доброго дня, $username! 🙂 Ви успішно авторизувалися, ось доступні ко-
манди:\n\n/dialogues";
        }
        $this->api->sendMessage($request->get('message')['chat']['id'], $message);
    }
```

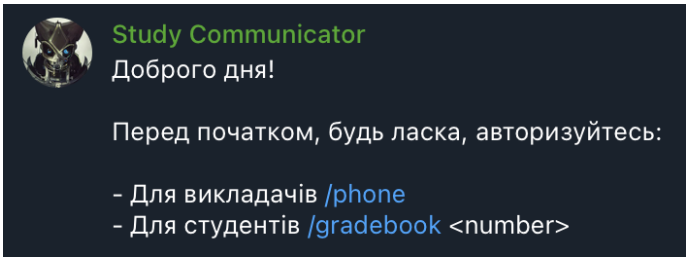

Рисунок 14 – Знімок з екрана чату з ботом /start

Команда /gradebook <number> реєструє студента за номером облікової книжки. RegisterAction перевіряє переданий номер код на валідність. Якщо код не існує в базі даних або вже використовувався, користувач отримає повідомлення про помилку. У разі успішної перевірки користувач в БД оновиться, авторизується, і в telegram йому приходить сповіщення про успішну реєстрацію разом зі списком доступних команд.

Всі ці етапи реалізовані в сервісі Commander, що виконує розпізнавання та обробку повідомлень від користувачів бота. Команди /start та /gradebook <code> визначають відповідні виклики та перехоплюються для подальшої обробки.

4.3 Реалізація системи реєстрації та автентифікації

Вже розроблені /start та /gradebook команди покликані спростити процес взаємодії з ботом, але не враховують випадки, коли користувач вже зареєстрований. Для подальшої реалізації імплементується Middleware, який використовується для автентифікації користувача при кожному запиті до бота. В ньому здійснюється перевірка інформації, що надходить в request від Telegram API, таких як username та chat_id. Намагаючись знайти відповідний запис у базі даних за цими даними, виконується пошук моделі Student. Якщо користувач знайдений, він додається до параметра user у request як модель Student, в іншому випадку

додається значення null. Тепер у кожному Action для команд можна перевірити наявність автентифікації через перевірку $request->user === null. Для викладачів працює така сама логіка, з однією різницею під час реєстрації – від них очікується номер телефону.

Оновлення логіки для команд /start, /phone та /gradebook, де імплементовано додаткові перевірки, наприклад якщо користувач не null, це означає, що він вже автентифікований і додаткова реєстрація недоступна. Навпаки, якщо хтось намагається пройти автентифікацію під вже зареєстрованим користувачем для іншого chat_id, цей запит блокується. Цей підхід дозволяє контролювати доступ до функцій бота, забезпечуючи автентифікацію користувачів лише один раз та уникнення можливих спроб маніпуляцій або незаконного доступу до функціонала бота. Технічно це реалізується шляхом залучення Middleware для обробки кожного запиту до бота, в якому проводиться перевірка наявності користувача в базі за вказаними даними, що надходять у request від Telegram API. Оновлені функції команд враховують ці обмеження для кращої безпеки та користувацького досвіду. Нижче наведена імплементація Middleware який перевіряє чи авторизувався користувач за його username який надає telegram api під час будь-якого запиту до webhook. Як видно з методу, перевірка здійснюється окремо як для викладачів, так і для студентів, але глобально, використовується один тип сутності – BotUser.

```
public function handle(Request $request, Closure $next): Response{
        $pedagogue = null;
        $pedagogues = Pedagogue::whereNotNull('username')->get();
        foreach ($pedagogues as $pg) {
            if(Crypt::decrypt($pg->username ?? '') === $request-
>get('message')['from']['username']) {
                $pedagogue = $pg;
                break;
            }
        }
        if ($pedagogue !== null) {
            $request->merge(['user' => $pedagogue]);
        }
```

```
        $student = null;
        $students = Student::whereNotNull('username')->get();
        foreach ($students as $st) {
            if(Crypt::decrypt($st->username ?? '') === $request-
>get('message')['from']['username']) {
                $student = $st;
                break;
            }
        }
        if ($student !== null) {
            $request->merge(['user' => $student]);
        }
        return $next($request);
}
```

4.4 Розробка механізмів комунікації та управління діалогами

Відповідно до попереднього функціонала, наступним кроком в реалізації бота є імплементація функцій надсилання повідомлень та керування доступними діалогами. Команда /dialogues (DialoguesAction) використовується для вибору доступних діалогів та має наступний функціонал:

- перевірка користувача (перевіряється, чи користувач не має активний діалог (active_dialogue_id = null));
- вибір доступних діалогів (якщо у користувача немає активного діалогу, бот відправляє повідомлення з клавіатурою вибору діалогів);
- створення клавіатури (створюється клавіатура з кнопками для кожного доступного діалогу. Кожна кнопка містить інформацію про діалог (його ідентифікатор та ім'я співрозмовника));
- відправлення повідомлення (після створення клавіатури з діалогами, бот відправляє повідомлення з клавіатурою користувачеві).

Ця команда дозволяє користувачу обирати доступні діалоги для подальшого спілкування з вибраним співрозмовником. У випадку, якщо користувач вже

має активний діалог, команда не виконує жодних дій та не відправляє повідомлення. Нижче наведено код класу DialoguesAction:

```
public function act(Request $request): void  {
        /** @var BotUser|null $user */
        $user = $request->user;
        if ($user !== null && $user->active_dialogue_id === null) {
            $message = "↪ Будь ласка, оберіть діалог ";
            $availableDialogues = $user->dialogues();
            $buttons = [];
            /** @var Dialogue $dialogue */
            foreach ($availableDialogues as $dialogue) {
                if ($user->uuid === $dialogue->student_uuid) {
                    $receiver = 'pedagogue_uuid';
                } else {
                    $receiver = 'student_uuid';
                }
                $buttons[] = ['/d ' . $dialogue->id . ' ' . Crypt::decrypt($dialogue-
>actor($receiver)->full_name)];
            }
            $keyboard = new ReplyKeyboardMarkup( $buttons );
            $this->api->sendMessage(chatId: $user->getChatId(), text: $message,
replyMarkup: $keyboard);
        }
    }
```

Додатково, враховано можливість виходу користувача з діалогу, а також опрацювання повідомлень для користувача, який вийшов з діалогу. До основних кроків на цьому етапі можна віднести:

- обрання доступних діалогів (можливість обрати діалог для подальшої комунікації між користувачами. Включає обробку команди /d <dialogue id>, де користувач може обрати діалог з доступних опцій);
- надсилання повідомлень (можливість відправлення повідомлень у вибраний діалог. Тобто логіку, яка дозволяє користувачу вводити текст повідомлення та надсилати його обраному співрозмовнику у вибраному діалозі);

- управління виходом з діалогу (функціонал виходу користувача з діалогу /e. Користувач має можливість вийти з поточного діалогу, щоб припинити спілкування з обраним співрозмовником);
- обробка повідомлень для користувача, який вийшов (функціонал який обробляє повідомлення для користувача, який вийшов з діалогу. Він має можливість отримувати повідомлення, які були надіслані йому після виходу з діалогу);
- імплементація сервісу надсилання повідомлень (DeliveryManager) (створення сервісу, який відповідає за надсилання повідомлень у вибрані діалоги).

Всі ці кроки реалізовані відповідно до попереднього плану роботи, з урахуванням розширення функціонала бота для забезпечення зручної та безпечної комунікації між учасниками освітнього процесу. Команда /d використовується для вибору певного діалогу, обробляється в StartDialogueAction і має наступний функціонал:

- отримання інформації (пошук в текстовому повідомленні виклику команди (/d) цифрового ідентифікатора діалогу, що передається в самій команді);
- перевірка користувача (перевірка, чи користувач вже знаходиться в активному діалозі, тобто чи має він прив'язаний активний ідентифікатор діалогу (active_dialogue_id) у своєму профілі);
- вибір діалогу (якщо користувач не перебуває в активному діалозі (active_dialogue_id = null) і в тексті команди /d знайдено ідентифікатор діалогу, відбувається спроба знайти цей конкретний діалог за його ID);
- зміна активного діалогу (якщо діалог знайдено і користувач не перебуває в активному діалозі, активний ідентифікатор діалогу змінюється на ID обраного діалогу);

- відправлення повідомлень (після успішної зміни активного діалогу, система відправляє повідомлення обраному співрозмовнику в зазначений діалог, повідомляючи, що користувач приєднався до діалогу);
- зворотний зв'язок (після виконання вищезазначених операцій, бот відправляє повідомлення користувачу з підтвердженням результату вибору діалогу).

Даний механізм використовується для вибору діалогу задля подальшої комунікації між викладачем та студентом та підтвердження цієї дії. Початок діалогу відбувається на основі його ідентифікатора який надсилає користувач в команду /d <num>, де <num> ідентифікатор. Перед тим як користувач доєднається до діалогу, відбувається перевірка, чи дійсно цей користувач має доступ до цього діалогу і є його учасником. Нижче наведено імплементацію метода act команди StartDialogue.

```
public function act(Request $request): void {
        $user = $request->user;
        $text = $request->get('message')['text'];
        if (preg_match('/\/d\s(\d+)/', $text, $matches) && $user !== null) {
            $dialogueId = str_replace('/d ', '', $matches[0]);
            /** @var Access|null $concreteDialogue */
            $concreteDialogue = Access::find((int)$dialogueId);
       if ($concreteDialogue === null || $user->active_dialogue_id !== null) {
          $message = '↪ Йой... щось не так, спробуйте знову ';
       } else {
                $user->active_dialogue_id = $concreteDialogue->id;
                if ($user->save()) {
  $message = '↪ Ви успішно увійшли в діалог. Для виходу використовуйте команду /e ';
  $actorRole = $user instanceof Student ? 'pedagogue_uuid' : 'student_uuid';
  $receiver = $concreteDialogue->actor($actorRole);
   if ($receiver->active_dialogue_id !== null) {
      $this->api->sendMessage($receiver->getChatId(), '↪ ' . $user->getName() . '
доєдна(вся)лася до діалогу');
   }
                }
            }
        }
```

```
        $this->api->sendMessage($request->user->getChatId(), $message ?? '↪ Йой...
щось не так, спробуйте знову ', new ReplyKeyboardRemove(true));
}
```

Команда /e (CloseDialogueAction) використовується для виходу з активного діалогу і має наступний функціонал:

- перевірка користувача (перевіряється, чи користувач має активний діалог (active_dialogue_id != null));
- закриття діалогу (якщо користувач має активний діалог, то йому автоматично призначається null в активному діалозі. Це означає, що користувач завершує спілкування в діалозі);
- відправлення повідомлення (після успішного виходу з діалогу, система відправляє повідомлення співрозмовнику, що діалог був закритий і надсилатиме повідомлення тільки після того, як супутник повернеться до діалогу);
- зворотній зв'язок (після виконання вищезазначених операцій, бот надсилає повідомлення користувачу з підтвердженням результату виходу з діалогу).

Основна функція цієї команди – вихід з діалогу користувача (деактивація діалогу з конкретним користувачем) і повернення можливості використовувати інші команди бота. Нижче наведено фрагмент імплементації класу CloseDialogueAction, а саме частина з перевіркою на активний діалог і закриттям діалогу:

```
if ($user !== null && $user->active_dialogue_id !== null) {
       $id = $user->active_dialogue_id;
       $dialogue = $user->getActiveDialogue();
       $user->active_dialogue_id = null;
       if ($user->save()) {
              $actorRole = $user instanceof Student ? 'pedagogue_uuid' :
       'student_uuid';
              $receiver = $dialogue->actor($actorRole);
```

```
                    $message = '↪Успішно вийшли з діалогу #' . $id . ', Ви можете завжди
            повернутися в діалог ';
            if ($receiver->active_dialogue_id !== null) {
            $this->api->sendMessage($receiver->getChatId(), '↪А най його... ' . $user-
>getName() . ' закри(ла)в діалог. '. 'Повідомлення будуть доставлені одразу після того
співрозмовник повернеться до діалогу '
                    );
                }
            }
        }
```

У випадку помилок, таких як неможливість знайти активний діалог для закриття або проблеми з оновленням даних користувача, бот повідомляє користувача про непередбачені ситуації та прохання спробувати знову. `DeliveryManager` відповідає за спробу доставлення невідправлених повідомлень збережених у сховищі. Нижче описано його функціонал:

- отримання невідправлених повідомлень (отримує невідправлені повідомлення зі сховища за допомогою методу `findWhere()`);
- ітерація через кожне невідправлене повідомлення (отримання отримувача (`$receiver`) повідомлення та перевірка наявності активного діалогу між отримувачем та автором повідомлення);
- відправлення повідомлення (якщо у відправника та отримувача є активний діалог, `DeliveryManager` намагається надіслати повідомлення на основі його типу (текст, стікер, документ, фото, голосове));
- оновлення статусу доставлення (після намагання доставити повідомлення, встановлюється прапорець `is_delivered` для позначення успішної спроби доставки. Це запобігає подвійному доставленню повідомлення).

DeliveryManager намагається доставити невідправлені повідомлення між користувачами, які мають активний діалог і необхідний контент для надсилання.

Нижче наведено схему, що зображає загальну схему обміну повідомлень між учасниками освітнього процесу та шифруванням.

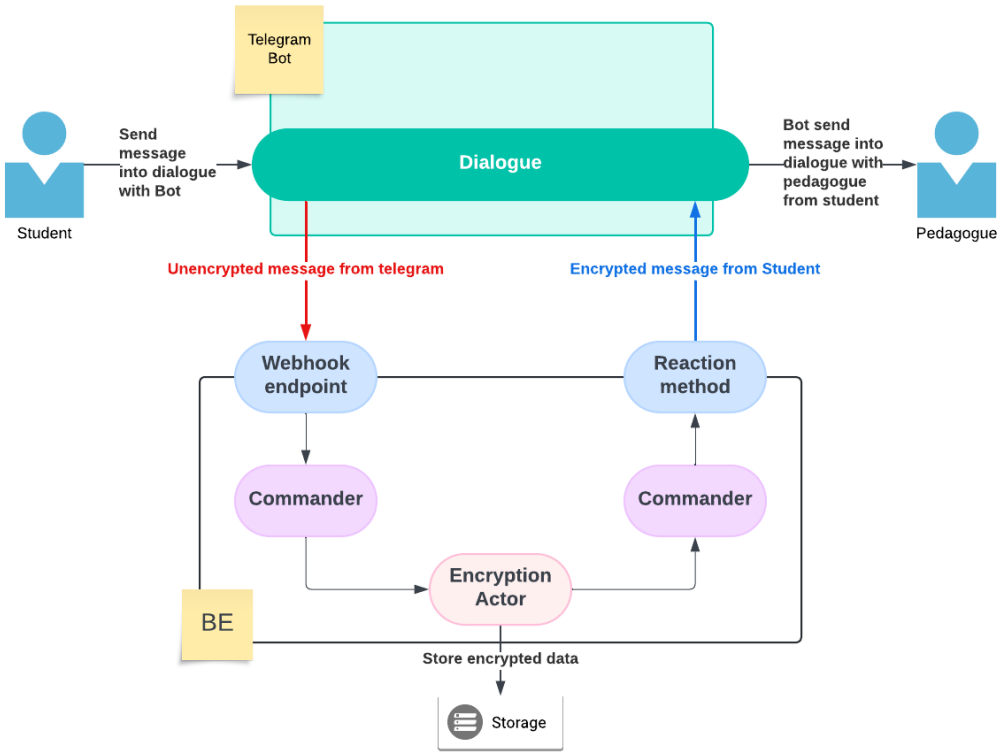

Рисунок 15 – Спрощена схема шифрування повідомлень між користувачами

ВИСНОВКИ

Розроблений чат-бот для взаємодії між учасниками освітнього процесу відповідає основній меті роботи, а саме забезпеченню захисту особистих даних користувачів. Система чат-бота підтримує реєстрацію та автентифікацію користувачів, що дозволяє ефективно відстежувати й аутентифікувати учасників. Особлива увага була приділена валідації запитів та перевірці дозволів доступу до даних, забезпечуючи, що особиста інформація користувачів залишається конфіденційною. Контроль доступу до діалогів, надсилання повідомлень, а також виведення списку доступних команд реалізовані з урахуванням безпеки. До цього додано механізми, що унеможливлюють несанкціонований доступ та захищають особисті дані в ході спілкування.

В цілому, розроблена система чат-бота успішно розв'язує проблему захисту особистих даних учасників освітнього процесу, надаючи зручний та безпечний інструмент для взаємодії та обміну інформацією. Реалізація різноманітних функцій, таких як реєстрація, автентифікація, вибір діалогів та надсилання повідомлень є підтвердженням того, що система чат-бота успішно впроваджує інструменти для захисту конфіденційності даних учасників освітнього процесу. Всі ключові функції були ретельно розроблені з урахуванням забезпечення безпеки, цілісності та доступності особистих даних. Цей проєкт відповідає актуальним вимогам до захисту особистої інформації, що набуває особливого значення в контексті освітнього процесу. Реалізація чат-бота дозволяє учасникам зручно спілкуватися, забезпечуючи високий рівень конфіденційності та захисту їхніх особистих даних. Імплементація захисту персональних даних в даній системі є досить важливим підходом, що відображає здатність до розв'язання проблеми захисту даних у сфері освіти та демонструє впровадження високих стандартів безпеки в ІС.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація Laravel фреймворку. URL: https://laravel.com/docs/10.x (дата звернення 08.02.2024)
2. Офіційна документація Telegram API. URL: https://core.telegram.org/bots/api (дата звернення 10.02.2024)
3. Usage statistics of server-side programming languages for websites. URL: https://w3techs.com/technologies/overview/programming_language (дата звернення 15.03.2024)
4. Ресурс з бенчмарками залежно від версії PHP. URL: https://onlinephp.io/benchmarks (21.03.2024)
5. Офіційна документація PHP. URL: https://www.php.net/docs.php (дата звернення 11.04.2024)
6. Офіційна документація з Tailwind css. URL: https://tailwindcss.com/docs/installation (дата звернення 21.04.2024)
7. PSR. URL: https://www.php-fig.org/psr/ (дата звернення 02.05.2024)
8. Top 10 Best PHP frameworks. URL: https://www.cloudways.com/blog/best-php-frameworks (дата звернення 10.05.2024)
9. Comparative Analysis of AES and RSA Algorithms for Data Security in Cloud Computing. URL: https://www.mdpi.com/2504-3900/2/1/5 (дата звернення 11.05.2024)