Міністерство освіти і науки України

Черкаський національний університет імені Богдана Хмельницького

Факультет обчислювальної техніки, інтелектуальних та управляючих систем

Кафедра програмного забезпечення автоматизованих систем

КУРСОВА РОБОТА З БАЗ ДАНИХ

на тему «Облік результатів ICPC-олімпіади з програмування»

Студента 3 курсу, групи КС-22

спеціальності 121 «Інженерія програмного забезпечення»

Шафієва Д.Ю.

(прізвище та ініціали)

Керівник:

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Оцінка за шкалою:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(національною, кількість балів, ECTS)

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

Черкаси – 2025 рік

ЗМІСТ

[Вступ 3](#_Toc200924185)

[1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. 5](#_Toc200924186)

[1.1 Актуальність автоматизації обліку результатів командних олімпіад 5](#_Toc200924187)

[1.2 Реалізація інформаційної системи обліку результатів 5](#_Toc200924188)

[2. ПРОЕТУВАННЯ БАЗИ ДАНИХ. 7](#_Toc200924189)

[2.1. Проектування інфологічної та даталогічної моделей. 7](#_Toc200924190)

[2.1.1 Проектування інфологічної моделі. 7](#_Toc200924191)

[2.1.2. Проектування даталогічної моделі. 8](#_Toc200924192)

[2.2 Проектування серверної частини 13](#_Toc200924193)

[2.2.1. Проектування моделей бази данних. 13](#_Toc200924194)

[2.2.2 Проектування запитів 20](#_Toc200924195)

[3. ОПИС КЛІЄНТСЬКОГО ДОДАТКУ 39](#_Toc200924196)

[3.1 Функціональні можливості клієнтського інтерфейсу 39](#_Toc200924197)

[3.2 Огляд інтерфейса додатку 41](#_Toc200924198)

[3.3 Технічна реалізація та вимоги до клієнтської частини 44](#_Toc200924199)

[ВИСНОВКИ. 45](#_Toc200924200)

[СПИСОК ЛІТЕРАТУРИ. 47](#_Toc200924201)

# Вступ

У сучасному світі програмування та інформаційних технологій дедалі більшої ваги набувають системи автоматизації обробки даних. Це особливо актуально у контексті організації та проведення змагань з програмування, таких як ICPC (International Collegiate Programming Contest) — престижна міжнародна олімпіада для студентів вищих навчальних закладів.

Під час проведення таких змагань виникає необхідність у точному та оперативному обліку результатів команд, фіксації спроб надсилання розв’язків, часу їх виконання та визначення команд, які першими правильно розв’язали певні задачі. Усе це вимагає наявності зручної, гнучкої та надійної інформаційної системи, здатної ефективно зберігати, обробляти й відображати великі обсяги структурованих даних.

Метою даної курсової роботи є розробка інформаційної системи для обліку результатів ICPC-олімпіади. Система повинна забезпечувати можливість зберігання інформації про команди, учасників, задачі, мови програмування, заклади освіти, тренерів, а також обробку сабмішенів (відправлених розв’язків) та автоматичне нарахування балів, включно з визначенням перших правильних розв’язків кожної задачі (так званих «кульок»).

Крім базових CRUD-операцій (створення, читання, оновлення, видалення даних), система дозволяє здійснювати розширені вибірки за освітнім рівнем команд, належністю до певного закладу освіти, використаною мовою програмування, вердиктом перевірки тощо. Такий функціонал надає можливість не лише вести облік, а й проводити аналітику та порівняння.

Для реалізації програмної частини було використано мову програмування JavaScript з платформою Node.js та фреймворком Express, а також систему управління базами даних PostgreSQL. REST API забезпечує взаємодію між клієнтом і сервером, а вебінтерфейс, реалізований за допомогою HTML, CSS та JavaScript, дозволяє переглядати інформацію у вигляді динамічних таблиць і здійснювати взаємодію з системою в інтерактивному форматі.

Робота включає повне проектування структури бази даних, реалізацію серверної частини з API-інтерфейсом та клієнтської частини, що дозволяє візуалізувати отриману інформацію. Враховуючи архітектуру системи, вона легко масштабована та може бути розширена для підтримки нових форматів змагань або додаткових типів статистичних запитів.

# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.

## 1.1 Актуальність автоматизації обліку результатів командних олімпіад

Командні змагання з програмування, подібні до ICPC, стали важливою складовою навчального та професійного розвитку студентів технічних спеціальностей. Такі події потребують не лише високої технічної підготовки, а й ретельної організації: ведення обліку результатів, фіксації складу команд, змін протягом сезону та контролю за дотриманням правил участі. Застосування паперових записів або простих інструментів, таких як Google Sheets чи Excel, не забезпечує достатнього рівня ефективності в управлінні цими процесами.

Організатори стикаються з низкою труднощів. Зокрема, важко простежити участь одного й того ж учасника у різні роки, автоматично перевірити відповідність учасників правилам (наприклад, віковим обмеженням або максимальній кількості участей), зафіксувати всі зміни у складі команд, а також здійснити повноцінний аналіз статистики виступів або структури команд.

Ці проблеми вирішуються шляхом створення централізованої бази даних, яка дозволяє зберігати та обробляти структуровану інформацію. У рамках розробленої системи реалізоване збереження основних даних, таких як учасники (зокрема ПІБ, стать, дата народження, роль, унікальний ICPC ID), команди (їх назви, пов’язані організації, історія складу), сезони та етапи змагань (із зазначенням назв, дат та рівнів), результати виступів (місця, кількість розв’язаних задач, штрафний час), заміни у командах протягом сезону, а також організації, які формують команди.

Окрім цього, у системі реалізовано автоматичне нарахування «кульок» — позначення команд, які першими правильно розв’язали кожну задачу. Це є важливою складовою формування остаточного рейтингу і часто впливає на розподіл місць при однаковій кількості балів.

## 1.2 Реалізація інформаційної системи обліку результатів

Програмна реалізація системи побудована з використанням таких технологій. Для реалізації серверної частини використано JavaScript з платформою Node.js та фреймворком Express. У якості системи управління базами даних використано PostgreSQL. Для взаємодії між клієнтом і сервером застосовується REST API. Фронтенд реалізовано з використанням HTML, CSS та JavaScript, що дозволяє створювати інтерактивний інтерфейс користувача з можливістю перегляду таблиць, фільтрації даних і виконання запитів до серверу.

Інтерфейс користувача дозволяє зручно переглядати таблиці з результатами, учасниками, сабмішенами, визначати перші правильні розв’язки задач, переглядати статистику виступів команд та окремих учасників.

У системі реалізовано виконання складних SQL-запитів із можливістю фільтрації даних. Зокрема, користувач може фільтрувати інформацію за роллю (учасник, тренер), за рівнем змагання, за кількістю участей, за вердиктом розв’язку (наприклад, AC, WA, TL), а також за періодом участі.

База даних охоплює низку ключових сутностей і логічно побудованих зв’язків, що дозволяє виконувати запити на кшталт: учасники, які більше двох разів брали участь; усі команди, які отримали хоча б одну «кульку»; тренери, які підготували найуспішніші команди; перша команда, що правильно розв’язала кожну задачу у певному етапі.

Система також підтримує автоматичну ініціалізацію бази даних через SQL-скрипт або програмну логіку із заповненням тестовими даними. На поточному етапі не реалізовано валідацію введених даних, редагування або перевірку відповідності учасників правилам, однак ці функції можуть бути додані у наступних версіях.

Таким чином, розроблена система забезпечує централізоване надійне зберігання інформації, інтерактивний перегляд і пошук даних, підтримку правил ICPC, а також модульність і гнучкість для подальшого масштабування.

# ПРОЕТУВАННЯ БАЗИ ДАНИХ.

## 2.1. Проектування інфологічної та даталогічної моделей.

### 2.1.1 Проектування інфологічної моделі.

У ході аналізу предметної області була сформована інфологічна модель, яка відображає ключові сутності, їхні характеристики та взаємозв’язки між об’єктами реального світу, що підлягають зберіганню й обробці в рамках інформаційної системи.

Під час проєктування цієї моделі було враховано основні поняття, що формують логіку змагань. Зокрема, сезон розглядається як визначений часовий період, у межах якого організовуються етапи змагань. Кожен етап є окремим заходом — наприклад, регіональним або фінальним туром, — і проходить у межах відповідного сезону. Команда виступає як основна одиниця участі у змаганнях та складається з кількох учасників. Учасником вважається окрема особа, яка може мати роль основного учасника, тренера або резервного члена. Організація, у свою чергу, є установою, яку представляє команда на змаганнях.

Окрему увагу приділено збереженню результатів, які містять інформацію про виступ команди на конкретному етапі: зайняте місце, кількість розв’язаних задач і штрафний час. Події замін у командному складі також відображаються в моделі, коли один учасник змінюється іншим у визначену дату. Членство в команді фіксує періоди перебування кожного учасника в складі певної команди, що дає змогу відслідковувати динаміку складу протягом сезону.

Модель також описує логіку зв’язків між сутностями. Один сезон може включати кілька етапів змагань. Кожен із цих етапів пов'язаний із результатами виступів команд. Кожен результат належить до певної команди, а команда — до однієї з організацій. Склад команди формується через сутність, яка дозволяє вказувати дату приєднання й вибуття кожного учасника. Крім того, організація може бути представлена декількома командами. Події замін зв’язані не лише з командою, а й із двома учасниками — тим, кого замінили, і тим, хто його замінив, — із фіксацією дати зміни.

Кожна з наведених сутностей має унікальний ідентифікатор (ID) і набір атрибутів, що відповідають потребам моделі. У результаті розробки було створено ER-діаграму, яка слугує концептуальним представленням структури даних незалежно від конкретної реалізації в системі керування базами даних. Побудована модель є фундаментом для подальшого етапу — переходу до даталогічного проєктування.

Изображение выглядит как диаграмма, зарисовка, рисунок, План

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 2.1 Інфологічна модель

### 2.1.2. Проектування даталогічної моделі.

Виходячи з інфологічної моделі, було розроблено даталогічну модель предметної області, яка реалізована у вигляді реляційної бази даних із використанням СКБД PostgreSQL. Типи даних обрано з урахуванням ефективного зберігання інформації та забезпечення цілісності зв’язків між таблицями.

Таблиця Institution (Заклади освіти):

institution\_id — первинний ключ, тип SERIAL, забезпечує автоматичне зростання значень.

name — назва освітнього закладу, тип TEXT, є обов’язковим до заповнення.

Таблиця Team (Команди):

team\_id — первинний ключ, тип SERIAL.

name — назва команди, тип TEXT, обов’язковий.

education\_level — рівень освіти команди (наприклад, «школа», «університет»), тип TEXT.

institution\_id — зовнішній ключ на таблицю Institution, тип INT, дозволяє зв’язати команду з відповідним навчальним закладом.

Таблиця Participant (Учасники):

participant\_id — первинний ключ, тип SERIAL.

name — ім’я учасника, тип TEXT, обов’язкове.

email — електронна пошта, тип TEXT, може бути відсутньою.

team\_id — зовнішній ключ на таблицю Team, тип INT. Зв’язок із командою, при видаленні команди — учасник також буде видалений (ON DELETE CASCADE).

Таблиця Coach (Тренери):

coach\_id — первинний ключ, тип SERIAL.

name — ім’я тренера, тип TEXT, обов’язкове.

email — електронна пошта тренера, тип TEXT, необов’язкове поле.

team\_id — зовнішній ключ на таблицю Team, тип INT, унікальний (UNIQUE). Це забезпечує правило: одна команда — один тренер. При видаленні команди — тренер також буде видалений (ON DELETE CASCADE).

Таблиця Language (Мови програмування):

language\_id — первинний ключ, тип SERIAL.

name — назва мови програмування (наприклад, C++, Python), тип TEXT, обов’язкове поле.

Таблиця Submission (Спроби):

submission\_id — первинний ключ, тип SERIAL.

team\_id — зовнішній ключ на таблицю Team, тип INT.

language\_id — зовнішній ключ на таблицю Language, тип INT.

problem\_code — код задачі, тип TEXT, обов’язкове поле.

verdict — результат виконання (наприклад, «OK», «Wrong Answer»), тип TEXT.

code — програмний код, поданий командою, тип TEXT.

submission\_time — час подачі розв’язку, тип TIMESTAMP, за замовчуванням поточна мітка часу (CURRENT\_TIMESTAMP).

Таблиця Kulki (Нагороди/Кульки):

problem\_code — первинний ключ, тип TEXT. Кожна задача може бути відзначена лише один раз.

team\_id — зовнішній ключ на таблицю Team, тип INT.

submission\_id — зовнішній ключ на таблицю Submission, тип INT.

awarded\_at — час надання нагороди, тип TIMESTAMP, за замовчуванням поточний час (CURRENT\_TIMESTAMP).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Веб-сайт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 2.2 Даталогічна модель

Аналіз на нормальні форми

Розглянута база даних побудована відповідно до вимог третьої нормальної форми (3НФ), що забезпечує логічну цілісність, мінімізацію надлишковості та оптимізацію структури збереження даних.

Насамперед варто зазначити, що всі таблиці відповідають першій нормальній формі (1НФ), оскільки атрибути у кожній з них є атомарними, тобто не піддаються подальшому поділу без втрати сенсу. Усі значення зберігаються у вигляді елементарних даних, що виключає наявність множинних або повторюваних груп.

У другій нормальній формі (2НФ) передбачається, що кожен неключовий атрибут повністю функціонально залежить від первинного ключа, а не лише від його частини. У поданій схемі всі таблиці мають прості (не складені) первинні ключі, тому повна функціональна залежність між ключами та неключовими атрибутами виконується автоматично. Наприклад, у таблиці Team атрибути name, education\_level та institution\_id повністю залежать від ключа team\_id. Аналогічна ситуація спостерігається і в таблицях Participant, Coach, Submission та інших.

Щодо третьої нормальної форми (3НФ), її умова полягає в усуненні транзитивних залежностей між неключовими атрибутами. У всіх таблицях, включених до даної моделі, не спостерігається залежності одного неключового атрибута від іншого. Наприклад, у таблиці Coach атрибути name та email залежать лише від первинного ключа coach\_id, і не існує залежності email → name або подібної. У таблиці Submission атрибути problem\_code, verdict, code та submission\_time мають залежність лише від submission\_id, і між ними не виникає додаткових функціональних залежностей.

Особливу увагу можна звернути на таблицю Kulki, де первинним ключем є problem\_code. У цій таблиці також відсутні транзитивні залежності збереження зовнішнього ключа submission\_id доцільно, до кожне значення problem\_code може мати кілька пов'язаних спроб (Submission), то структура є обґрунтованою й відповідає нормалізації.

Таким чином, можна зробити висновок, що проєктована база даних повністю відповідає вимогам третьої нормальної форми. Усі атрибути мають чітку функціональну залежність від первинних ключів без транзитивних залежностей, що забезпечує ефективність, узгодженість та відсутність логічних аномалій при оновленні даних.

## Проектування серверної частини

У цьому розділі представлено реалізацію структури бази даних із використанням системи управління базами даних PostgreSQL. Описано створені таблиці, зокрема їх ключові поля, типи даних та накладені обмеження. Після кожного опису демонструється візуальне відображення відповідної таблиці в інтерфейсі pgAdmin. Окрім цього, наведено приклади SQL-запитів, що виконуються над сформованою базою даних з метою отримання потрібної аналітичної інформації.

### 2.2.1. Проектування моделей бази данних.

**1.** Institution (Заклади освіти)

institution\_id: int, PK

name: text, NOT NULL

Обмеження: institution\_id — унікальний, автоматично зростаючий (SERIAL)

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.3 Список закладів освіти

2. Team (Команди)

team\_id: int, PK

name: text, NOT NULL

education\_level: text, NOT NULL

institution\_id: int, FK → Institution(institution\_id), NULL

Обмеження: team\_id — унікальний, автоматично зростаючий  
institution\_id — зовнішній ключ, допускає NULL

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.4 Список команд

3. Participant (Учасники команд)

participant\_id: int, PK

name: text, NOT NULL

email: text, NULL

team\_id: int, FK → Team(team\_id), NOT NULL

Обмеження: participant\_id — унікальний, автоматично зростаючий  
При видаленні команди — учасники також видаляються (ON DELETE CASCADE)

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.5 Список учасників команд

4. Coach (Тренери)

coach\_id: int, PK

name: text, NOT NULL

email: text, NULL

team\_id: int, FK → Team(team\_id), NOT NULL, UNIQUE

Обмеження: coach\_id — унікальний, автоматично зростаючий  
team\_id — зовнішній ключ, причому для кожної команди може бути лише один тренер (UNIQUE). При видаленні команди — тренер також видаляється (ON DELETE CASCADE)

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.6 Список тренерів

5. Language (Мови програмування)

language\_id: int, PK

name: text, NOT NULL

Обмеження: language\_id — унікальний, автоматично зростаючий

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.7 Список мов програмування

6. Submission (Спроби / Сабмішени)

submission\_id: int, PK

team\_id: int, FK → Team(team\_id), NOT NULL

language\_id: int, FK → Language(language\_id), NOT NULL

problem\_code: text, NOT NULL

verdict: text, NOT NULL

code: text, NULL

submission\_time: timestamp, NOT NULL, за замовчуванням — поточний час

Обмеження: submission\_id — унікальний, автоматично зростаючий

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.8 Список спроб

7. Kulki (Кульки за перші правильні розв'язки)

problem\_code: text, PK

team\_id: int, FK → Team(team\_id), NOT NULL

submission\_id: int, FK → Submission(submission\_id), NOT NULL

awarded\_at: timestamp, NOT NULL, за замовчуванням — поточний час

Обмеження: problem\_code — унікальний (кожна задача може мати тільки одну кульку)

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.9 Список кульок

### 2.2.2 Проектування запитів

1. Отримати всіх учасників

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.10 Виконання запиту на отримання всіх учасників

1. Отримати учасників певної команди

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.11 Виконання запиту на отримання учасників певної команди

1. Додати учасника

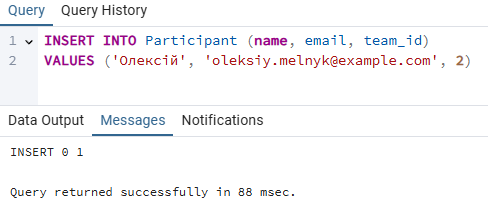


Рис 2.12 Виконання запиту на додавання учасника

1. Отримати всіх тренерів

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.13 Виконання запиту на отримання всіх тренерів

1. Отримати тренера певної команди

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.14 Виконання запиту на отримання тренера певної команди

1. Додати тренера

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.15 Виконання запиту на додавання тренера

1. Оновити учасника

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.16 Виконання запиту на оновлення учасника

1. Видалити учасника

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.17 Виконання запиту на видалення учасника

1. Оновити тренера

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.18 Виконання запиту на оновлення тренера

1. Видалити тренера

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.19 Виконання запиту на видалення тренера

1. Отримати список спроб по командам

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.20 Виконання запиту на отримання списока спроб по командам

1. Фільтрація сабмішенів

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.21 Виконання запиту на фільтрацію сабмішенів

1. Статистика: кількість сабмішенів за мовами

Изображение выглядит как текст, снимок экрана, число, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.22 Виконання запиту на отримання кількості сабмішенів за мовами

1. Статистика: кількість сабмішенів за днями

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.23 Виконання запиту на отримання кількості сабмішенів за днями

1. Останні сабмішени

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.24 Виконання запиту на отримання останніх сабмішенів

1. Команди без жодного OK сабмішена

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.25 Виконання запиту на отримання команд без жодного OK сабмішена

1. Статистика найпоширеніших помилок

Изображение выглядит как текст, снимок экрана, число, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.26 Виконання запиту на отримання найпоширеніших помилок

1. Пошук команд за назвою або закладом освіти

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.27 Виконання запиту на пошук команд за назвою або закладом освіти

1. Спроби команди

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.28 Виконання запиту на отримання спроб команд

1. Результати по закладам освіти

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.29 Виконання запиту на отримання результатів по закладам освіти

1. Результати за рівнем освіти

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.30 Виконання запиту на отримання результатів за рівнем освіти

1. Фільтрація за мовою

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.31 Виконання запиту на фільтрацію за мовою

1. Статистика по вердикту

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.32 Виконання запиту на статистику по вердикту

1. Таблиця лідерів

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.33 Виконання запиту на отримання таблиці лідерів

1. Заклади освіти

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.34 Виконання запиту на отримання закладів освіти

1. Команди

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.35 Виконання запиту на отримання команд

1. Мови програмування

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.36 Виконання запиту на отримання мов програмування

1. Список кульок

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.37 Виконання запиту на отримання списока кульок

1. Виявлення нових кульок

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.38 Виконання запиту на виявлення нових кульок

1. Присвоїти кульки

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис 2.39 Виконання запиту на присвоєння кульок

# ОПИС КЛІЄНТСЬКОГО ДОДАТКУ

## 3.1 Функціональні можливості клієнтського інтерфейсу

Клієнтський додаток є невід’ємною складовою розробленої системи, що забезпечує інтерактивну взаємодію користувача з базою даних результатів ICPC-олімпіади засобами вебінтерфейсу. Основною функцією даного модуля є реалізація доступу до серверного API, обробка введених параметрів запитів, відображення отриманих даних та підтримка зворотного зв’язку з користувачем. Завдяки використанню сучасних підходів до організації фронтенду, забезпечено швидку реакцію інтерфейсу та ефективну візуалізацію інформації.

Функціональні можливості клієнтської частини охоплюють перегляд основних сутностей бази даних, зокрема інформації про команди, учасників, тренерів, заклади освіти, мови програмування та результати сабмішенів. Передбачена підтримка GET-запитів, які користувач може обирати зі спеціального випадаючого списку, що динамічно формується на основі відповідей сервера. Для кожного запиту реалізовано можливість задання маршрутних параметрів та параметрів запиту у відповідних текстових полях. Також передбачено окрему форму для формування POST, PUT та DELETE-запитів, що дає змогу створювати, оновлювати та видаляти записи у базі даних.

Отримані результати відображаються у вигляді таблиць HTML, структура яких адаптується відповідно до вмісту відповіді. Система підтримує виконання аналітичних запитів, зокрема відображення статистики перших правильних розв’язків (так званих «кульок»), розподілу рішень за мовами програмування, частотності помилкових відповідей, а також побудову рейтингів команд та закладів за кількістю успішно вирішених задач. Клієнтський додаток є універсальним у сенсі адаптації до нових API-маршрутів — у разі їхнього додавання на сервері, інтерфейс автоматично починає їх підтримувати без потреби змін у коді фронтенду.

Розробка клієнтської частини здійснювалася з використанням мов HTML, CSS та JavaScript. Стилізація інтерфейсу реалізована засобами стандартного CSS без залучення сторонніх бібліотек чи UI-фреймворків. Комунікація з серверною частиною реалізована через стандартні HTTP-запити за допомогою функції fetch, що забезпечує прямий доступ до REST API, створеного на платформі Node.js із використанням фреймворку Express.

У якості системи керування базами даних було використано PostgreSQL. Обґрунтуванням такого вибору є висока продуктивність системи, стабільна підтримка транзакцій, складних запитів та агрегованих операцій, а також широке поширення цієї СКБД у промислових та академічних проєктах. PostgreSQL також забезпечує зручну інтеграцію з Node.js через модуль pg, що дає змогу ефективно реалізувати доступ до даних на серверному рівні.

Інтерфейс користувача розроблено таким чином, щоб забезпечити зручність введення параметрів, швидке отримання відповідей та чітке відображення інформації. Основна структура інтерфейсу включає панель з випадаючими списками запитів, текстові поля для параметрів, а також динамічну область для відображення таблиць з результатами. Реалізовано обробку повідомлень про успішне виконання запиту або помилки, що виникли в процесі.

Для повноцінного використання клієнтського додатку необхідна наявність персонального комп’ютера або ноутбука з базовими технічними характеристиками. Зокрема, система коректно функціонує на пристроях з двоядерним процесором рівня Intel Core i3 або AMD Ryzen 3, оперативною пам’яттю обсягом не менше 4 ГБ, операційною системою Windows 10, Linux або macOS, а також наявністю сучасного веббраузера (Google Chrome, Mozilla Firefox або Microsoft Edge у поточних версіях). Для цілей локальної розробки також передбачена необхідність встановлення середовища Node.js.

## 3.2 Огляд інтерфейса додатку

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.1 Головне меню додатку

Інтерфейс клієнтської частини вебзастосунку призначений для взаємодії користувача з API, що обслуговує базу даних результатів ICPC-олімпіади. Архітектурно він поділений на два функціональні блоки, кожен з яких реалізує окремі сценарії взаємодії з даними.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.2 Розгорнутий дропдаун гет запитів

Перший блок відповідає за виконання GET-запитів до бази даних. У ньому реалізовано випадаючий список, що містить повний перелік доступних маршрутів запитів, серед яких присутні: вивід команд, учасників, тренерів, сабмішенів, мов програмування, навчальних закладів, статистичних зрізів тощо. Користувач має можливість задати додаткові фільтри у спеціальному полі, що дозволяє обмежити результати (наприклад, за назвою закладу або мовою програмування), а також зазначити параметри до запиту.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.3 Таблиця

Після активації кнопки «Згенерувати таблицю» здійснюється звернення до відповідного ендпоінта, результат якого відображається у вигляді HTML-таблиці нижче. Таблиця автоматично генерується залежно від структури відповіді і підтримує горизонтальне прокручування у випадку великої кількості колонок. Таким чином, цей блок виконує роль інструмента для перегляду, аналізу та фільтрації даних у зручному табличному форматі.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.4 Розгорнутий дропдаун не гет запитів

Другий блок відповідає за зміну стану бази даних через виконання POST, PUT або DELETE-запитів. У ньому також реалізовано випадаючий список, що містить варіанти змін, зокрема: додавання, оновлення або видалення учасника чи тренера, а також призначення «кульок» (відзначень перших вдалих сабмішенів). У відповідному текстовому полі користувач може ввести JSON-об’єкт, що описує новий або змінений запис, або ж зазначити ідентифікатор елемента для видалення. Після натискання кнопки «Виконати» система надсилає запит до серверної частини, яка обробляє зміну і оновлює стан бази.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.5 Результат виконання пост запиту

## 3.3 Технічна реалізація та вимоги до клієнтської частини

Уся клієнтська логіка реалізована засобами HTML, CSS та JavaScript без використання сторонніх UI-бібліотек. Обробка запитів відбувається асинхронно через Fetch API. Отримані результати форматуються динамічно та виводяться у вигляді таблиць, які є універсальними для різних типів отриманих даних. Такий підхід забезпечує гнучкість, масштабованість та зручність використання інтерфейсу для адміністрування результатів олімпіади.

# ВИСНОВКИ.

У межах виконання курсової роботи було розроблено повнофункціональну веборієнтовану інформаційну систему для ведення обліку результатів командної олімпіади з програмування, подібної до формату ICPC. Система підтримує повний цикл CRUD-операцій (створення, читання, оновлення, видалення) для ключових об'єктів предметної області, включаючи команди, учасників, тренерів, задачі, сабмішени, мови програмування, інституції, сезони та етапи змагань.

У процесі реалізації проєкту було здійснено глибокий аналіз предметної області, на основі якого побудовано інфологічну та даталогічну моделі. База даних спроєктована відповідно до принципів реляційного моделювання з урахуванням нормалізації до третьої нормальної форми. Це дозволило мінімізувати надлишковість, уникнути аномалій оновлення та забезпечити цілісність даних.

Для реалізації серверної частини використано стек технологій Node.js з Express.js. База даних розгорнута на PostgreSQL, а взаємодія із нею здійснюється через параметризовані SQL-запити без використання ORM, що забезпечує гнучкість і високу продуктивність. Реалізовано понад 30 запитів, зокрема аналітичні (агрегати, підзапити, JOIN, GROUP BY, FILTER) для статистики рішень, успішності команд, найчастіших помилок тощо.

Фронтенд реалізовано з використанням HTML, CSS та JavaScript. Інтерфейс підтримує динамічну генерацію HTML-таблиць на основі отриманих з сервера даних. Для зручності користувача реалізовано два випадаючих списки для GET та POST/PUT/DELETE-запитів, окремі текстові поля для параметрів та фільтрів, а також функціональність формування запиту у відповідь на дію користувача. Дані з сервера обробляються у форматі JSON, після чого перетворюються на зручну для сприйняття табличну форму.

Особливістю реалізації є підтримка "кульок" — індикаторів команд, які першими правильно розв’язали кожну із задач, що дозволяє візуалізувати першість і досягнення команд. Система також дозволяє виконувати фільтрацію результатів за командами, етапами змагань, мовами програмування та іншими параметрами.

Загалом було реалізовано повноцінну клієнт-серверну архітектуру, адаптовану до ICPC-подібних подій. Система може ефективно використовуватись у навчальних закладах — на кафедрах інформатики, у підрозділах олімпіадної підготовки, а також організаторами міських, регіональних та всеукраїнських змагань з програмування.

Практична значущість розробленої системи полягає в автоматизації обліку участі у змаганнях, зменшенні впливу людського фактору, оптимізації адміністративних процесів і забезпеченні прозорості результатів. З наукової та освітньої точки зору проєкт сприяє розвитку цифрової культури, популяризації алгоритмічного мислення та формуванню механізмів відкритого обліку досягнень студентів.

Таким чином, у межах курсової роботи було спроєктовано інфологічну та фізичну моделі бази даних, реалізовано реляційну базу на основі PostgreSQL із використанням ручного написання SQL-запитів, створено RESTful API-сервер на платформі Node.js з використанням фреймворку Express, розроблено адаптивний фронтенд із підтримкою динамічного відображення таблиць, а також впроваджено розширену аналітику та засоби візуалізації даних.

Отримані результати повністю відповідають поставленим цілям і свідчать про готовність системи до практичного використання та подальшого масштабування.

# СПИСОК ЛІТЕРАТУРИ.

1. Беккер С. SQL. Основи програмування баз даних. — Київ: Діалектика, 2021.
2. Пастернак І.Ю. Бази даних. Проектування, реалізація та використання. — Львів: Видавництво Львівської політехніки, 2019.
3. Date C.J. Вступ до систем баз даних. — 8-ме видання. — Київ: Вільямс, 2020.
4. Сілберштадт Р., Корт Г. Основи розробки інформаційних систем. — Київ: Лібра, 2018.
5. Чен І. Моделювання баз даних за допомогою ER-діаграм. — Київ: Видавництво КНЕУ, 2017.
6. Кормен Т.Х., Лейзерсон Ч.Е., Рівест Р.Л., Стайн К. Алгоритми: побудова та аналіз. — Київ: Вільямс, 2020.
7. Буч Г. Об’єктно-орієнтований аналіз і проектування з UML. — Київ: Арт-ПРЕС, 2021.
8. Харт Д. Розробка вебдодатків з використанням Node.js. — Київ: Фактор, 2022.
9. Поллак А. Express.js: Гнучка розробка вебдодатків за допомогою Node.js. — Київ: Нова Книга, 2021.
10. Фленаган Д. JavaScript. Повний довідник. — 7-ме видання. — Київ: Діалектика, 2020.
11. Гудман Д. HTML і CSS. Розробка вебінтерфейсів. — Київ: Видавнича група BHV, 2019.
12. Mozilla Developer Network (MDN). JavaScript Documentation. — <https://developer.mozilla.org/uk/docs/Web/JavaScript>
13. PostgreSQL Global Development Group. Офіційна документація PostgreSQL. — <https://www.postgresql.org/docs/>
14. International Collegiate Programming Contest (ICPC). Офіційний сайт ICPC. — [https://icpc.global](https://icpc.global/)
15. W3Schools. HTML, CSS та JavaScript Підручники. — [https://www.w3schools.com](https://www.w3schools.com/)
16. Robson E., Freeman E. Head First: Веброзробка. — Київ: Наш Формат, 2021.
17. Ющенко А.І. Інформаційні системи та технології в управлінні. — Київ: КНЕУ, 2020.
18. Войцеховський Д. Вебпрограмування на JavaScript, Node.js та React. — Харків: Фоліо, 2023.
19. Федоренко Л.П. Теорія інформаційних систем. — Львів: Видавництво Львівської політехніки, 2019.
20. Назаров А.С. Побудова клієнт-серверних додатків. — Київ: Видавництво НТУУ "КПІ", 2021.