

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 ІСРС-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
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.	5
1.1 Актуальність автоматизації обліку результатів командних олімпіад...	5
1.2 Реалізація інформаційної системи обліку результатів.....	5
2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ.	7
2.1. Проектування інфологічної та даталогічної моделей.....	7
2.1.1 Проектування інфологічної моделі.	7
2.1.2. Проектування даталогічної моделі.	8
2.2 Проектування серверної частини	13
2.2.1. Проектування моделей бази даних.....	13
2.2.2 Проектування запитів	20
3. ОПИС КЛІЄНТСЬКОГО ДОДАТКУ.....	39
3.1 Функціональні можливості клієнтського інтерфейсу	39
3.2 Огляд інтерфейса додатку	41
3.3 Технічна реалізація та вимоги до клієнтської частини	44
ВИСНОВКИ.....	45
СПИСОК ЛІТЕРАТУРИ.....	47

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.

1. 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 кожен етап.

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.

2. ПРОЕКТУВАННЯ 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 проектування 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 проектування.

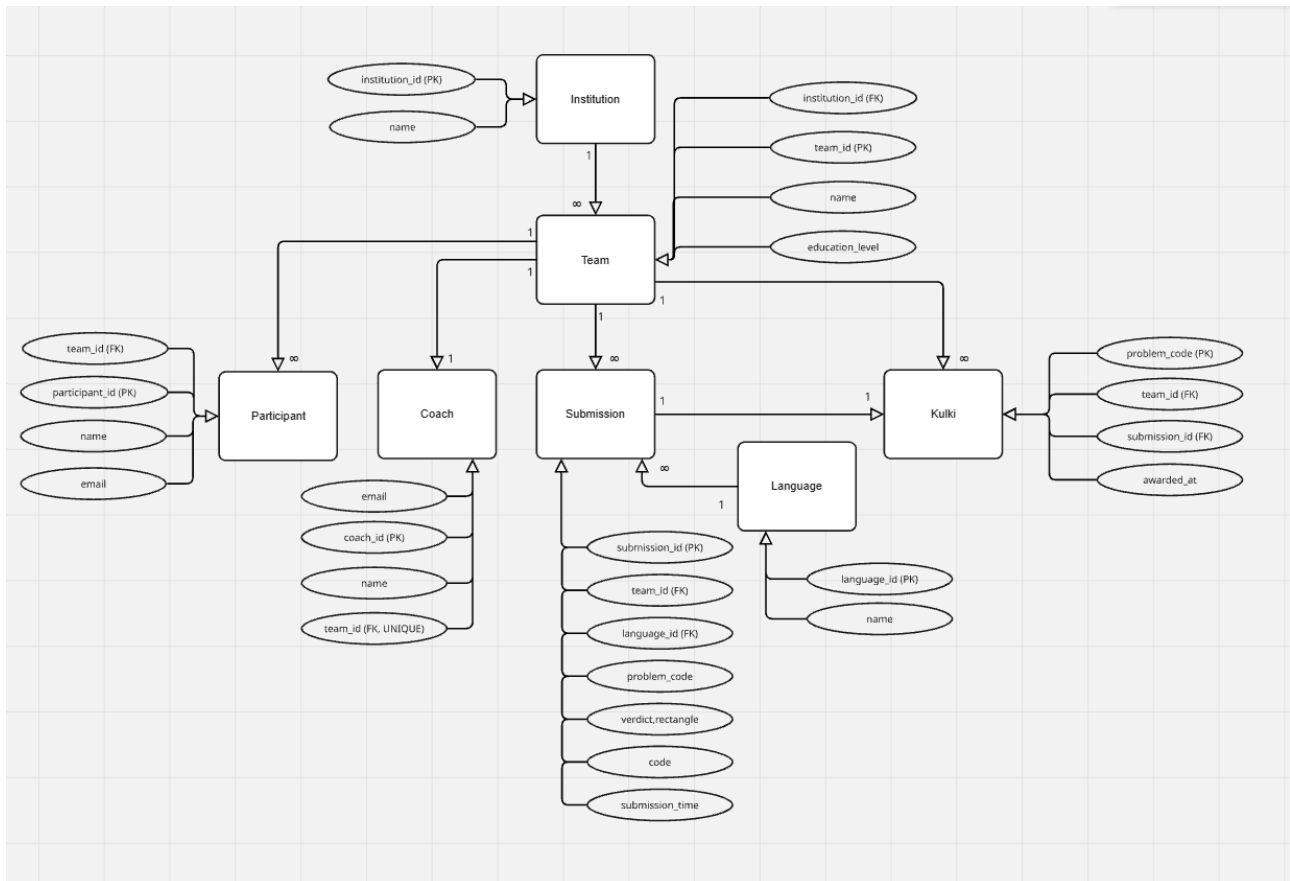


Рис. 2.1 Інфологічна модель

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).

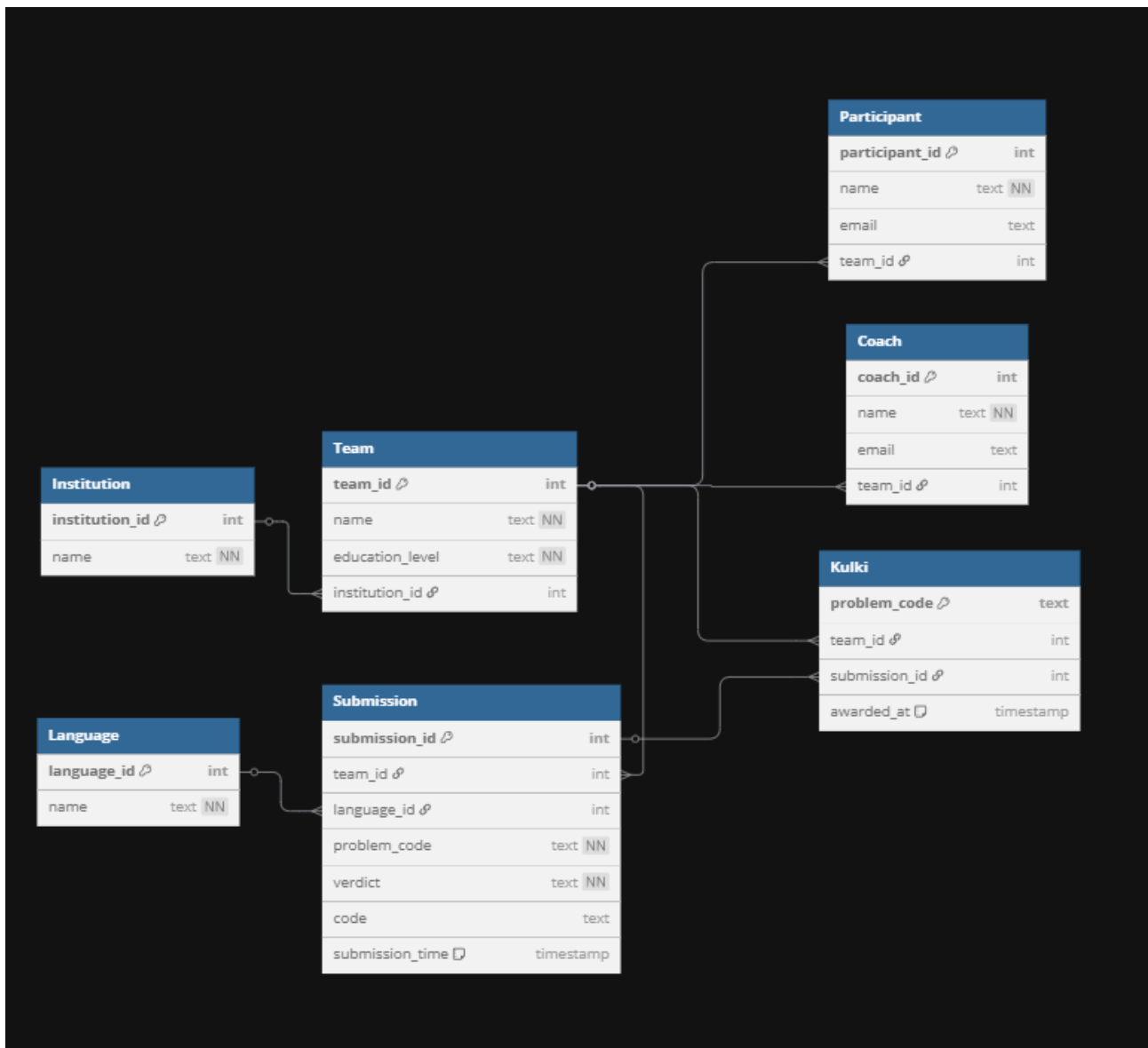


Рис. 2.2 Даталогічна модель

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 кожного 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.

2.2 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.

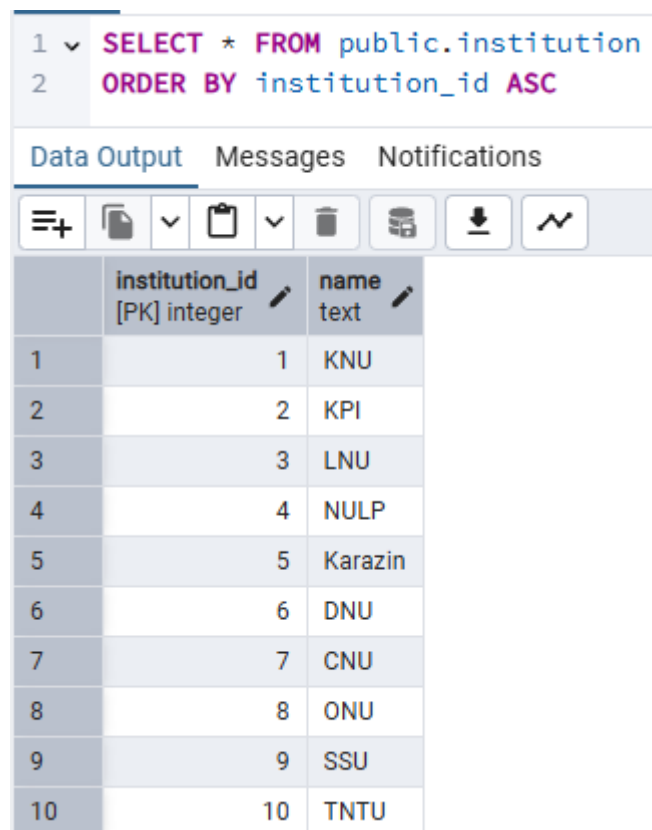
2.2.1. Проектування моделей бази даних.

1. Institution (Заклади освіти)

institution_id: int, PK

name: text, NOT NULL

Обмеження: institution_id — унікальний, автоматично зростаючий (SERIAL)



The screenshot shows the pgAdmin interface. At the top, a SQL query is entered in the query editor:

```
1 SELECT * FROM public.institution
2 ORDER BY institution_id ASC
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with 10 rows of data. The table has two columns: 'institution_id' (integer, PK) and 'name' (text). The data is as follows:

	institution_id [PK] integer	name text
1	1	KNU
2	2	KPI
3	3	LNU
4	4	NULP
5	5	Karazin
6	6	DNU
7	7	CNU
8	8	ONU
9	9	SSU
10	10	TNTU

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

Query

Query History

1

2

SELECT * FROM public.team

ORDER BY team_id ASC

Data Output

Messages

Notifications

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)

Query

Query History

1

2

SELECT * FROM public.participant

ORDER BY participant_id ASC

Data Output

Messages

Notifications

≡

+

<

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)

Query

Query History

1 SELECT * FROM public.coach

2 ORDER BY coach_id ASC

Data Output

Messages

Notifications

	coach_id [PK] integer	name text	email text	team_id integer
1	2	Марія Гончар	mariia.honchar@example.com	2
2	3	Тарас Коваль	taras.koval@example.com	3
3	4	Андрій Романенко	andriy.romanenko@example.com	4
4	5	Тарас Коваль	taras.koval@example.com	5
5	6	Світлана Мельник	svitlana.melnyk@example.com	6
6	7	Олексій Коваль	oleksiy.koval@example.com	7
7	8	Світлана Романен...	svitlana.romanenko@example.com	8
8	9	Тарас Сидоренко	taras.sydorenko@example.com	9
9	10	Наталія Степаненко	nataliia.stepanenko@example.com	10
10	11	Олена Романенко	olena.romanenko@example.com	11
11	12	Юрій Петренко	iuriy.petrenko@example.com	12
12	13	Світлана Петренко	svitlana.petrenko@example.com	13
13	14	Світлана Шевченко	svitlana.shevchenko@example.com	14
14	15	Дмитро Сидоренко	dmytro.sydorenko@example.com	15
15	16	Юрій Іванов	iuriy.ivanov@example.com	16
16	17	Тарас Іванов	taras.ivanov@example.com	17
17	18	Юрій Степаненко	iuriy.stepanenko@example.com	18
18	19	Юрій Бондаренко	iuriy.bondarenko@example.com	19
19	20	Олексій Шевченко	oleksiy.shevchenko@example.com	20
20	21	Андрій Степаненко	andriy.stepanenko@example.com	21
21	22	Марія Гончар	mariia.honchar@example.com	22
22	23	Олена Сидоренко	olena.sydorenko@example.com	23
23	24	Наталія Коваль	nataliia.koval@example.com	24

Total rows: 100

Query complete 00:00:00.108

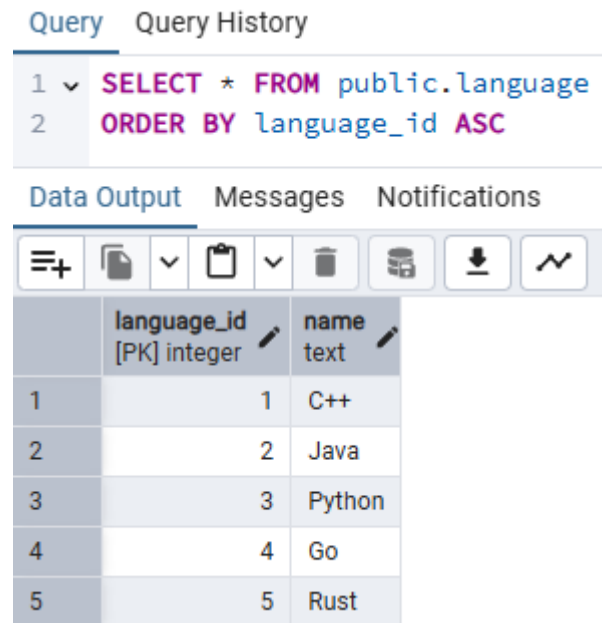
Рис 2.6 Список тренерів

5. Language (Мови програмування)

language_id: int, PK

name: text, NOT NULL

Обмеження: language_id — унікальний, автоматично зростаючий



The screenshot shows a database query interface. At the top, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query: `SELECT * FROM public.language ORDER BY language_id ASC`. Below the query, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with two columns: 'language_id' (integer, PK) and 'name' (text). The table contains five rows of data: (1, C++), (2, Java), (3, Python), (4, Go), and (5, Rust). Above the table, there is a toolbar with various icons for editing and viewing the data.

	language_id [PK] integer	name text
1	1	C++
2	2	Java
3	3	Python
4	4	Go
5	5	Rust

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

QueryQuery History

12

SELECT * FROM public.submission
ORDER BY submission_id ASC

Data OutputMessagesNotifications

	submission_id [PK] integer	team_id integer	language_id integer	problem_code text	verdict text	code text	submission_time timestamp without time zone
1	1	1	4	A	RE	// code for A by team 1	2025-06-14 12:35:00
2	2	1	2	B	OK	// code for B by team 1	2025-06-14 12:57:00
3	3	1	5	B	RE	// code for B by team 1	2025-06-14 11:34:00
4	4	1	3	C	OK	// code for C by team 1	2025-06-14 12:51:00
5	5	1	5	C	RE	// code for C by team 1	2025-06-14 11:28:00
6	6	1	5	D	RE	// code for D by team 1	2025-06-14 12:29:00
7	7	1	1	D	OK	// code for D by team 1	2025-06-14 12:57:00
8	8	1	4	D	WA	// code for D by team 1	2025-06-14 10:14:00
9	9	1	2	E	RE	// code for E by team 1	2025-06-14 12:15:00
10	10	1	3	E	OK	// code for E by team 1	2025-06-14 12:46:00
11	11	1	1	E	OK	// code for E by team 1	2025-06-14 11:12:00
12	12	1	1	F	OK	// code for F by team 1	2025-06-14 10:35:00
13	13	1	4	F	RE	// code for F by team 1	2025-06-14 12:47:00
14	14	1	1	F	OK	// code for F by team 1	2025-06-14 10:37:00
15	15	1	5	G	TLE	// code for G by team 1	2025-06-14 12:29:00
16	16	2	5	A	OK	// code for A by team 2	2025-06-14 11:35:00
17	17	2	4	A	TLE	// code for A by team 2	2025-06-14 11:19:00
18	18	2	3	B	RE	// code for B by team 2	2025-06-14 11:48:00
19	19	2	2	B	RE	// code for B by team 2	2025-06-14 12:19:00
20	20	2	1	B	WA	// code for B by team 2	2025-06-14 10:21:00
21	21	2	4	C	WA	// code for C by team 2	2025-06-14 10:58:00
22	22	2	2	C	RE	// code for C by team 2	2025-06-14 10:33:00
23	23	2	2	D	OK	// code for D by team 2	2025-06-14 11:05:00

Total rows: 1403Query complete 00:00:00.144

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)

Query

Query History

1

2

SELECT

*

FROM

public.kulki

ORDER BY

problem_code

ASC

Data Output

Messages

Notifications

	problem_code [PK] text	team_id integer	submission_id integer	awarded_at timestamp without time zone
1	A	29	423	2025-06-15 01:25:31.269511
2	B	93	1302	2025-06-15 01:25:31.269511
3	C	48	696	2025-06-15 01:25:31.269511
4	D	24	354	2025-06-15 01:25:31.269511
5	E	2	25	2025-06-15 01:25:31.269511
6	F	8	120	2025-06-15 01:25:31.269511
7	G	87	1239	2025-06-15 01:25:31.269511

Рис 2.9 Список кульок

2.2.2 Проектування запитів

1. Отримати всіх учасників

The screenshot displays a database query interface. At the top, there are tabs for 'Query', 'Query History', and 'Scratch Pad'. The 'Query' tab is active, showing an SQL query:

```
1 SELECT p.*, t.name AS team_name
2 FROM Participant p
3 JOIN Team t ON p.team_id = t.team_id
4 ORDER BY p.participant id;
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table of results. The table has 6 columns: 'participant_id' (integer), 'name' (text), 'email' (text), 'team_id' (integer), and 'team_name' (text). The results are displayed in a table with 12 rows.

	participant_id integer	name text	email text	team_id integer	team_name text
1	1	Олексій Мельник	oleksiy.melnyk@example.com	1	Team #1
2	2	Олена Коваль	olena.koval@example.com	1	Team #1
3	3	Андрій Степаненко	andriy.stepanenko@example.com	1	Team #1
4	4	Наталія Степаненко	nataliia.stepanenko@example.com	2	Team #2
5	5	Олексій Романенко	oleksiy.romanenko@example.com	2	Team #2
6	6	Іван Степаненко	ivan.stepanenko@example.com	2	Team #2
7	7	Іван Сидоренко	ivan.sydorenko@example.com	3	Team #3
8	8	Марія Петренко	mariia.petrenko@example.com	3	Team #3
9	9	Юрій Бондаренко	iuriy.bondarenko@example.com	3	Team #3
10	10	Іван Коваль	ivan.koval@example.com	4	Team #4
11	11	Дмитро Петренко	dmytro.petrenko@example.com	4	Team #4
12	12	Дмитро Гончар	dmytro.honchar@example.com	4	Team #4

Рис 2.10 Виконання запиту на отримання всіх учасників

2. Отримати учасників певної команди

The screenshot shows a database query interface. At the top, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query: `SELECT * FROM Participant WHERE team_id = 1;`. Below the query, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with 4 columns: `participant_id` (integer, PK), `name` (text), `email` (text), and `team_id` (integer). The table contains 3 rows of data. Above the table, there is a toolbar with icons for various actions and a status bar indicating 'Showing rows: 1 to 3' and 'Page No: 1 of 1'.

	<code>participant_id</code> [PK] integer	<code>name</code> text	<code>email</code> text	<code>team_id</code> integer
1	1	Олексій Мельник	oleksiy.melnyk@example.com	1
2	2	Олена Коваль	olena.koval@example.com	1
3	3	Андрій Степаненко	andriy.stepanenko@example.com	1

Рис 2.11 Виконання запиту на отримання учасників певної команди

3. Додати учасника

The screenshot shows a database query interface. At the top, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query: `INSERT INTO Participant (name, email, team_id) VALUES ('Олексій', 'oleksiy.melnyk@example.com', 2)`. Below the query, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, showing the message: 'INSERT 0 1' and 'Query returned successfully in 88 msec.'

INSERT 0 1

Query returned successfully in 88 msec.

Рис 2.12 Виконання запиту на додавання учасника

4. Отримати всіх тренерів

Query Query History

```
2 FROM Coach c
3 JOIN Team t ON c.team_id = t.team_id
4 ORDER BY c.coach_id;
5
```

Data Output Messages Notifications

Showing rows: 1 to 100 Page No: 1 of 1

	coach_id integer	name text	email text	team_id integer	team_name text
1	1	Наталія Гончар	nataliia.honchar@example.com	1	Team #1
2	2	Марія Гончар	mariia.honchar@example.com	2	Team #2
3	3	Тарас Коваль	taras.koval@example.com	3	Team #3
4	4	Андрій Романенко	andriy.romanenko@example.com	4	Team #4
5	5	Тарас Коваль	taras.koval@example.com	5	Team #5
6	6	Світлана Мельник	svitlana.melnyk@example.com	6	Team #6
7	7	Олексій Коваль	oleksiy.koval@example.com	7	Team #7
8	8	Світлана Романен...	svitlana.romanenko@example.com	8	Team #8
9	9	Тарас Сидоренко	taras.sydorenko@example.com	9	Team #9
10	10	Наталія Степаненко	nataliia.stepanenko@example.com	10	Team #10
11	11	Олена Романенко	olena.romanenko@example.com	11	Team #11
12	12	Юрій Петренко	iuriy.petrenko@example.com	12	Team #12
13	13	Світлана Петренко	svitlana.petrenko@example.com	13	Team #13

Рис 2.13 Виконання запиту на отримання всіх тренерів

5. Отримати тренера певної команди

The screenshot shows a database query interface with a 'Query' tab selected. The query text is: `SELECT * FROM Coach WHERE team_id = 1;`. Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with 4 columns: `coach_id` [PK] integer, `name` text, `email` text, and `team_id` integer. The table contains one row with the following values: `coach_id` is 1, `name` is 'Наталія Гончар', `email` is 'nataliia.honchar@example.com', and `team_id` is 1. The interface also includes a toolbar with icons for query execution, saving, and other database operations, and a pagination control showing 'Showing rows: 1 to 1' and 'Page No: 1 of 1'.

	<code>coach_id</code> [PK] integer	<code>name</code> text	<code>email</code> text	<code>team_id</code> integer
1	1	Наталія Гончар	nataliia.honchar@example.com	1

Рис 2.14 Виконання запиту на отримання тренера певної команди

6. Додати тренера

The screenshot shows a database query interface with a 'Query' tab selected. The query text is: `INSERT INTO Coach (name, email, team_id) VALUES ('Наталія', 'nataliia.honchar@example.com', 1) RETURNING *;`. Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with 4 columns: `coach_id` [PK] integer, `name` text, `email` text, and `team_id` integer. The table contains one row with the following values: `coach_id` is 101, `name` is 'Наталія', `email` is 'nataliia.honchar@example.com', and `team_id` is 1. The interface also includes a toolbar with icons for query execution, saving, and other database operations, and a pagination control showing 'Showing rows: 1 to 1' and 'Page No: 1 of 1'.

	<code>coach_id</code> [PK] integer	<code>name</code> text	<code>email</code> text	<code>team_id</code> integer
1	101	Наталія	nataliia.honchar@example.com	1

Рис 2.15 Виконання запиту на додавання тренера

7. Оновити учасника

The screenshot shows a database query editor with two tabs: "Query" and "Query History". The "Query" tab is active, displaying the following SQL statement:

```
1 UPDATE Participant
2 SET name = 'Олексій Мельник', email = 'oleksiy.melnyk@example.com', team_id = 2
3 WHERE participant_id = 301
4 RETURNING *;
5
```

Below the query editor, there are tabs for "Data Output", "Messages", and "Notifications". The "Data Output" tab is active, showing a table with the results of the query. The table has four columns: "participant_id [PK] integer", "name text", "email text", and "team_id integer". The table contains one row with the following data:

participant_id [PK] integer	name text	email text	team_id integer
301	Олексій Мельник	oleksiy.melnyk@example.com	2

Below the table, there is a status bar showing "Showing rows: 1 to 1" and "Page No: 1 of 1".

Рис 2.16 Виконання запиту на оновлення учасника

8. Видалити учасника

The screenshot shows a database query editor with two tabs: "Query" and "Query History". The "Query" tab is active, displaying the following SQL statement:

```
1 DELETE FROM Participant WHERE participant_id = 301;
2
```

Below the query editor, there are tabs for "Data Output", "Messages", and "Notifications". The "Messages" tab is active, showing the following message:

```
DELETE 1
```

Below the message, there is a status bar showing "Query returned successfully in 64 msec."

Рис 2.17 Виконання запиту на видалення учасника

9. Оновити тренера

Query Query History

```
3 SET name = 'Наталія Гончар'
4 WHERE coach_id = 101
5 RETURNING *;
6
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	coach_id [PK] integer	name text	email text	team_id integer
1	101	Наталія Гончар	nataliia.honchar@example.com	1

Рис 2.18 Виконання запиту на оновлення тренера

10. Видалити тренера

Query Query History

```
1 DELETE FROM Coach WHERE coach_id = 1;
2
```

Data Output Messages Notifications

DELETE 1

Рис 2.19 Виконання запиту на видалення тренера

11. Отримати список спроб по командам

Query Query History

```
1 SELECT s.submission_id, t.name AS team, i.name AS institution,
2       t.education_level, l.name AS language,
3       s.problem_code, s.verdict, s.submission_time
4 FROM Submission s
5 JOIN Team t ON s.team_id = t.team_id
6 JOIN Institution i ON t.institution_id = i.institution_id
7 JOIN Language l ON s.language_id = l.language_id
8 WHERE 1=1
9 AND t.name = 'Team #1'
```

Data Output Messages Notifications

	submission_id integer	team text	institution text	education_level text	language text	problem_code text	verdict text	submission_time timestamp without time zone
1	1	Team #1	LNU	School	Go	A	RE	2025-06-14 12:35:00
2	2	Team #1	LNU	School	Java	B	OK	2025-06-14 12:57:00
3	3	Team #1	LNU	School	Rust	B	RE	2025-06-14 11:34:00
4	4	Team #1	LNU	School	Python	C	OK	2025-06-14 12:51:00
5	5	Team #1	LNU	School	Rust	C	RE	2025-06-14 11:28:00
6	6	Team #1	LNU	School	Rust	D	RE	2025-06-14 12:29:00
7	7	Team #1	LNU	School	C++	D	OK	2025-06-14 12:57:00
8	8	Team #1	LNU	School	Go	D	WA	2025-06-14 10:14:00
9	9	Team #1	LNU	School	Java	E	RE	2025-06-14 12:15:00
10	10	Team #1	LNU	School	Python	E	OK	2025-06-14 12:46:00
11	11	Team #1	LNU	School	C++	E	OK	2025-06-14 11:12:00
12	12	Team #1	LNU	School	C++	F	OK	2025-06-14 10:35:00
13	13	Team #1	LNU	School	Go	F	RE	2025-06-14 12:47:00
14	14	Team #1	LNU	School	C++	F	OK	2025-06-14 10:37:00
15	15	Team #1	LNU	School	Rust	G	TLE	2025-06-14 12:29:00

Рис 2.20 Виконання запиту на отримання списка спроб по командам

12. Фільтрація сабмішенів

Query Query History

```
1 SELECT
2   s.*,
3   t.name AS team_name,
4   l.name AS language_name,
5   i.name AS institution_name,
6   t.education_level
7 FROM Submission s
8 JOIN Team t ON s.team_id = t.team_id
9 JOIN Language l ON s.language_id = l.language_id
10 JOIN Institution i ON t.institution_id = i.institution_id
11 WHERE 1=1
12 AND t.name = 'Team #99'
13 AND t.education_level = 'School'
14 AND s.verdict = 'OK'
15 ORDER BY s.submission_time DESC;
16
```

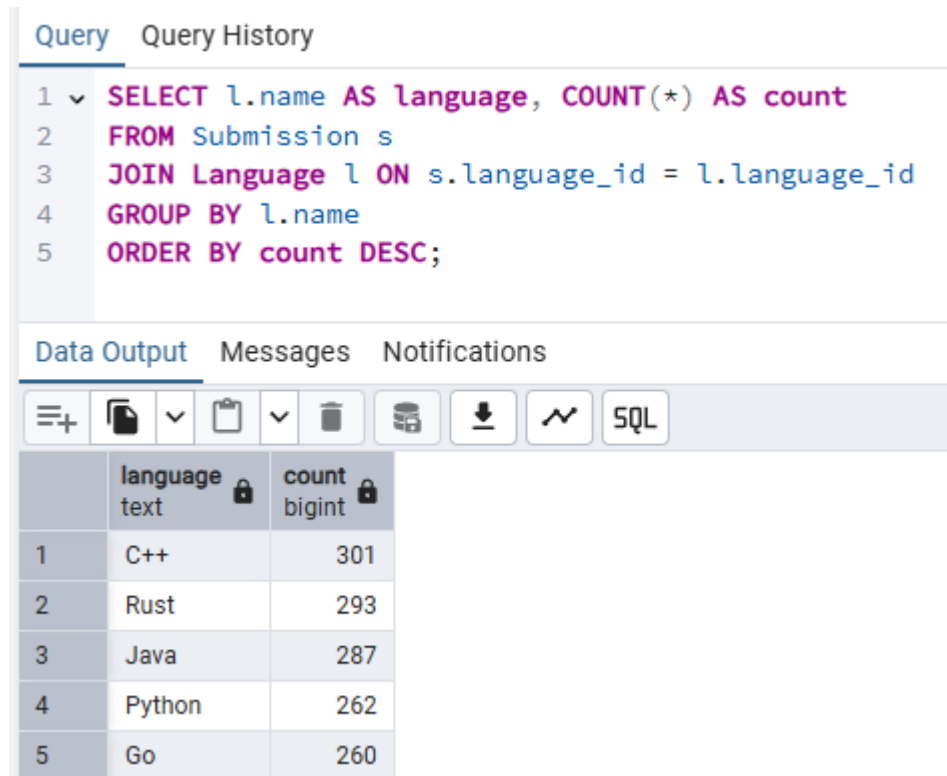
Data Output Messages Notifications

Showing rows: 1 to 4 Page No: 1

	submission_id integer	team_id integer	language_id integer	problem_code text	verdict text	code text	submission_time timestamp without time zone	team_name text	language_name text	institution_name text	education_level text
1	1382	99	1	D	OK	// code for D by team ...	2025-06-14 12:27:00	Team #99	C++	Karazin	School
2	1376	99	4	A	OK	// code for A by team 99	2025-06-14 12:17:00	Team #99	Go	Karazin	School
3	1387	99	3	F	OK	// code for F by team 99	2025-06-14 11:01:00	Team #99	Python	Karazin	School
4	1379	99	1	C	OK	// code for C by team 99	2025-06-14 10:53:00	Team #99	C++	Karazin	School

Рис 2.21 Виконання запиту на фільтрацію сабмішенів

13. Статистика: кількість сабмішенів за мовами



The screenshot shows a database query interface with a 'Query' tab selected. The SQL query is as follows:

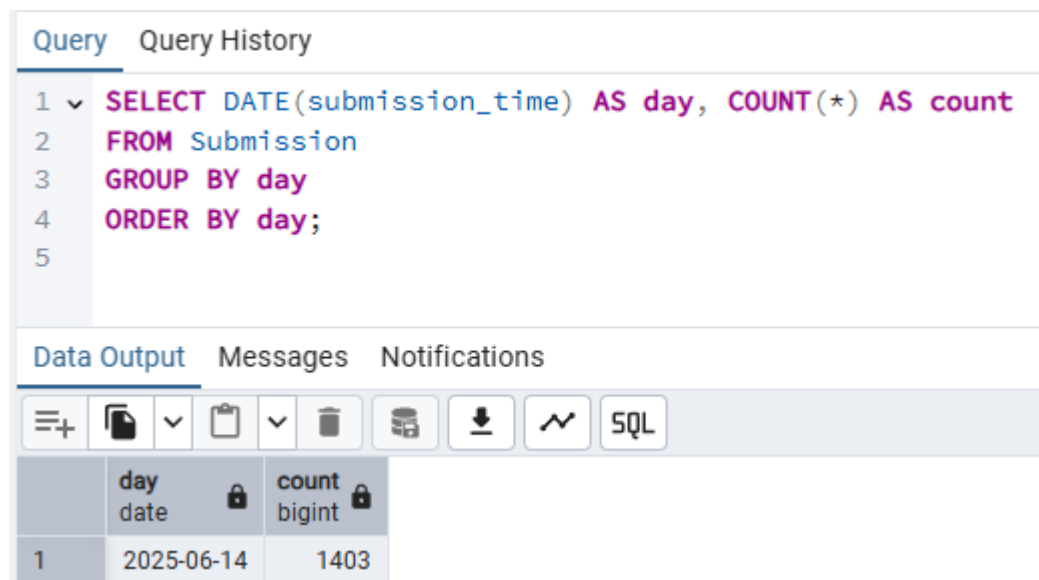
```
1 SELECT l.name AS language, COUNT(*) AS count
2 FROM Submission s
3 JOIN Language l ON s.language_id = l.language_id
4 GROUP BY l.name
5 ORDER BY count DESC;
```

Below the query, the 'Data Output' tab is selected, displaying the results in a table. The table has two columns: 'language' (text) and 'count' (bigint). The results are ordered by count in descending order.

	language text	count bigint
1	C++	301
2	Rust	293
3	Java	287
4	Python	262
5	Go	260

Рис 2.22 Виконання запиту на отримання кількості сабмішенів за мовами

14. Статистика: кількість сабмішенів за днями



The screenshot shows a database query interface with a 'Query' tab selected. The SQL query is as follows:

```
1 SELECT DATE(submission_time) AS day, COUNT(*) AS count
2 FROM Submission
3 GROUP BY day
4 ORDER BY day;
```

Below the query, the 'Data Output' tab is selected, displaying the results in a table. The table has two columns: 'day' (date) and 'count' (bigint). The results are ordered by day.

	day date	count bigint
1	2025-06-14	1403

Рис 2.23 Виконання запиту на отримання кількості сабмішенів за днями

15. Останні сабмішени

Query

Query History

```

2  SELECT s.*, t.name AS team_name
3  FROM Submission s
4  JOIN Team t ON s.team_id = t.team_id
5  ORDER BY submission_time DESC
6  LIMIT 1;
7

```

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

🗄️

📥

📈

SQL

	submission_id integer	team_id integer	language_id integer	problem_code text	verdict text	code text	submission_time timestamp without time zone	team_name text
1	82	6	2	D	OK	// code for D by team 6	2025-06-14 13:00:00	Team #6

Рис 2.24 Виконання запиту на отримання останніх сабмішенів

16. Команди без жодного ОК сабмішена

```
Query  Query History
1  SELECT t.team_id, t.name AS team_name, i.name AS institution_name
2  FROM Team t
3  JOIN Institution i ON t.institution_id = i.institution_id
4  LEFT JOIN Submission s ON t.team_id = s.team_id AND s.verdict = 'OK'
5  GROUP BY t.team_id, t.name, i.name
6  HAVING COUNT(s.verdict) = 0;
```

Data Output Messages Notifications

	team_id integer	team_name text	institution_name text
1	72	Team #72	TNTU

Рис 2.25 Виконання запиту на отримання команд без жодного ОК
сабмішена

17. Статистика найпоширеніших помилок

The screenshot shows a SQL query editor with a query window and a data output window. The query is as follows:

```
1 SELECT verdict, COUNT(*) AS count
2 FROM Submission
3 WHERE verdict != 'OK'
4 GROUP BY verdict
5 ORDER BY count DESC;
```

The data output window shows the following results:

	verdict text	count bigint
1	RE	379
2	TLE	346
3	WA	318

Рис 2.26 Виконання запиту на отримання найпоширеніших помилок

18. Пошук команд за назвою або закладом освіти

The screenshot shows a SQL query editor with a query window and a data output window. The query is as follows:

```
1 SELECT t.team_id, t.name, i.name AS institution
2 FROM Team t
3 JOIN Institution i ON t.institution_id = i.institution_id
4 WHERE t.name ILIKE 'Team #100' OR i.name ILIKE 'KNU'
5 ORDER BY t.name;
```

The data output window shows the following results:

	team_id integer	name text	institution text
1	100	Team #100	TNTU
2	2	Team #2	KNU
3	21	Team #21	KNU
4	23	Team #23	KNU
5	25	Team #25	KNU
6	26	Team #26	KNU
7	4	Team #4	KNU
8	62	Team #62	KNU
9	66	Team #66	KNU
10	7	Team #7	KNU
11	75	Team #75	KNU
12	82	Team #82	KNU
13	90	Team #90	KNU

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

19. Спроби команди

Query

Query History

1

SELECT

*

FROM

Submission

2

WHERE

team_id

=

1

3

ORDER BY

submission_id

DESC;

4

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

📦

📥

📈

SQL

	submission_id [PK] integer	team_id integer	language_id integer	problem_code text	verdict text	code text	submission_time timestamp without time zone
1	15	1	5	G	TLE	// code for G by team...	2025-06-14 12:29:00
2	14	1	1	F	OK	// code for F by team 1	2025-06-14 10:37:00
3	13	1	4	F	RE	// code for F by team 1	2025-06-14 12:47:00
4	12	1	1	F	OK	// code for F by team 1	2025-06-14 10:35:00
5	11	1	1	E	OK	// code for E by team 1	2025-06-14 11:12:00
6	10	1	3	E	OK	// code for E by team 1	2025-06-14 12:46:00
7	9	1	2	E	RE	// code for E by team 1	2025-06-14 12:15:00
8	8	1	4	D	WA	// code for D by team...	2025-06-14 10:14:00
9	7	1	1	D	OK	// code for D by team...	2025-06-14 12:57:00
10	6	1	5	D	RE	// code for D by team...	2025-06-14 12:29:00
11	5	1	5	C	RE	// code for C by team 1	2025-06-14 11:28:00
12	4	1	3	C	OK	// code for C by team 1	2025-06-14 12:51:00
13	3	1	5	B	RE	// code for B by team 1	2025-06-14 11:34:00
14	2	1	2	B	OK	// code for B by team 1	2025-06-14 12:57:00
15	1	1	4	A	RE	// code for A by team 1	2025-06-14 12:35:00

Рис 2.28 Виконання запиту на отримання спроб команд

20. Результати по закладам освіти

Query
Query History

```

1 SELECT s.*
2 FROM Submission s
3 JOIN Team t ON s.team_id = t.team_id
4 JOIN Institution i ON t.institution_id = i.institution_id
5 WHERE i.name ILIKE 'KNU';
6

```

Data Output
Messages
Notifications

≡
📄
▼
🗑️
▼
🔍
⬇️
📶
SQL

	submission_id <small>[PK] integer</small>	team_id <small>integer</small>	language_id <small>integer</small>	problem_code <small>text</small>	verdict <small>text</small>	code <small>text</small>	submission_time <small>timestamp without time zone</small>
1	16	2	5	A	OK	// code for A by team 2	2025-06-14 11:35:00
2	17	2	4	A	TLE	// code for A by team 2	2025-06-14 11:19:00
3	18	2	3	B	RE	// code for B by team 2	2025-06-14 11:48:00
4	19	2	2	B	RE	// code for B by team 2	2025-06-14 12:19:00
5	20	2	1	B	WA	// code for B by team 2	2025-06-14 10:21:00
6	21	2	4	C	WA	// code for C by team 2	2025-06-14 10:58:00
7	22	2	2	C	RE	// code for C by team 2	2025-06-14 10:33:00
8	23	2	2	D	OK	// code for D by team 2	2025-06-14 11:05:00
9	24	2	5	D	TLE	// code for D by team 2	2025-06-14 10:42:00
10	25	2	3	E	OK	// code for E by team 2	2025-06-14 10:00:00
11	26	2	5	E	TLE	// code for E by team 2	2025-06-14 11:14:00
12	27	2	3	E	OK	// code for E by team 2	2025-06-14 11:33:00
13	28	2	4	F	RE	// code for F by team 2	2025-06-14 12:06:00
14	29	2	3	F	WA	// code for F by team 2	2025-06-14 11:36:00
15	30	2	4	F	RE	// code for F by team 2	2025-06-14 10:00:00
16	31	2	2	G	OK	// code for G by team 2	2025-06-14 10:18:00
17	48	4	2	A	RE	// code for A by team 4	2025-06-14 12:53:00
18	49	4	1	A	TLE	// code for A by team 4	2025-06-14 12:25:00

Total rows: 177
Query complete 00:00:00.135

Рис 2.29 Виконання запиту на отримання результатів по закладам освіти

21. Результати за рівнем освіти

Query

Query History

1

2

3

4

5

SELECT s.*

FROM Submission s

JOIN Team t ON s.team_id = t.team_id

WHERE t.education_level ILIKE 'PhD';

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

🗄️

📥

📶

SQL

	submission_id [PK] integer	team_id integer	language_id integer	problem_code text	verdict text	code text	submission_time timestamp without time zone
1	16	2	5	A	OK	// code for A by team 2	2025-06-14 11:35:00
2	17	2	4	A	TLE	// code for A by team 2	2025-06-14 11:19:00
3	18	2	3	B	RE	// code for B by team 2	2025-06-14 11:48:00
4	19	2	2	B	RE	// code for B by team 2	2025-06-14 12:19:00
5	20	2	1	B	WA	// code for B by team 2	2025-06-14 10:21:00
6	21	2	4	C	WA	// code for C by team 2	2025-06-14 10:58:00
7	22	2	2	C	RE	// code for C by team 2	2025-06-14 10:33:00
8	23	2	2	D	OK	// code for D by team 2	2025-06-14 11:05:00
9	24	2	5	D	TLE	// code for D by team 2	2025-06-14 10:42:00
10	25	2	3	E	OK	// code for E by team 2	2025-06-14 10:00:00
11	26	2	5	E	TLE	// code for E by team 2	2025-06-14 11:14:00
12	27	2	3	E	OK	// code for E by team 2	2025-06-14 11:33:00
13	28	2	4	F	RE	// code for F by team 2	2025-06-14 12:06:00
14	29	2	3	F	WA	// code for F by team 2	2025-06-14 11:36:00
15	30	2	4	F	RE	// code for F by team 2	2025-06-14 10:00:00
16	31	2	2	G	OK	// code for G by team 2	2025-06-14 10:18:00
17	32	3	5	A	TLE	// code for A by team 3	2025-06-14 10:29:00
18	33	3	4	A	RE	// code for A by team 3	2025-06-14 11:44:00

Total rows: 455

Query complete 00:00:00.180

Рис 2.30 Виконання запиту на отримання результатів за рівнем освіти

22. Фільтрація за мовою

QueryQuery History

1 SELECT s.*

2 FROM Submission s

3 JOIN Language l ON s.language_id = l.language_id

4 WHERE l.name ILIKE 'Go';

5

Data OutputMessagesNotifications

≡+📄⌵🗑️🗑️📄⬇️📶SQL

	submission_id [PK] integer	team_id integer	language_id integer	problem_code text	verdict text	code text	submission_time timestamp without time zone
1	1	1	4	A	RE	// code for A by team 1	2025-06-14 12:35:00
2	8	1	4	D	WA	// code for D by team 1	2025-06-14 10:14:00
3	13	1	4	F	RE	// code for F by team 1	2025-06-14 12:47:00
4	17	2	4	A	TLE	// code for A by team 2	2025-06-14 11:19:00
5	21	2	4	C	WA	// code for C by team 2	2025-06-14 10:58:00
6	28	2	4	F	RE	// code for F by team 2	2025-06-14 12:06:00
7	30	2	4	F	RE	// code for F by team 2	2025-06-14 10:00:00
8	33	3	4	A	RE	// code for A by team 3	2025-06-14 11:44:00
9	41	3	4	E	OK	// code for E by team 3	2025-06-14 12:03:00
10	42	3	4	E	TLE	// code for E by team 3	2025-06-14 10:57:00
11	50	4	4	B	TLE	// code for B by team 4	2025-06-14 10:49:00
12	54	4	4	C	RE	// code for C by team 4	2025-06-14 10:04:00
13	62	5	4	A	OK	// code for A by team 5	2025-06-14 12:37:00
14	67	5	4	D	WA	// code for D by team 5	2025-06-14 12:30:00
15	68	5	4	E	TLE	// code for E by team 5	2025-06-14 10:52:00
16	76	6	4	B	RE	// code for B by team 6	2025-06-14 11:00:00
17	83	6	4	E	WA	// code for E by team 6	2025-06-14 11:24:00
18	90	6	4	G	OK	// code for G by team 6	2025-06-14 10:35:00

Total rows: 260Query complete 00:00:00.142

Рис 2.31 Виконання запиту на фільтрацію за мовою

23. Статистика по вердикту

The screenshot shows a SQL query editor with two tabs: 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query. Below the query, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with two columns: 'count' and 'bigint'. The first row of data shows a count of 360.

```
1 SELECT COUNT(*) FROM Submission
2 WHERE verdict = 'OK';
3
```

	count bigint
1	360

Рис 2.32 Виконання запиту на статистику по вердикту

24. Таблиця лідерів

The screenshot shows a SQL query editor with two tabs: 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query. Below the query, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with five columns: 'team_id', 'team_name', 'institution_name', and 'ok_count'. The first row of data shows team_id 39, team_name 'Team #39', institution_name 'ONU', and ok_count 9.

```
1 SELECT
2     t.team_id,
3     t.name AS team_name,
4     i.name AS institution_name,
5     COUNT(*) AS ok_count
6 FROM Submission s
7 JOIN Team t ON s.team_id = t.team_id
8 JOIN Institution i ON t.institution_id = i.institution_id
9 WHERE s.verdict = 'OK'
10 GROUP BY t.team_id, t.name, i.name
11 ORDER BY ok_count DESC
12 LIMIT 10;
```

	team_id integer	team_name text	institution_name text	ok_count bigint
1	39	Team #39	ONU	9
2	11	Team #11	NULP	8
3	17	Team #17	CNU	7
4	1	Team #1	LNU	7
5	6	Team #6	KPI	7
6	55	Team #55	Karazin	6
7	64	Team #64	TNTU	6
8	85	Team #85	Karazin	6
9	88	Team #88	ONU	6
10	94	Team #94	Karazin	6

Рис 2.33 Виконання запиту на отримання таблиці лідерів

25. Заклади освіти

Query		Query History	
1	SELECT	*	FROM Institution
2	ORDER BY	name;	
3			

Data Output		Messages		Notifications	
≡+	📄	▼	📋	▼	🗑️
🗄️	📥				📶

	institution_id [PK] integer	name text
1	7	CNU
2	6	DNU
3	5	Karazin
4	1	KNU
5	2	KPI
6	3	LNU
7	4	NULP
8	8	ONU
9	9	SSU
10	10	TNTU

Рис 2.34 Виконання запиту на отримання закладів освіти

26. Команди

Query

Query History

1

SELECT t.*, i.name AS institution_name

2

FROM Team t

3

JOIN Institution i ON t.institution_id = i.institution_id

4

ORDER BY t.name;

5

Data Output

Messages

Notifications

≡+

▼

▼

SQL

	team_id integer	name text	education_level text	institution_id integer	institution_name text
1	1	Team #1	School	3	LNU
2	10	Team #10	School	8	ONU
3	100	Team #100	Master	10	TNTU
4	11	Team #11	Master	4	NULP
5	12	Team #12	Bachelor	3	LNU
6	13	Team #13	School	7	CNU
7	14	Team #14	School	9	SSU
8	15	Team #15	Bachelor	5	Karazin
9	16	Team #16	Master	4	NULP
10	17	Team #17	School	7	CNU
11	18	Team #18	Master	3	LNU
12	19	Team #19	School	4	NULP
13	2	Team #2	PhD	1	KNU
14	20	Team #20	Master	9	SSU
15	21	Team #21	Master	1	KNU

Total rows: 100

Query complete 00:00:00.129

Рис 2.35 Виконання запиту на отримання команд

27. Мови програмування

Query

Query History

1

▼

SELECT * FROM Language

2

ORDER BY name;

3

Data Output

Messages

Notifications

≡+

📄

▼

📋

🗑️

🗄️

⬇️

	language_id [PK] integer	name text
1	1	C++
2	4	Go
3	2	Java
4	3	Python
5	5	Rust

Рис 2.36 Виконання запиту на отримання мов програмування

28. Список кульок

Query

Query History

1

SELECT

k.problem_code,

k.team_id,

t.name AS team_name,

k.submission_id,

k.awarded_at

FROM Kulki k

JOIN Team t ON t.team_id = k.team_id

ORDER BY k.awarded_at;

10

Data Output

Messages

Notifications

≡

📄

▼

📄

▼

🗑️

📦

⬇️

📈

SQL

	problem_code text	team_id integer	team_name text	submission_id integer	awarded_at timestamp without time zone
1	E	2	Team #2	25	2025-06-15 01:25:31.269511
2	F	8	Team #8	120	2025-06-15 01:25:31.269511
3	D	24	Team #24	354	2025-06-15 01:25:31.269511
4	A	29	Team #29	423	2025-06-15 01:25:31.269511
5	C	48	Team #48	696	2025-06-15 01:25:31.269511
6	G	87	Team #87	1239	2025-06-15 01:25:31.269511
7	B	93	Team #93	1302	2025-06-15 01:25:31.269511

Рис 2.37 Виконання запиту на отримання списока кульок

29. Виявлення нових кульок

The screenshot shows a SQL query editor with a 'Query' tab selected. The query is as follows:

```
3 JOIN (  
4     SELECT problem_code, MIN(submission_time) AS first_ok_time  
5     FROM Submission  
6     WHERE verdict = 'OK'  
7     GROUP BY problem_code  
8 ) AS first_ok ON s.problem_code = first_ok.problem_code  
9                AND s.submission_time = first_ok.first_ok_time  
10 WHERE s.verdict = 'OK'  
11     AND s.problem_code NOT IN (  
12         SELECT problem_code FROM Kulki  
13     );  
14
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the following columns:

problem_code	team_id	submission_id
text	integer	[PK] integer

Рис 2.38 Виконання запиту на виявлення нових кульок

30. Присвоїти кульки

The screenshot shows a SQL query editor with a 'Query' tab selected. The query is as follows:

```
4 JOIN (  
5     SELECT problem_code, MIN(submission_time) AS first_ok_time  
6     FROM Submission  
7     WHERE verdict = 'OK'  
8     GROUP BY problem_code  
9 ) AS first_ok ON s.problem_code = first_ok.problem_code  
10                AND s.submission_time = first_ok.first_ok_time  
11 WHERE s.verdict = 'OK'  
12     AND s.problem_code NOT IN (  
13         SELECT problem_code FROM Kulki  
14     );  
15
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, showing the following message:

```
INSERT 0 0  
  
Query returned successfully in 93 msec.
```

Рис 2.39 Виконання запиту на присвоєння кульок

3. 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 ІСПС-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. Стилзація 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

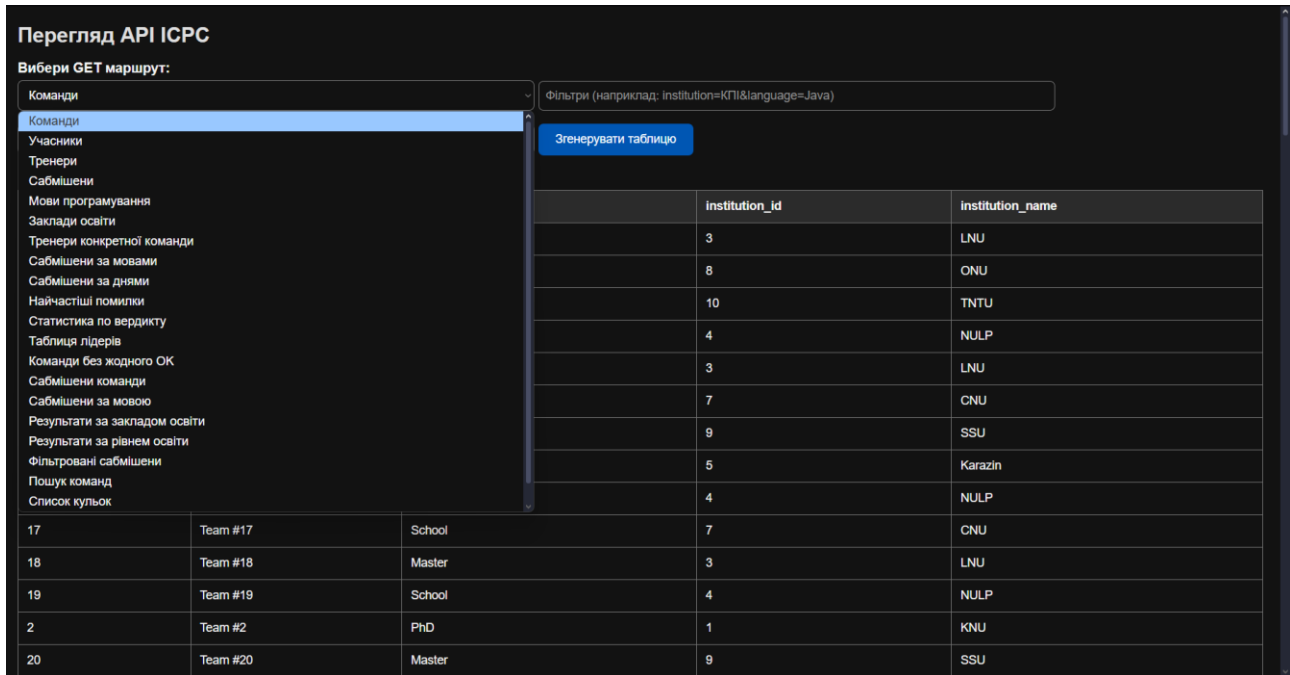


Рис. 3.1 Головне меню додатку

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.

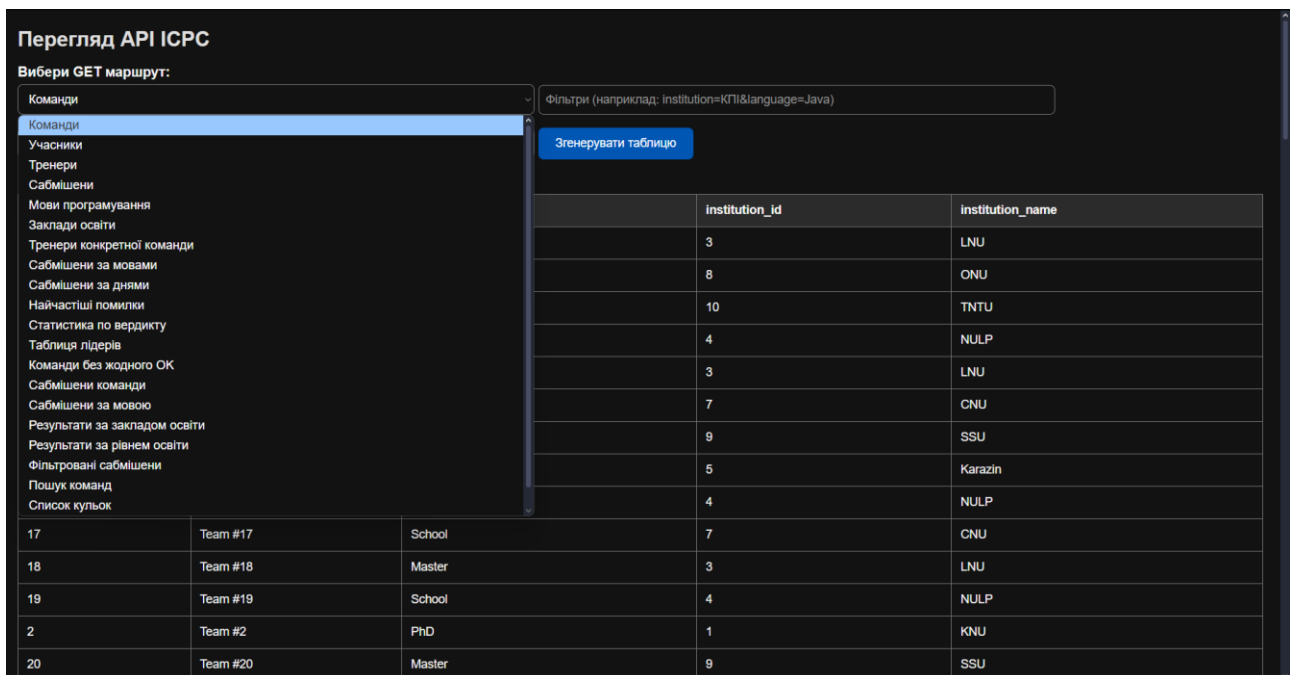


Рис. 3.2 Розгорнутий дропдаун гет запитів

Перший блок відповідає за виконання GET-запитів до бази даних. У ньому реалізовано випадуючий 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.

Перегляд API ICPC

Вибери GET маршрут:

Команди

team_id	name	education_level	institution_id	institution_name
1	Team #1	School	3	LNU
10	Team #10	School	8	ONU
100	Team #100	Master	10	TNTU
11	Team #11	Master	4	NULP
12	Team #12	Bachelor	3	LNU
13	Team #13	School	7	CNU
14	Team #14	School	9	SSU
15	Team #15	Bachelor	5	Karazin
16	Team #16	Master	4	NULP
17	Team #17	School	7	CNU
18	Team #18	Master	3	LNU
19	Team #19	School	4	NULP
2	Team #2	PhD	1	KNU
20	Team #20	Master	9	SSU

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.

87	Team #87	PhD	10	TNTU
88	Team #88	Bachelor	8	ONU
89	Team #89	PhD	4	NULP
9	Team #9	Bachelor	7	CNU
90	Team #90	PhD	1	KNU
91	Team #91	PhD	9	SSU
92	Team #92	PhD	6	DNU
93	Team #93	PhD	10	TNTU
94	Team #94	Master	5	Karazin
95	Team #95	PhD	4	NULP
96	Team #96	Master	3	LNU
97	Team #97	Bachelor	10	TNTU
98	Team #98	PhD	10	TNTU
99	Team #99	School	5	Karazin

Додати учасника (POST)
Оновити учасника (PUT)
Видалити учасника (DELETE)
Додати тренера (POST)
Оновити тренера (PUT)
Видалити тренера (DELETE)
Присвоїти кулью (PUT)
Додати учасника (POST)

Введіть JSON дані або ID для DELETE

Виконати

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.

90	Team #90	PhD	1	KNU
91	Team #91	PhD	9	SSU
92	Team #92	PhD	6	DNU
93	Team #93	PhD	10	TNTU
94	Team #94	Master	5	Karazin
95	Team #95	PhD	4	NULP
96	Team #96	Master	3	LNU
97	Team #97	Bachelor	10	TNTU
98	Team #98	PhD	10	TNTU
99	Team #99	School	5	Karazin

Редагування / Додавання / Видалення
Вибери дію:

{
"name": "Іван Петренко",
"email": "ivan.petrenko@example.com",
"team_id": 3
}

Виконати

Додати учасника (POST)

Відповідь сервера:

```

{
  "participant_id": 302,
  "name": "Іван Петренко",
  "email": "ivan.petrenko@example.com",
  "team_id": 3
}

```

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 спроектована 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 ІСПС-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 спроектовано 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. Базы даних. Проектування, реалізація та використання. — Львів: Видавництво Львівської політехніки, 2019.
3. Date С.Ј. 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>

- 15.W3Schools. HTML, CSS та JavaScript Підручники. — <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.