

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

КУРСОВИЙ ПРОЕКТ

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: **Моніторингова система контролю успішності учнів**
(студентів)

Студент

групи КП-81

Ганін Владислав В'ячеславович

(ПІБ)

(підпис)

Викладач

к.т.н, доцент кафедри

СПіСКС

Петрашенко А.В.

(підпис)

Захищено з оцінкою _____

Київ – 2021

Анотація

Метою розробки даного курсового проекту є набуття практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з пост реляційними базами даних, а також здобуття навичок оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. У результаті виконання курсового проекту було опановано навик розробляти програмне забезпечення для пост реляційних баз даних, володіння основами використання СУБД, а також інструментальними засобами аналізу великих обсягів даних.

Темою даного курсового проекту є створення моніторингової системи контролю успішності учнів (студентів). У документі викладена актуальність та проблематика аналізу великого обсягу даних, аналіз використаного інструментарію (опис мови програмування, використаних бібліотек та СУБД), описана структура бази даних, опис розробленого програмного забезпечення (загальний, опис модулів та основних алгоритмів роботи), аналіз функціонування засобів масштабування, та опис результатів проведеного аналізу.

Результатами даного проекту стали діаграми та графіки, що зображають результати дослідження успішності навчання учнів (студентів). З ними можна ознайомитися в додатку А.

Зміст

Анотація	1
Зміст	2
Вступ	3
Аналіз інструментарію для виконання курсового проект	5
Аналіз СКБД	5
Обґрунтування вибору мови програмування	8
Обґрунтування вибору бібліотек і фреймворків	8
Структура бази даних	9
Опис програмного забезпечення	10
Загальна структура програмного забезпечення	10
Опис модулів програмного забезпечення	10
Опис основних алгоритмів роботи	11
Аналіз функціонування засобів масштабування	12
Загальний алгоритм додавання нового кластеру шардингу, який складається з кількох серверів реплікації	13
Загальний алгоритм додавання нового вузла реплікації серверів конфігурації	14
Висновки	16
Література	17
Додаток А	18
Додаток Б	20

Вступ

У ході роботи була створена система контролю успішності учнів (студентів). За dataset були взяті дані у відкритому доступі з результатами успішності більш ніж 1000 учнів.

Актуальність. Особливої уваги у сучасному світі набуває якість навчання. Якісний рівень освіти забезпечується за допомогою моніторингу - систематичних процедур збору даних про важливі аспекти освіти на всіх рівнях з метою безперервного відстеження її стану та прогнозу розвитку. Очевидно, що система контролю успішності учнів (студентів) є, без перебільшення, дуже впливовою на дану галузь.

Проект збирає результати успішності, що знаходяться у відкритому доступі, обробляє їх, та видає середні показники балів учнів чи студентів навчального закладу. Також система аналізує тренди та моду різних параметрів успішності учнів (студентів) та вираховує залежність оцінок за курсові проекти від результатів екзаменів проведених в різні періоди навчального року.

Мета розробки:

Створення програмного забезпечення, що забезпечує роботу наведених далі пунктів:

1. Попередня обробка даних

- Засоби генерації даних. Розроблення утиліти для збору інформації про успішність навчання учнів.
- Засоби фільтрації і валідації даних. Розроблення додаткового функціоналу у вищезазначеній утиліті задля корегування отриманих даних та переходу до їх подальшої обробки та структуризації.

2. Основний модуль: процесу. Встановлення залежності між результатами екзаменів та курсових робіт. Складання таблиці успішності учнів сортованих за повним ім'ям.

3. Забезпечення масштабування та реплікації даних: шардинг та створення репліка сетів у Mongo DB

Дані для аналізу були запозичені із відкритого сховища даних за посиланням <https://www.kaggle.com/alejandropaige/student-grades-record>.

Аналіз інструментарію для виконання курсового проект

Аналіз СКБД

В процесі виконання цього курсового проекту перед нами встала потреба кешувати та зберігати дані про вакансії між запусками аналізатора. Кожного разу читати дані із CSV - файлів є дуже дорогою операцією, тож було прийняте рішення використати СКБД. В якості СКБД були розглянуті варіанти: PostgreSQL, MongoDB. З порівняльною характеристикою цих СУБД можна ознайомитися в таблиці 1.

таблиця 1. Порівняльна характеристика СКБД

Критерій порівняння	Назва СКБД		
	MongoDB	PostgreSQL	CassandraDB
Має відкритий вихідний код	так	так	так
Схема даних	динамічна	статична і динамічна	статична і динамічна
Підтримка ієрархічних даних	так	так (з 2012)	ні
Реляційні дані	ні	так	так
Транзакції	ні	так	так

Атомарність операцій	всередині документа	по всій БД	всередині партиції
Мова запитів	JSON/JavaScript	SQL	CQL
Найлегший спосіб масштабування	горизонтальний	вертикальний	горизонтальний
Підтримка шардингів	так	так (важка конфігурація)	так (може зберігати партиції на різних машинах)
Приклад використання	Великі дані (мільярди записів) з великою кількістю паралельних оновлень, де цілісність і узгодженість даних не потрібно.	Транзакційні і операційні програми, вигода яких в нормалізованому формі, об'єднаннях, обмеження даних і підтримки транзакцій.	Багато запитів на запис/читання у одиницю часу, до даних можна задіяти партиціювання за ключем, дані мають лише первинні індекси
Наявність бібліотек для мови програмування Node 12	так	так	так
Підтримка реплікації	так, автоматичне переобрання головного процесу	За принципом master-slave	так, через партиціювання
Засіб збереження та відновлення даних	mongodump	pg_dump	не має окремого доданка, виконується засобами CQL

Форма збереження даних	документи JSON	таблиця	таблиця
------------------------	----------------	---------	---------

За результатами порівнянні цих СУБД було прийнято рішення зупинитися на NoSQL рішеннях. Оскільки вона чудово поєднує в собі переваги неструктурованих баз даних та простоту використання горизонтального масштабування. Крім цього класичним прикладом використання NoSQL СУБД є системи збору та аналізу даних, до яких можна застосувати індексування за первинними та вторинними ключами.

Ця база даних є об'єктно орієнтованою та дозволяє зберігати великі масиви неструктурованих даних. На відміну від SQL баз даних ми можемо зберігати дані у “сирому” об'єктному вигляді, який використовується програмою та є більш близьким за структурою до моделі даних, яку буде використовувати ПЗ написане з використанням мови програмування JavaScript. Це пришвидшить збір, збереження та отримання даних програмним забезпеченням. Оскільки MongoDB є представником NoSQL баз даних, вона не потребує жорсткої схеми даних, що дозволяє пришвидшити процес розробки та зробити його більш гнучким. Окрім цього дана СУБД підтримує горизонтальне масштабування за допомогою шардингу з метою зменшення навантаження на кожен окремий вузол шляхом розподілення навантаження між ними всіма.

Нижче наведено перелік основних переваг:

- Підтримка ієрархічних даних
- Динамічна схема
- Швидкість запису у колекцію
- Швидкість читання із колекції
- Простота масштабування та відновлення даних

Обґрунтування вибору мови програмування

Мовою програмування для ПЗ було обрано JavaScript. Оскільки це сильна опціонально типізована мова програмування. Це дозволяє самостійно обирати типізацію і поєднувати швидкість розробки ПЗ із можливістю в майбутньому додати типи до модулів програми.

Обґрунтування вибору бібліотек і фреймворків

Використані бібліотеки:

chartjs — це безкоштовна бібліотека JavaScript з відкритим кодом для візуалізації даних, яка підтримує 8 типів діаграм: смуга, лінія, область, пиріг (пончик), міхур, радіолокатор, полярний та розкиданий. Створена лондонським веб-розробником Ніком Дауні в 2013 році, тепер вона підтримується громадою і є другою за популярністю бібліотекою графіків JS на GitHub за кількістю зірок після D3.js, що вважається значно простішою у використанні, хоча і менш налаштованою, ніж остання. Chart.js відображається в canvas HTML5 і широко висвітлюється як одна з найкращих бібліотек візуалізації даних. Вона доступна за ліцензією MIT.

mongoose — це бібліотека JavaScript, яка створена для взаємодії з базами даних включаючи MongoDB.

simple-statistics — це бібліотека JavaScript, яка реалізує статистичні методи. Вона допомагає програмістам використовувати силу статистики, а статистики розуміють код. Бібліотека вичерпно задокументована, написана у простому та доброзичливому стилі та ретельно перевірена.

Express.js, або просто Express — програмний каркас розробки серверної частини веб-застосунків для Node.js, реалізований як вільне і відкрите програмне забезпечення під ліцензією MIT. Він спроектований для створення веб-застосунків і API. Де-факто є стандартним каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описує його як створений на основі написаного на мові Ruby каркаса Sinatra, маючи на увазі, що він мінімалістичний, але має велику кількість плагінів, що підключаються.

Структура бази даних

База даних складається з однієї колекції, в якій зберігаються відомості про параметри атмосферного повітря в місті. Загальна структура документа в базі даних приведена у таблиці 2.

таблиця 2. Опис властивостей документа у базі даних

Назва властивості	Тип	Опис
_id	ObjectId	Ідентифікатор запису
first_name	String	Ім'я учня (студента)
last_name	String	Прізвище учня (студента)
mid_term_exam	Number	Оцінка за середньострокові екзамени
final_term_exam	Number	Оцінка за випускні екзамени
cw1	Number	Оцінка за першу курсову роботу
cw2	Number	Оцінка за другу курсову роботу
total_points	Number	Загальний бал
average	Number	Середній бал
grade	String	Оцінка в форматі американської 5-ти бальної системи (F - A)

Опис програмного забезпечення

Загальна структура програмного забезпечення

Структура ПО є класичною для даного стеку бібліотек.

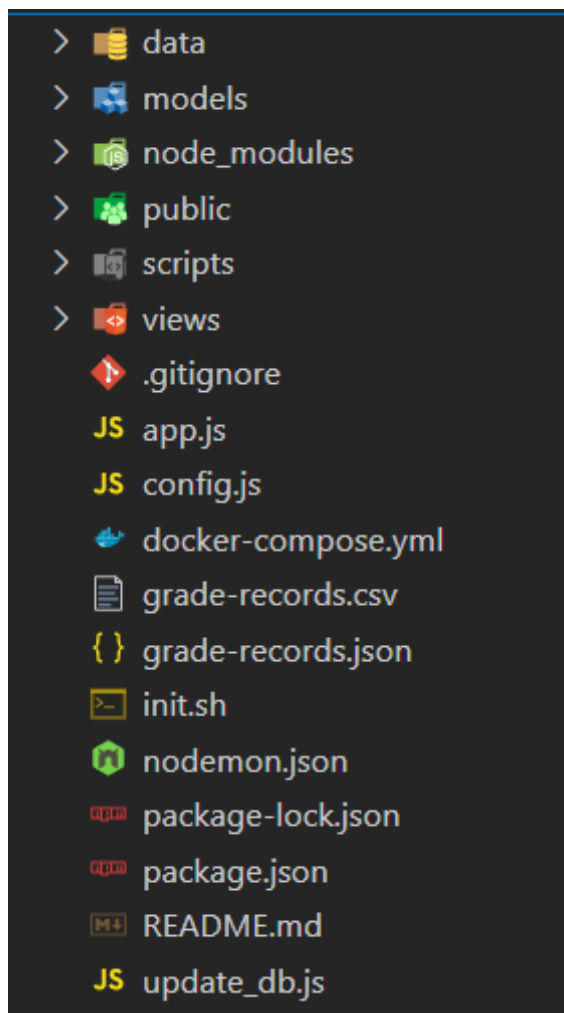


рис 1. Файлова структура проекту

Усі логічно розподілені частини програми виокремлені у свій файл та структуровані у окремій директорії.

Опис модулів програмного забезпечення

app.js — модуль, що відповідає за основу серверної логіки додатку та керує іншими модулями. Встановлює кінцеві точки взаємодії, роботу з базою даних, а також є відповідальним за експорт/імпорт даних із бази даних додатку.

`config.js` — модуль, що відповідає за конфігурацію програмного забезпечення.

`student.js` — модуль, що відповідає за взаємодію додатку із сутностями бази даних додатку.

`public` — модуль, що відповідає за візуальну складову веб додатку: стилі, медіа файли та скрипти анімації та візуалізації даних.

`scripts` — скриптовий модуль, проводить первинне налаштування бази даних та її масштабуючих можливостей (шардинг та реплікація даних).

`views` — модуль, що складає собою файли розмітки веб додатку.

Опис основних алгоритмів роботи

1. `app.js` встановлює оновлення даних кожних 30 секунд після запуску ПО.
2. `update_db.js` після отримання даних валідує їх на основі основних параметрів, а саме: довжина імені та фамілії студента в символах, реально можлива оцінка студента по кожній з можливих робіт.
3. `browser-charts.js` проводить геометричні обчислення на основі вхідних даних за допомогою алгоритмів “simple-statistics”, щоб побудувати графіки.

Аналіз функціонування засобів масштабування

В якості засобів масштабування було обрано реплікацію та шардинг.

Реплікація - це процес синхронізації даних на декількох серверах. Даний механізм зменшує витрати ресурсів і збільшує доступність даних, копії яких зберігаються на різних серверах. Реплікація захищає базу даних (далі - БД) від втрати єдиної сервера і дозволяє зберегти дані в разі технічної несправності на одному з серверів. У MongoDB реплікація досягається шляхом використання набору копій (replica set). Це група примірників mongod, який зберігають однакові набори даних. У копії один вузол - це ключовий вузол, який отримує всі операції запису. Всі інші вузли - вторинні, приймають операції з першого, таким чином, зберігаючи такі ж записи, як і первинний вузол. Набір копій може мати тільки один первинний вузол.

Шардінг - це процес зберігання документів на декількох серверах і це спосіб, яким MongoDB справляється з великими даними. З ростом кількості даних, один сервер не може зберігати всі дані, ні записувати їх, ні давати до них доступ. Шардінг вирішує проблему шляхом горизонтального масштабування. Завдяки даному механізму ми можемо підключати додаткові сервери для зберігання, записи і читання даних.

Для емуляції існування багатьох серверів із СКБД було використано Docker машину. Конфігурація docker-compose наведено у Додатку А.

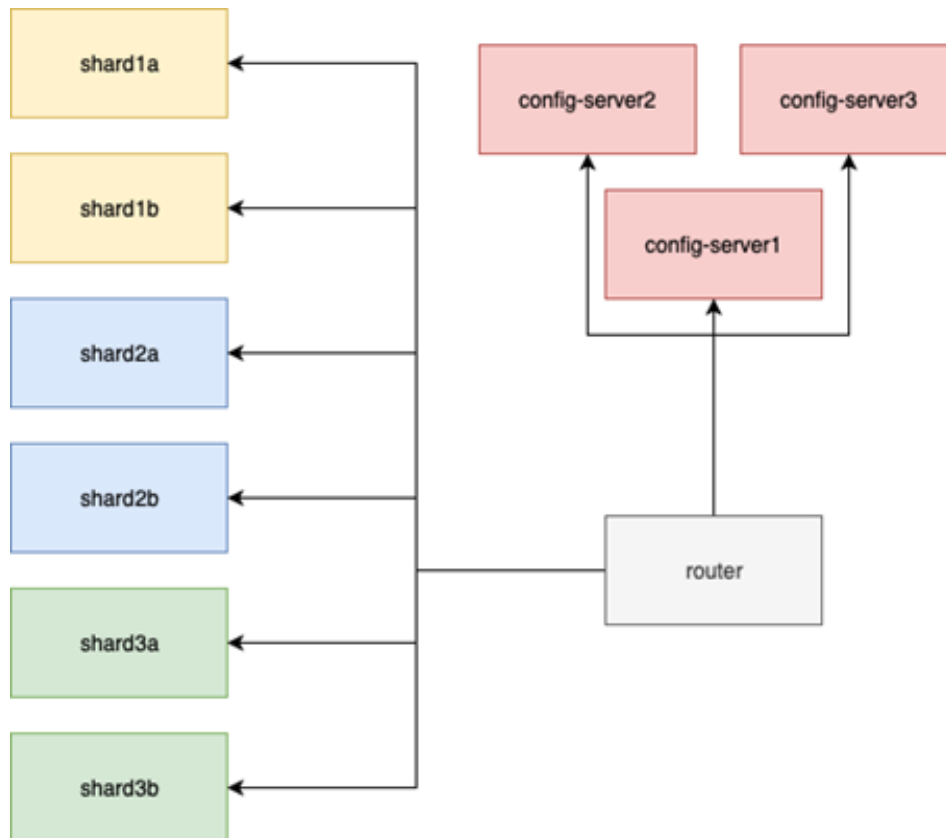


рис 2.Схема використаної моделі шардингу MongoDB у даному ПЗ

Загальний алгоритм додавання нового кластеру шардингу, який складається з кількох серверів реплікації

1. Запустити на виконання один або декілька процесів mongod, які стануть вузлами шардингу за допомоги команди у таблиці 3.

таблиця 3. Скрипт запуску нових вузлів шардингу

```
mongod --port <PORT> --shardsvr --replSet <SHARD_1_ID> --noprealloc --smallfiles
--oplogSize 16

mongod --port <PORT> --shardsvr --replSet <SHARD_2_ID> --noprealloc --smallfiles
--oplogSize 16
```

2. Дочекатися їх ініціалізації, під'єднатися до процесу mongod одного з них та виконати команду наведену у таблиці 4.

таблиця 4. Скрипт створення нових вузлів шардингу

```
rs.initiate(  
  {  
    _id: "<SHARD_1_ID>",  
    version: 1,  
    members: [  
      { _id: 0, host : "<SHARD_HOST_1>:<PORT>" },  
      { _id: 1, host : "<SHARD_HOST_1>:<PORT>" },  
    ]  
  }  
)
```

3. Під'єднатися до консолі процесу `mongoose` сервера роутера та виконати команду, наведену у таблиці 5. Це зареєструє новий сервер конфігурації шардингу у сервері маршрутизації.

таблиця 5. Скрипт додавання нового кластеру до існуючого шардингу

```
sh.addShard("<SHARD_HOST_1>/<SHARD_1_ID>:<PORT>")  
  
sh.addShard("<SHARD_HOST_1>/<SHARD_2_ID>:<PORT>")
```

Загальний алгоритм додавання нового вузла реплікації серверів конфігурації

Запустити на виконання один або декілька процесів `mongod`, які стануть вузлами реплікації.

1. Дочекатися їх ініціалізації, під'єднатися до процесу `mongod` одного з них та виконати команду наведену у таблиці 6.

таблиця 6. Скрипт додавання нового серверу конфігурації до реплікації

```
rs.add( { host: "<hostnameNew>:<portNew>", priority: 0, votes: 0 } )
```

Опис результатів аналізу предметної галузі

В результаті виконання курсового проекту було проаналізовано дані про успішність навчання учнів (студентів). Було отримано наступні дані:

- Проаналізована залежність між оцінками за екзаменами та курсові роботи;
- Складена таблиця результатів навчання учнів сортованих за повним ім'ям;
- Знайдені мода та медіана оцінок різних типів;
- Знайдена лінія регресії оцінки за середньострокові екзамени та за другу курсову роботу;
- Розроблений графічний інтерфейс для візуалізації отриманих даних;
- Додана функція експорту та імпорту бази даних.

Висновки

В процесі виконання даного курсового проекту було отримано практичні навички обробки великих масивів даних за допомогою мови програмування JavaScript та СУБД MongoDB.

Було проаналізовано сучасні методи та інструменти для роботи із великими даними, знайдено гарно працюючу комбінацію із мови програмування JavaScript та бібліотек до нього. Було складено порівняльну таблицю із трьох баз даних, які найчастіше використовують для роботи із часовими рядами. На основі цих даних було обрано в якості СУБД NoSQL рішення MongoDB.

Для забезпечення горизонтального масштабування було використано засоби MongoDB, такі як: репліка сет та шардинг. Практичним шляхом було знайдено спосіб партиціювання даних часових рядів на основі хеш-функції від мітки часу. Це дозволило розподілити дані, які зберігаються в рамках однієї колекції між багатьма серверами кластерів шардингу. За допомогою реплікації було досягнуто відмовостійкості системи, в результаті чого, під час тестування ми впевнилися, що система продовжує функціонувати після виходу із ладу кількох реплік.

На основі зібраних даних було проаналізовано успішність навчання учнів (студентів). Знайдено кореляції та залежності між різними показниками. Дані аналізу приведені у Додатку Б. Текстовий опис надано в відповідному розділі цього курсового проекту.

В ході виконання даного курсового проекту було досягнуто поставленої мети: було набуто практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з NoSQL базами даних, а також були здобуті навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. У результаті виконання курсового проекту я навчився писати програмне забезпечення для NoSQL баз даних, володіти основами використання СУБД, а також інструментальними засобами аналізу великих обсягів даних, а саме відкритими бібліотеками мови JavaScript.

Література

1. Hashed sharding [Електронний ресурс]. Режим доступу до ресурсу: <https://docs.mongodb.com/manual/core/hashed-sharding/>
2. Aggregation [Електронний ресурс]. Режим доступу до ресурсу: <https://docs.mongodb.com/manual/aggregation/>
3. 5 причин, почему машинное обучение станет технологией 2018 [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://blogs.oracle.com/russia/machine-learning-2018;>
4. Сложности накопления данных для интеллектуального анализа [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: [https://habr.com/post/154787/;](https://habr.com/post/154787/)
5. PostgreSQL лучше чем MongoDB [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: [https://habr.com/post/272735/;](https://habr.com/post/272735/)
6. 27 шпаргалок по машинному обучению [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: [https://proglib.io/p/ds-cheatsheets/;](https://proglib.io/p/ds-cheatsheets/)
7. Горизонтальное масштабирование: когда и как? [Електронний ресурс]. – Режим доступу до ресурсу: <https://habr.com/company/oleg-bunin/blog/319526>

Додаток А


таблиця А1. Вміст файлу *docker-compose.yml*

docker-compose.yml

```
version: '3.7'
services:
  config01:
    image: mongo:4.0
    command: mongod --port 27017 --configsvr --replSet configserver --noprealloc
    --smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/config01:/data/db
  config02:
    image: mongo:4.0
    command: mongod --port 27017 --configsvr --replSet configserver --noprealloc
    --smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/config02:/data/db
  config03:
    image: mongo:4.0
    command: mongod --port 27017 --configsvr --replSet configserver --noprealloc
    --smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/config03:/data/db
  shard01a:
    image: mongo:4.0
    command: mongod --port 27018 --shardsvr --replSet shard01 --noprealloc
    --smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/shard01a:/data/db
  shard01b:
    image: mongo:4.0
    command: mongod --port 27018 --shardsvr --replSet shard01 --noprealloc
    --smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/shard01b:/data/db
  shard02a:
    image: mongo:4.0
    command: mongod --port 27019 --shardsvr --replSet shard02 --noprealloc
    --smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/shard02a:/data/db
  shard02b:
    image: mongo:4.0
    command: mongod --port 27019 --shardsvr --replSet shard02 --noprealloc
    --smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/shard02b:/data/db
  shard03a:
    image: mongo:4.0
    command: mongod --port 27020 --shardsvr --replSet shard03 --noprealloc
    --smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
```

```
    - ./data/shard03a:/data/db
shard03b:
  image: mongo:4.0
  command: mongod --port 27020 --shardsvr --replSet shard03 --noprealloc
--smallfiles --oplogSize 16
  volumes:
    - ./scripts:/scripts
    - ./data/shard03b:/data/db
router:
  image: mongo:4.0
  command: mongos --port 27017 --configdb
configserver/config01:27017,config02:27017,config03:27017 --bind_ip_all
  environment:
    USER: user
    PSW: psw
    DB: mydb
  ports:
    - "27017:27017"
  volumes:
    - ./scripts:/scripts
    - ./data/router:/data/db
  depends_on:
    - config01
    - config02
    - config03
    - shard01a
    - shard01b
    - shard02a
    - shard02b
    - shard03a
    - shard03b
```

Додаток Б



[Home](#)
[Students](#)
[Graphics](#)

[Export](#)
[Import](#)

<*/export to reserve

<*/import to restore

рис Б1.


[Home](#)
[Students](#)
[Graphics](#)

[Export](#)
[Import](#)

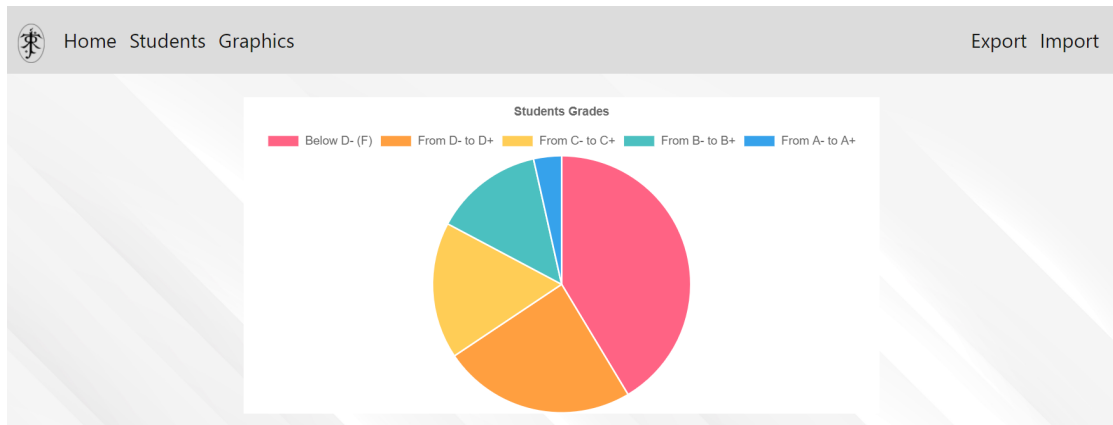
Students Mode and Median

#	Parameter	Mode	Median
1	Mid-term exams	45.00	65.00
2	Final exam	45.00	65.00
3	CW 1	45.00	65.00
4	CW 2	65.00	65.00
5	Total Points	216.00	251.00
6	Student Average	54.00	62.75
7	Grade	F	D+

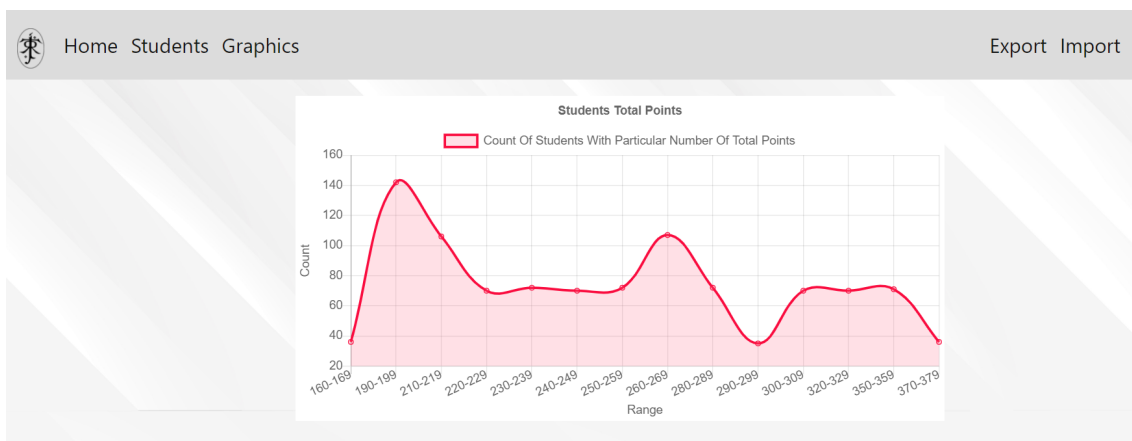
Students Grades

#	Full name	Mid-term exams(0-100)	Final exam(0-100)	CW 1(0-100)	CW 2(0-100)	Total Points(0-400)	Student Average(0-100)	Grade(F-A+)
8	Aaron Banford	89	45	89	45	268	67	D+
9	Abagael Wasmer	44	78	44	78	244	61	D-

рис Б2.



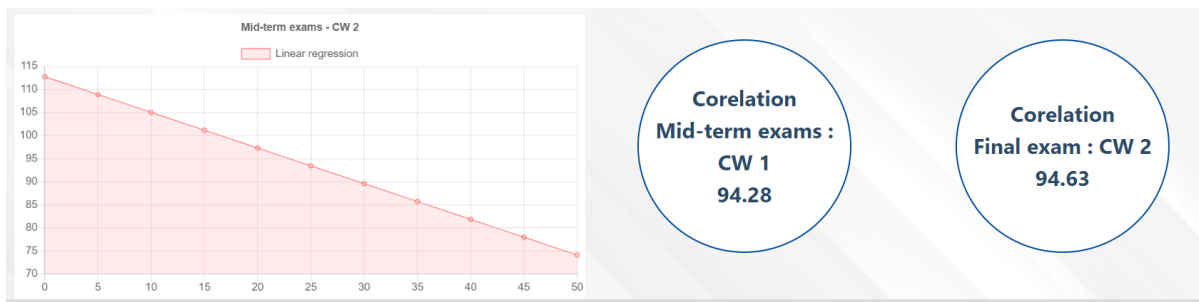
puc B3.



puc B4.



puc B5.



pus Б6.