

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

КУРСОВА РОБОТА

з дисципліни
«ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ»

на тему: «БД проектної організації»

Студента 3 курсу, групи КА-84
спеціальності 124 «Системний аналіз»
Забельський Володимир Віталійович
(прізвище, ім'я, по-батькові)

Керівник

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

Кількість балів: _____

Члени комісії

_____	_____
(підпис)	(науковий ступінь, вчене звання, прізвище та ініціали)
_____	_____
(підпис)	(науковий ступінь, вчене звання, прізвище та ініціали)
_____	_____
(підпис)	(науковий ступінь, вчене звання, прізвище та ініціали)

Київ - 2021

Анотація

Ця курсова робота створена як приклад реалізації взаємовідносин між backend і web інтерфейс. У даній роботі реалізуються базові завдання роботи з базами даних. Дана робота складається з трьох частин, кожна з яких є автономним етапом побудови повної архітектури. Дана реалізація буде корисна для backend розробників, так як є наочним прикладом повного взаємодії кожних частин сервісу.

Annotation

This course work is created as an example of the implementation of the relationship between the backend and the web interface. In this work, the basic tasks of working with databases are implemented. This work comprised of three parts, and each is a stand-alone step in building a complete architecture. This implementation will be useful for backend developers, as it is a clear example of the complete interaction of each part of the service.

ЗМІСТ

ВСТУП	4
1. Постановка задачі	5
2. Архітектура та інформаційне забезпечення БД	7
2.1 Аналіз функціонування та організаційні засади підприємства.....	7
2.2. Проектування структури бази даних	8
2.3 Життєві цикли бази даних	10
3. Реалізація програмної взаємодії з БД	11
3.1 Інструкція користувача	11
3.2 Реалізація механізмів БД	14
3.2.1 SQL-запити.....	14
3.2.2 Процедури і функції	17
3.2.3 Тригери	17
3.3 Вимоги до апаратних і програмних засобів.....	18
3.4 Випробування розроблених програм.....	18
3.5 Опис тестової бази даних.....	23
ВИСНОВКИ	24
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	25
Додаток А Текст програми	26

ВСТУП

Актуальність: У наш час інформаційна система більшості організацій та підприємств будуються на основі баз даних. Вони використовуються для зберігання даних та представлення їх у зручній та практичній формі. В цій курсовій роботі проектується база даних проектної організації, яка включає в себе всі базові елементи БД, тож добре підходить для освоєння їх проектування та розробки, та створимо зручний інтерфейс для користувача, тобто з візуалізуємо ВЕБ-інтерфейс.

Мета: розробка та створення бази даних та інтерфейсу для автоматизованої інформаційної системи проектної організації, що відповідає всім поставленим вимогам, а також створення зручного веб-інтерфейсу для взаємодії з базою даних, та виклик запитів та процедур для доступу до даних.

Завдання: Спроекувати базу даних та підготувати усі необхідні запити та процедури для роботи з нею, зв'язати та створити зі зручним веб-інтерфейсом для користування з базою даних.

Практичне значення: Відточення навичок SQL-програмування. Написання коду мовами Java, JS, також поглиблення у знаннях таких фреймворків як Spring Boot.

Програмне забезпечення: При виконанні роботи було використано таке програмне забезпечення: середовище розробки MySQL Workbench; операційна система Windows 10; база даних MySQL, VScode, Postman, IntelliJ IDEA, Opera, Google Chrome.

1. Постановка задачі

Проектна організація представлена наступними категоріями працівників: конструктори, інженери, техніки, лаборанти, інший обслуговуючий персонал, кожен з яких може мати притаманні лише їй атрибути. Наприклад, конструктор характеризується числом авторських свідоцтв, техніки - обладнанням, яке вони можуть обслуговувати, інженер або конструктор може управляти договором або проектом тощо. Співробітники розділені на відділи, керовані начальником так, що кожен співробітник числиться тільки в одному відділі.

В рамках укладених проектною організацією договорів з замовниками виконуються різного роду проекти, причому за одним договором може виконуватися більше одного проекту, і один проект може виконуватися для кількох договорів. Сумарна вартість договору визначається вартістю всіх проектних робіт, з яких складається договір. Кожен договір і проект має керівника і групу співробітників, які виконують цей договір або проект, причому це можуть бути співробітники не тільки одного відділу. Проекти виконуються з використанням різного обладнання, частина якого належить окремим відділам, а частина є колективною власністю проектної організації, при цьому в процесі роботи обладнання може передаватися з відділу у відділ. Для виконання проекту обладнання надається групі, що працює над проектом,

Для виконання ряду проектів підрядна організація може залучати субпідрядні організації, передаючи їм обсяги робіт.

Ведеться облік кадрів, облік виконання договорів і проектів, вартісний облік усіх виконаних робіт.

Види запитів в інформаційній системі:

1. Отримати дані про склад зазначеного відділу або всієї організації повністю, з вказаної категорії працівників, за віковим складом.
2. Отримати перелік керівників відділів.
3. Отримати перелік договорів або проектів, які виконуються в даний момент або в період зазначеного інтервалу часу.

4. Отримати інформацію про те, які проекти виконуються (виконувалися) в рамках зазначеного договору та договору підтримуються зазначеними проектами.
5. Отримати дані про вартість виконаних договорів (проектів) протягом зазначеного періоду часу.
6. Отримати дані про розподіл обладнання на даний момент або на деяку зазначену дату.
7. Відомості про використання обладнання зазначеними проектами (договорами).
8. Відомості про участь вказаного співробітника або категорії співробітників в проектах (договорах) за певний період часу.
9. Отримати перелік і вартість робіт, виконаних субпідрядними організаціями.
10. Отримати дані про чисельність і склад співробітників в цілому і за окремими категоріями, які беруть участь в даному проекті.
11. Отримати дані про ефективність використання обладнання (обсяги проектних робіт, виконаних з використанням того чи іншого обладнання).
12. Отримати відомості про ефективність договорів (вартість договорів співвіднести з витраченим часом або вартість з урахуванням залучених людських ресурсів).
13. Отримати дані про чисельність і склад співробітників в цілому і за окремими категоріями, які беруть участь в проектах за вказаний період часу.
14. Отримати відомості про ефективність проектів (вартість договорів співвіднести з витраченим часом або вартість з урахуванням залучених людських ресурсів).

2. Архітектура та інформаційне забезпечення БД

2.1 Аналіз функціонування та організаційні засади підприємства

В завданні представлено проектну організацію. Проаналізувавши запити, що необхідно буде реалізувати, можна зробити висновки, які дані нам потрібно мати у БД, також це дає нам можливість представити та приблизно спроектувати наш майбутній веб-інтерфейс для даної БД.

У даній роботі база даних складається з 14 таблиць, головна з яких Worker, Client і додаткові які або виконують функцію зв'язку “many-to-many”, або служать для збереження статичних даних.

Спершу, необхідно ввести правильну інформацію у відповідні таблиці. Вибираючи відповідний результат, який нам необхідно отримати, в середовищі програмного забезпечення потрібно вводити потрібні запити. Програма повертається в початковий етап при закритті всіх вкладок із запитам SQL. Кожне значення відповідає відповідному оператору select, що в свою чергу безпосередньо пов'язано з потрібним результатом. Також для кращого виведення необхідних результатів використані функції.

Також наступним кроком, треба підготувати середовище програмування IntelliJ IDEA, для фреймворку Spring Boot. Треба завантажити проект з спеціального сайту spring.io, та Postman для майбутньої перевірки запитів.

2.2. Проектування структури бази даних

У даній роботі використано метод знаходження кожного запиту окремо, для створення загальної картини про задану базу даних.

Для організації зв'язку між таблицями `worker` та `department` використовується зв'язок типу «Один до одного» (One to one). Даний тип зв'язків зустрічає не часто, але в даному прикладі ми задаємо головного робітника до відділу. У цьому випадку об'єкту однієї сутності можна порівняти тільки один об'єкт іншої сутності. Нерідко цей тип зв'язків передбачає розбиття однієї великої таблиці на декілька маленьких. Основна батьківська таблиця в цьому випадку продовжує утримувати часто використовувані дані (`worker`), а дочірня залежна таблиця (`department`) зазвичай зберігає дані, які використовуються рідше. В цьому відношенні первинний ключ залежною таблицею в той же час є зовнішнім ключем, який посиляється на первинний ключ з головної таблиці.

Таблиці `department` – `worker`, `worker` – `work_position`, `equipment` – `equipment_category`, `client` – `user_type` таблиці пов'язані типом зв'язку «Один до багатьох» (One to many). Цей тип зв'язків зустрічається найбільш часто. У цьому типі зв'язків кілька рядків з дочірній таблиці залежать від одного рядка в батьківській таблиці. Наприклад, один робочий знаходиться на одній посаді, і на одній посаді може знаходитися безліч робочих. В цьому випадку таблиця посад є батьківською, а таблиця робочих - дочірньою. Тобто один посада - багато робочих.

Зв'язок «Багато до багатьох» (Many to many)

При цьому типі зв'язків один рядок з таблиці А може бути пов'язана з безліччю рядків з таблиці В. В свою чергу один рядок з таблиці В може бути пов'язана з безліччю рядків з таблиці А. Типовий приклад – один контракт може мати безліч виконуваних проектів, та один проект може виконуватися багатьма контрактами. Але в SQL Server на рівні бази даних ми не можемо встановити прямий зв'язок багато до багатьох між двома таблицями. Це робиться за

допомогою допоміжної проміжної таблиці. В нашому випадку це таблиці contract_project, subcontracting_do, equipment_contract_project, project_staff.

Принцип роботи програми зображено на блок-схемі, представленій на рисунку 2.1.

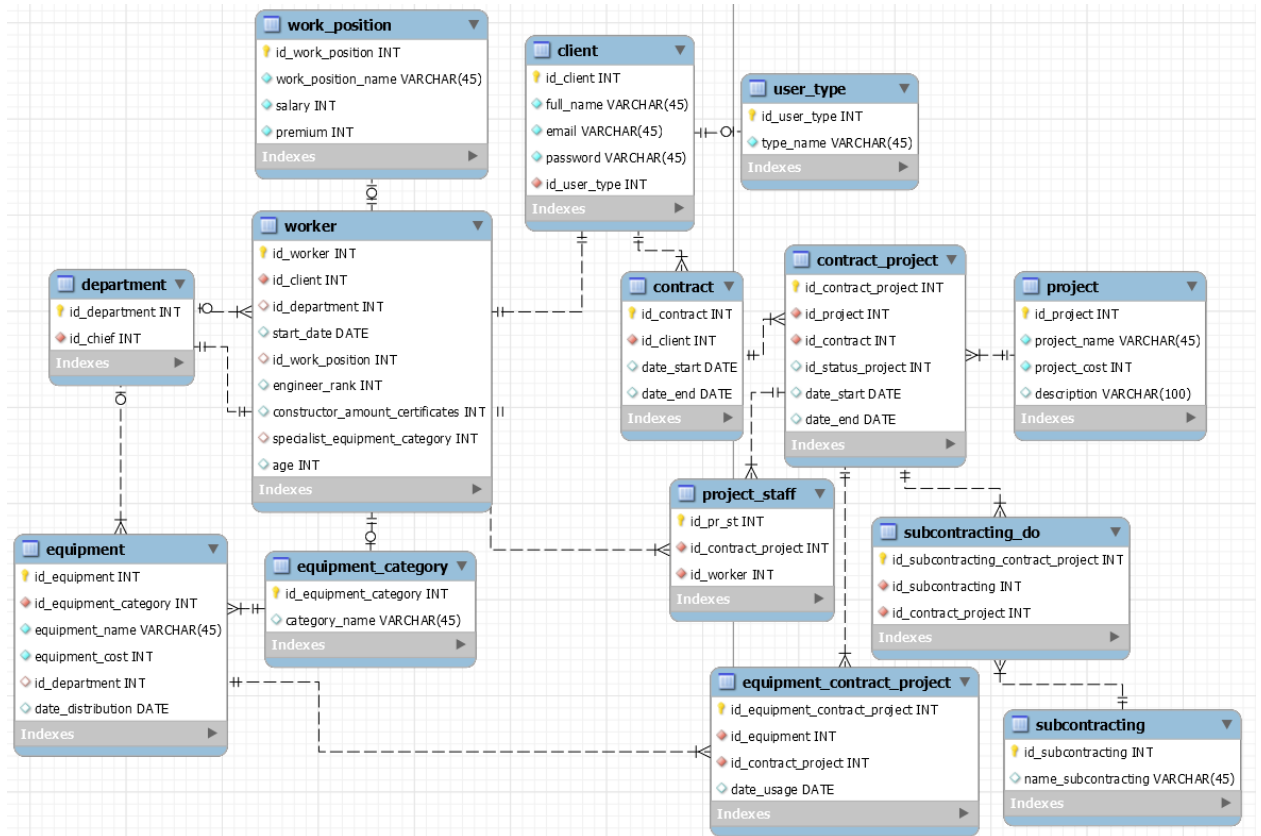


Рис. 2.1. Структурна блок-схема БД

2.3 Життєві цикли бази даних

1. Попереднє планування

Попереднє планування включало в себе письмове й усне планування та аналіз готової бази даних, формування необхідних моделей та їх важливих складових. Усі моделі мали відповідати потрібним вимогам, вказаним в завданні роботи. Реалізація відбувалась відповідно до планування. Завчасне проектування веб-інтерфейсу.

2. Перевірка здійсненності

Перевірка здійсненності реалізовувалась за допомогою наявності доступу до бази даних програми MySQL. Всі функції працюють правильно. Середовище програмування та фреймворк на якому здійснюється робота працює справно.

3. Визначення вимог

Вимоги реалізовані відповідно до листа-завдання. Задана БД детально зображує та демонструє усі потрібні елементи. Веб інтерфейс зрозумілий та має відображає всі можливості рівнів доступу.

4. Реалізація

Під реалізацією розуміється написання програмного коду SQL, виконання заданих запитів та коректна реалізація роботи всієї бази даних. Також виконання деяких базових CRUD запитів та виведення в якості ВЕБ інтерфейсу.

Під час роботи з програмою я вдосконалив навички володіння мовою структурованих запитів, а також програмне середовище MySQL, дослідив та вивчив усі необхідні функції та стандартні команди мови. Навчився створювати базові запити для витягу на ВЕБ, а саме: створення, читання, модифікація, видалення.

3. Реалізація програмної взаємодії з БД

3.1 Інструкція користувача

Оскільки сам проект має веб-інтерфейс користувачу потрібен тільки доступ до інтернету і будь-який браузер. Так як ми використовували набір вже готових шаблонів Bootstrap, то сайт оптимізований під будь-який з браузерів, отже у нього не буде проблем з візуалізацією.

Даний інтерфейс був логічно розбитий на 4 частини: режим Unauthorized для людей які не авторизовані, режим Client для зареєстрованих клієнтів, режим Worker для робітників організації, Manager для керівників організації.

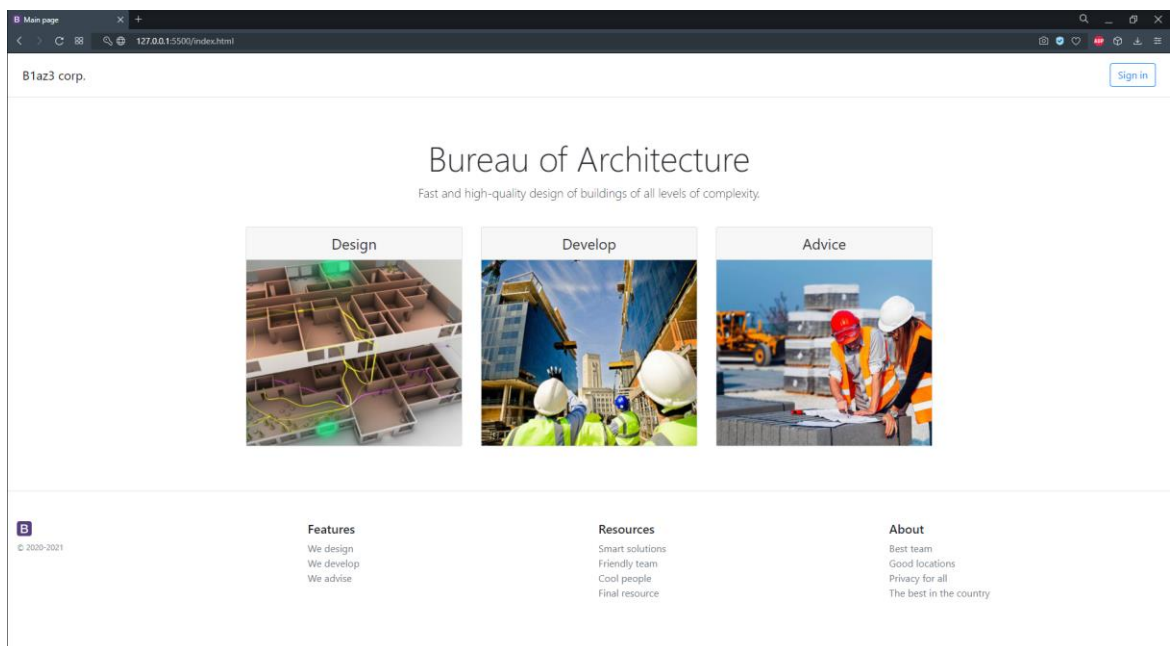


Рис. 3.1 Візуальна частина головної сторінки неавторизованого користувача.

Як тільки користувач заходить на сайт, його зустрічає головна сторінка сайту (Рис.3.1), де відображується головна інформація про організацію. На цій сторінці можна натиснути на кнопку авторизації, та вона перемістить користувача до відповідного вікна (Рис. 3.2).

У цьому вікні є два поля для вводу даних, а саме поле імейлу та пароллю, тобто це дає можливість увійти користувачеві до свого кабінету. Якщо у користувача немає особистого кабінету, то він натиснув на поле “Create an account” відкриється вікно де користувач може створити його.

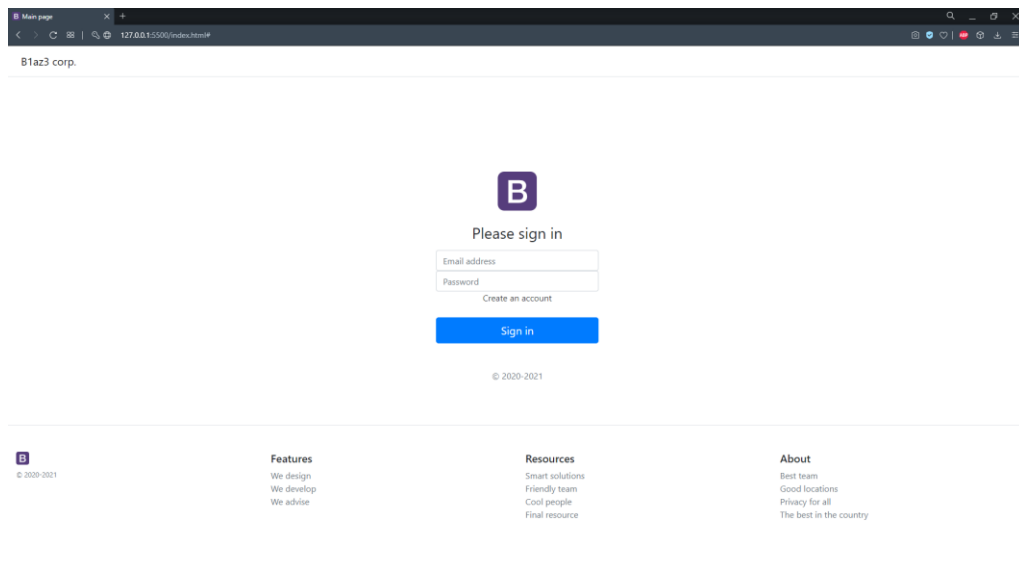


Рис. 3.2 Візуальна частина вікна авторизації користувача.

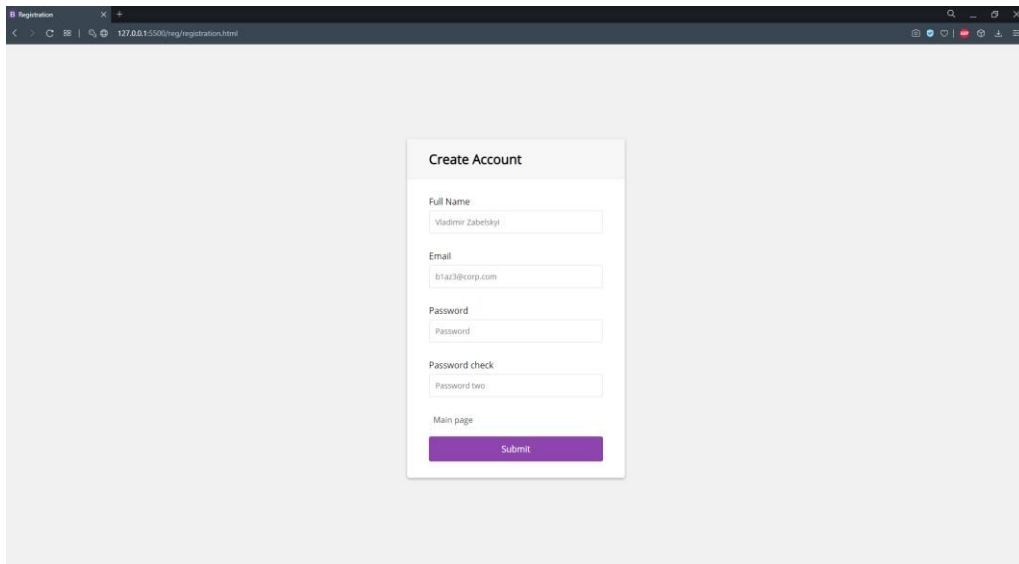


Рис. 3.3 Візуальна частина вікна реєстрації користувача.

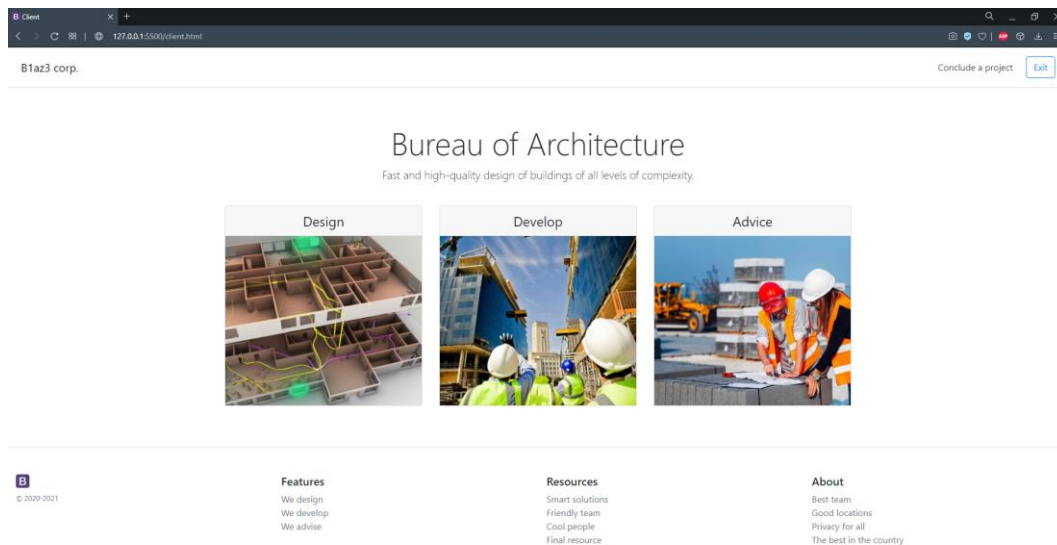


Рис. 3.4 Візуальна частина клієнтського вікна.

На цій сторінці створення особистого кабінету, користувач може ввести всі необхідні дані, а саме: ПІБ, імейл, пароль та перевірку паролю. Якщо користувач запише не вірно якесь поле, то форма створення аккаунту йому про це повідомить. Як тільки все вірно буде вказано та користувач натисне кнопку відправки, або користувач захоче вийти, та натисне кнопку “Main page” (Рис 3.3) то він перейде назад до головної сторінки, де і виконає авторизацію.

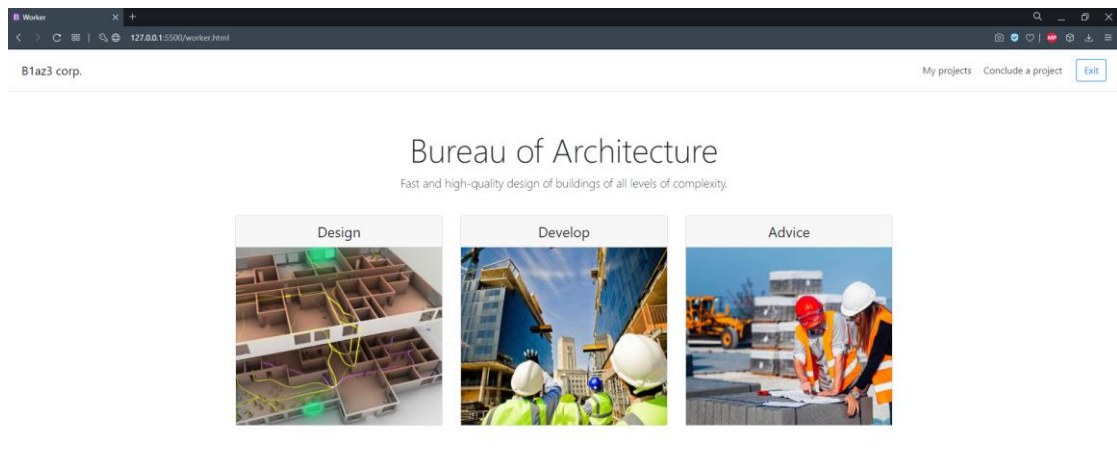


Рис. 3.5 Візуальна частина сторінки працівника організації .

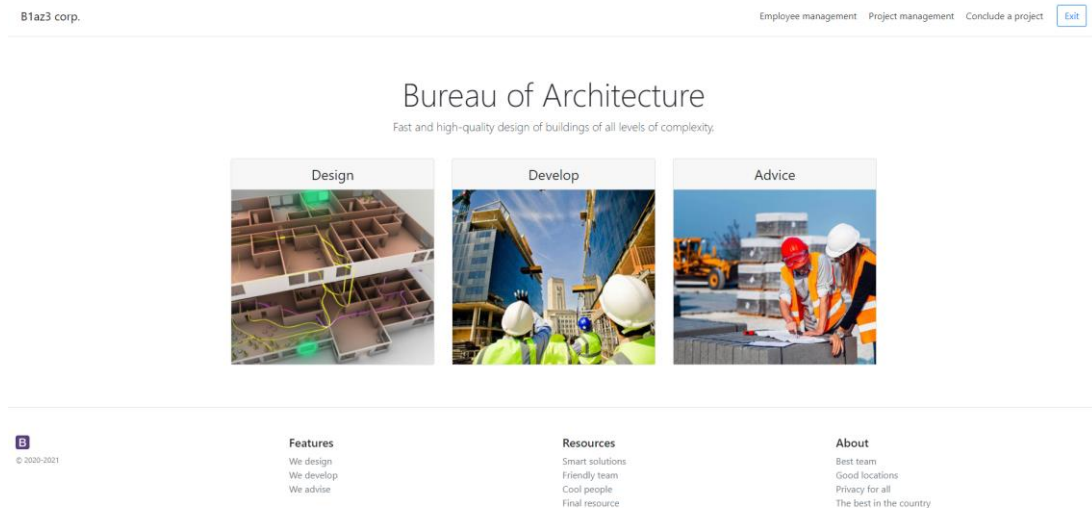


Рис. 3.6 Візуальна частина вікна менеджера.

З вікна авторизації, при вірно вказаних даних, клієнт попадає на сторінку клієнта (Рис.3.4), робочий на сторінку працівника організації (Рис. 3.5), а керівники на свою сторінку менеджера (Рис. 3.6).

Також, для демонстрації виконання всіх запитів, які залишилися, потрібно про інсталювати SQL-Server та MySQL Workbench (додаток А).

При запуску програми відкривається база даних «courseworkDB». Для того, щоб користувач мав можливість почати саму роботу з програмним забезпеченням, необхідно додати дані до конкретних таблиць. Це виконується власне в програмному середовищі MySQL. Після занесення усієї необхідної інформації потрібно зберегти зміни за допомогою функції «Apply».

Для виведення даного запиту, необхідно з'єднатися з вікном для запитів даної БД. Для цього потрібно натиснути «database», а потім «connect to database». У відкритому вікні для виконання потрібного запиту важливо скопіювати заданий код, після чого на екран буде виведено очікуваний результат.

3.2 Реалізація механізмів БД

3.2.1 SQL-запити

#1_1 Отримати дані про склад всієї організації повністю за віковим складом.

Запит 1.1

```
call pr1_1(21, 32);
```

#1_2 Отримати всієї організації повністю з вказаної категорії працівників

Запит 1.2

```
call pr1_2(2);
```

#1_3 Отримати дані про склад зазначеного відділу за віковим складом

Запит 1.3

```
call pr1_3(21, 32, 1);
```

#1_4 Отримати дані про склад зазначеного відділу з вказаної категорії працівників

Запит 1.4

```
call pr1_4(4, 1);
```

#2 Отримати перелік керівників відділів

Запит 2

```
call pr2();
```

#3_1 Отримати перелік договорів які виконуються в даний момент

Запит 3.1

```
call pr3_1();
```

#3_2 Отримати перелік договорів які виконуються в період зазначеного інтервалу часу

Запит 3.2

```
call pr3_2 ('2021-03-10');
```

#3_3 Отримати перелік проектів які виконуються в даний момент

Запит 3.3

```
call pr3_3();
```

#3_4 Отримати перелік проектів які виконуються в період зазначеного інтервалу часу

Запит 3.4

```
call pr3_4 ('2021-03-10');
```

#4_1 Отримати інформацію про те, які проекти виконуються (виконувалися) в рамках зазначеного договору

Запит 4.1

```
call pr4_1(2);
```

#4_2 Отримати інформацію про контракти, які підтримуються зазначеним договором

Запит 4.2

```
call pr4_2(2);
```

#5 Отримати дані про вартість виконаних проектів протягом зазначеного періоду часу.

Запит 5

```
call pr5_1 ('2021-03-10');
```

#6_1 Отримати дані про розподіл обладнання на даний момент

Запит 6.1

```
call pr6_1();
```

#6_1 Отримати дані про розподіл обладнання на деяку зазначену дату.

Запит 6.1

```
call pr6_2 ('2021-03-10');
```

#7_1 Відомості про використання обладнання зазначеними проектами.

Запит 7.1

```
call pr7_1(1);
```

#8_1 Відомості про участь вказаного співробітника в проектах за певний період часу.

Запит 8.1

```
call pr8_1(1, '2021-01-15', '2021-03-15');
```

#8_2 Відомості про участь категорії співробітників в проектах за певний період часу.

Запит 8.2

```
call pr8_2(1, '2021-01-15', '2021-03-15');
```

#9 Отримати перелік і вартість робіт, виконаних субпідрядними організаціями.

Запит 9

```
call pr9();
```

#10_1 Отримати дані про склад співробітників в цілому які беруть участь в даному проекті.

Запит 10.1

```
call pr10_1(1);
```

#10_2 Отримати дані про чисельність співробітників які беруть участь в даному проекті.

Запит 10.2

```
call pr10_2(1);
```

#10_3 Отримати дані про склад співробітників за окремими категоріями, які беруть участь в даному проекті.

Запит 10.3

```
call pr10_3(1, 1);
```

#11 Отримати дані про ефективність використання обладнання (обсяги проектних робіт, виконаних з використанням того чи іншого обладнання).

Запит 11 сделать подсчет для каждого еквиппмента во сколько он юзался

```
call pr11();
```

#12 Отримати відомості про ефективність договорів (вартість договорів співвіднести з урахуванням залучених людських ресурсів).

Запит 12

```
call pr12();
```

#13_1 Отримати дані про чисельність співробітників в цілому, які беруть участь в проектах за вказаний період часу.

Запит 13.1

```
call pr13_1('2021-05-18', '2021-07-18');
```

#13_2 Отримати дані про чисельність співробітників за окремими категоріями, які беруть участь в проектах за вказаний період часу.

Запит 13.2


```
call pr13_2(4, '2021-05-18', '2021-07-18');
```

#13_3 Отримати дані про склад співробітників в цілому, які беруть участь в проектах за вказаний період часу.

Запит 13.3

```
call pr13_3('2021-05-18', '2021-07-18');
```

#13_4 Отримати дані про склад співробітників за окремими категоріями, які беруть участь в проектах за вказаний період часу.

Запит 13.4

```
call pr13_4(4, '2021-05-18', '2021-07-18');
```

#14 Отримати відомості про ефективність проектів (вартість договорів співвіднести з витраченим часом)

Запит 14

```
call pr14();
```

Детальну реалізацію процедур можна знайти в лістингу ПЗ (Додаток А)

3.2.2 Процедури і функції

Таблиця 3.1 Опис головних процедур і функцій програми

№ з/п	Процедури та функції	Призначення
1	GROUP BY	Функція, яка необхідна для агрегації записів, вибраних за допомогою запиту SELECT
2	DATEDIFF()	Функція повертає інтервал часу, який пройшов між двома часовими позначками
3	CURRENT_DATE()	Функція, яка повертає поточну дату

3.2.3 Тригери

Очищує всю історію праці робітника при видаленні його з таблиці.

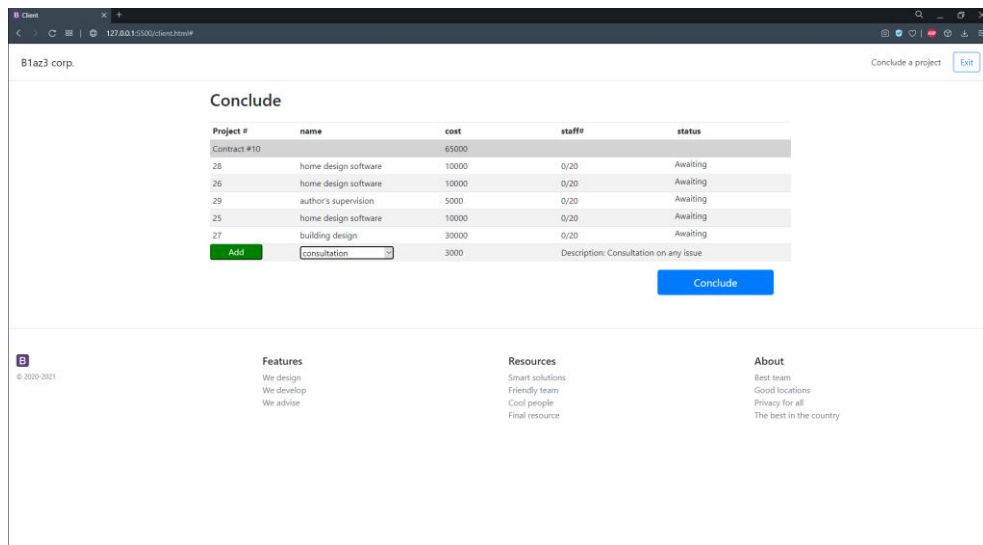
```
#DROP TRIGGER triggerDelWorkerStaff;  
DELIMITER //  
CREATE TRIGGER triggerDelWorkerStaff BEFORE delete ON worker  
FOR EACH ROW  
BEGIN  
DELETE FROM project_staff WHERE id_worker=old.worker;  
END//
```

3.3 Вимоги до апаратних і програмних засобів

У даній роботі реалізований сайт, тобто веб інтерфейс, що робить взаємодії з софтом максимально простим і не вимогливим, єдиними вимогами програми є доступ до інтернету і браузеру. Але, для початку роботи з MySQL, користувачу необхідно встановити Workbench та MySQL, також запустити сервер IDEA та через веб звертатися до запитів. Більшість запитів реалізовані в MySQL, та основні в SpringBoot. Після запуску сайту, ви можете створити новий аккаунт або увійти в уже існуючий.

3.4 Випробування розроблених програм

У таблиці “Conclude a project” (Рис. 3.7) виводяться контракти які належать тій людині яка зареєстрована, і разом с тим виводяться все проекти до тих контрактів.



Project #	name	cost	staff#	status
Contract #10		65000		
28	home design software	10000	0/20	Awaiting
26	home design software	10000	0/20	Awaiting
29	author's supervision	5000	0/20	Awaiting
25	home design software	10000	0/20	Awaiting
27	building design	30000	0/20	Awaiting
	consultation	3000		Description: Consultation on any issue

Рис. 3.7 Таблиця “Conclude a project”.

Також таблиця містить такі кнопки як “Add” (додавання нового проекту до контракту), селектор вибору типу проекту до контракту та кнопка “Conclude” яка виконує функцію додавання нового контракту клієнтові (Рис. 3.8).

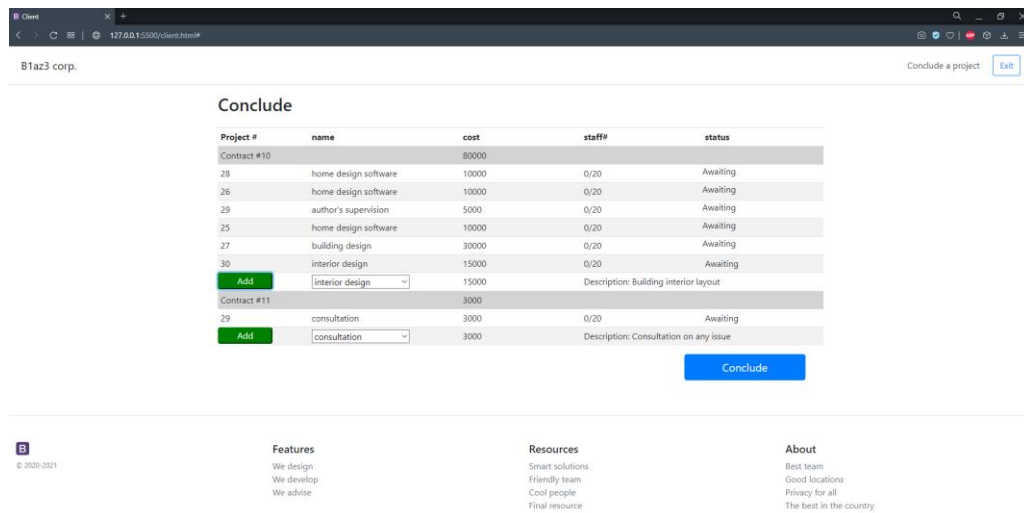


Рис. 3.8 Створення нових проектів та контрактів.

У таблиці “My projects” (Рис. 3.9) виводяться всі проекти які може взяти до себе робітник. Просто клацнувши на кнопку “Take” робітник може вступити до нового проекту, та біля проекту буде надпис “You already in project” (Рис. 3.10)

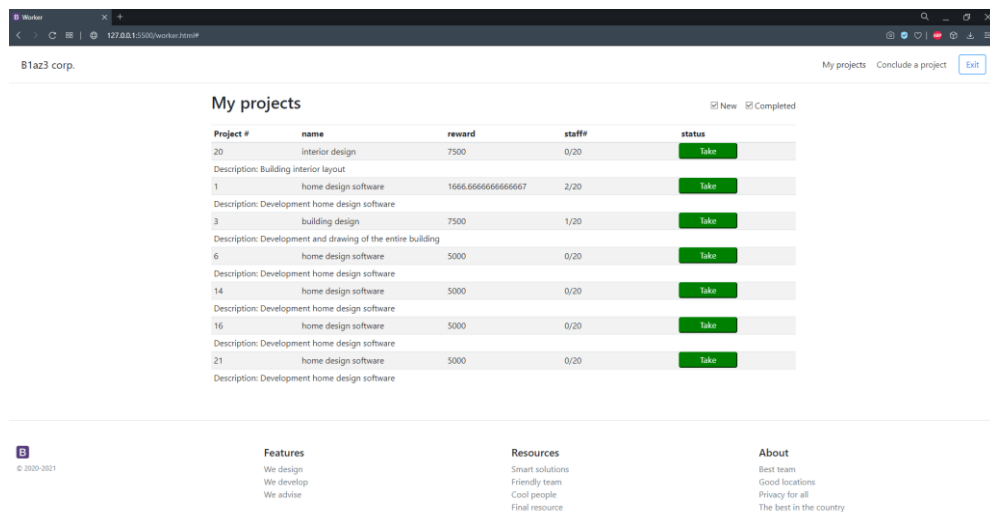


Рис. 3.9 Таблиця “My projects”.

Також при додаванні нового проекту до себе, у робітника зміниться на екрані лічильник поля “Staff” збільшиться на один. Також при натисненні на чекбокси можна відсортувати проекти, та прибрати записи нових та закінчених проектів.

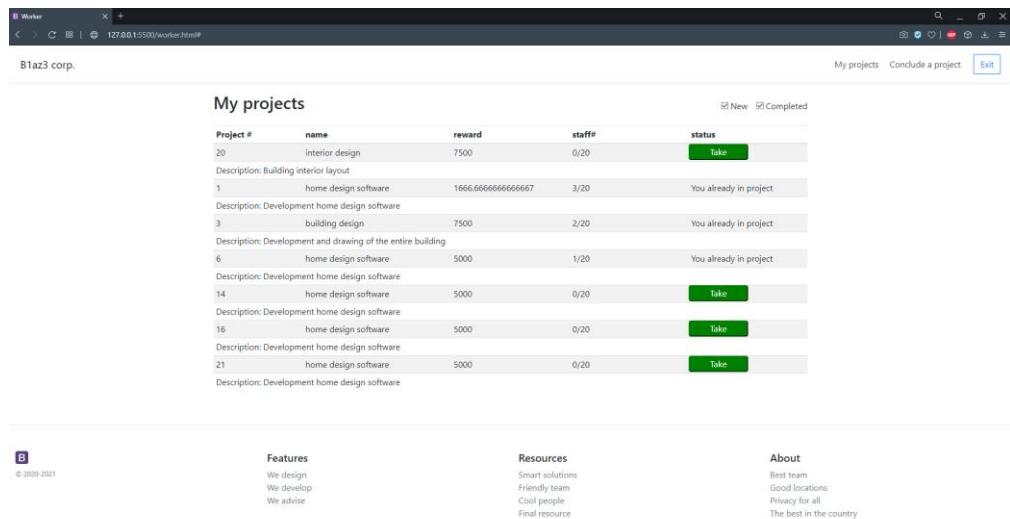


Рис. 3.10 Можливість взяти проект.

У таблиці “Еmployers” (Рис. 3.11), яка доступна тільки менеджерам, виводиться вся клієнтська база з якою можна взаємодіяти. Тобто менеджер може користувача взяти на роботу, або звільнити робітника (Рис. 3.12). Також цю таблицю можна відсортувати за допомогою чекбоксів та прибрати клієнтів та робочих (Рис. 3.13).

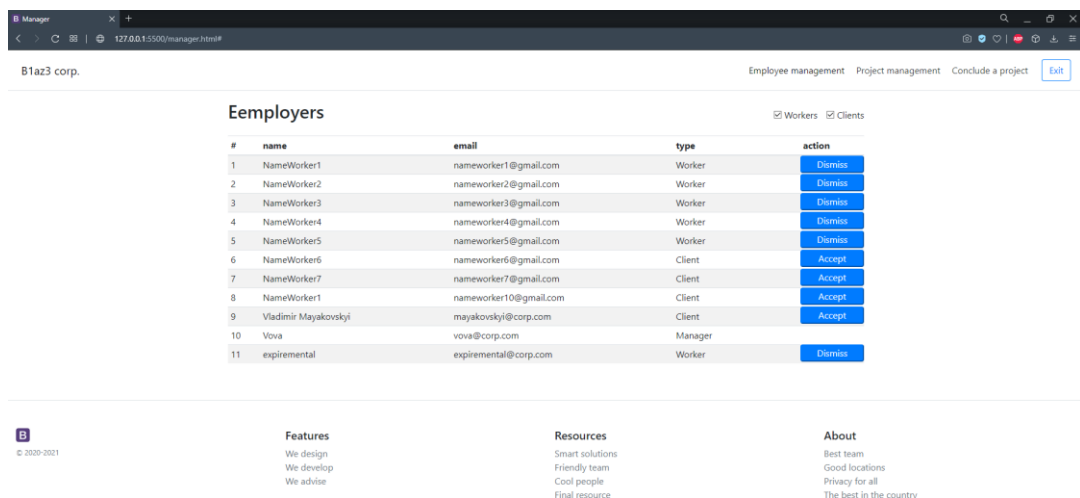


Рис. 3.11 Таблиця “Employeee managment”.

У таблиці “Contract and projects” (Рис. 3.14), яка доступна тільки менеджерам, виводиться всі контракти. При натисненні на кнопку “Projects” виведуться проекти до відповідного контракту (Рис. 3.14).

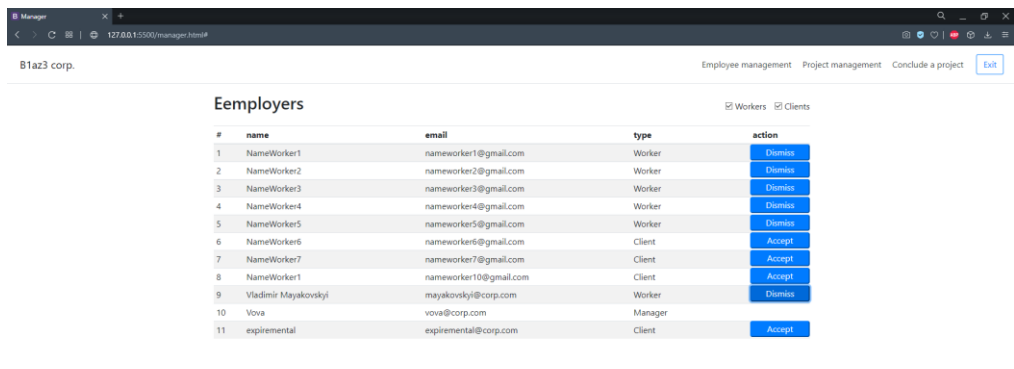


Рис. 3.12 Звільнення та прийняття робочих.

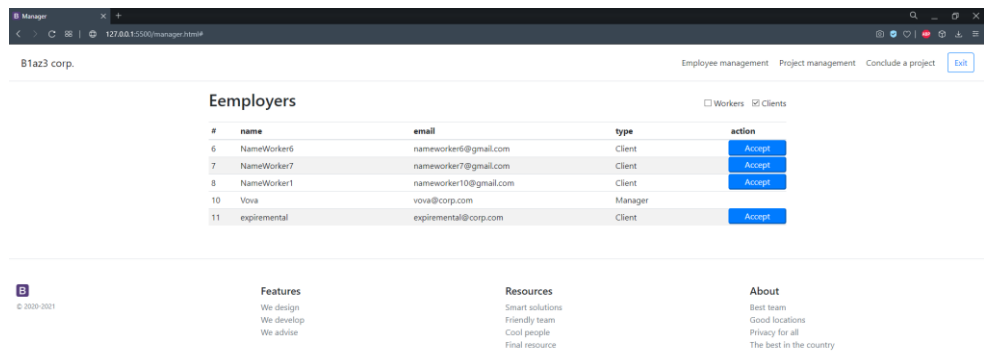


Рис. 3.13 Сортування по клієнтам.

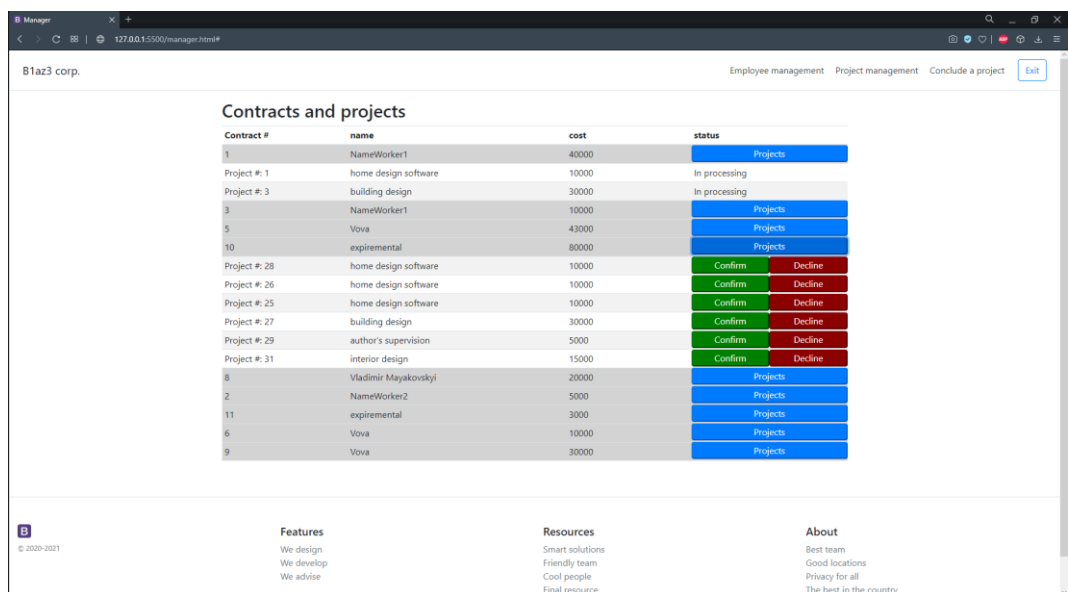
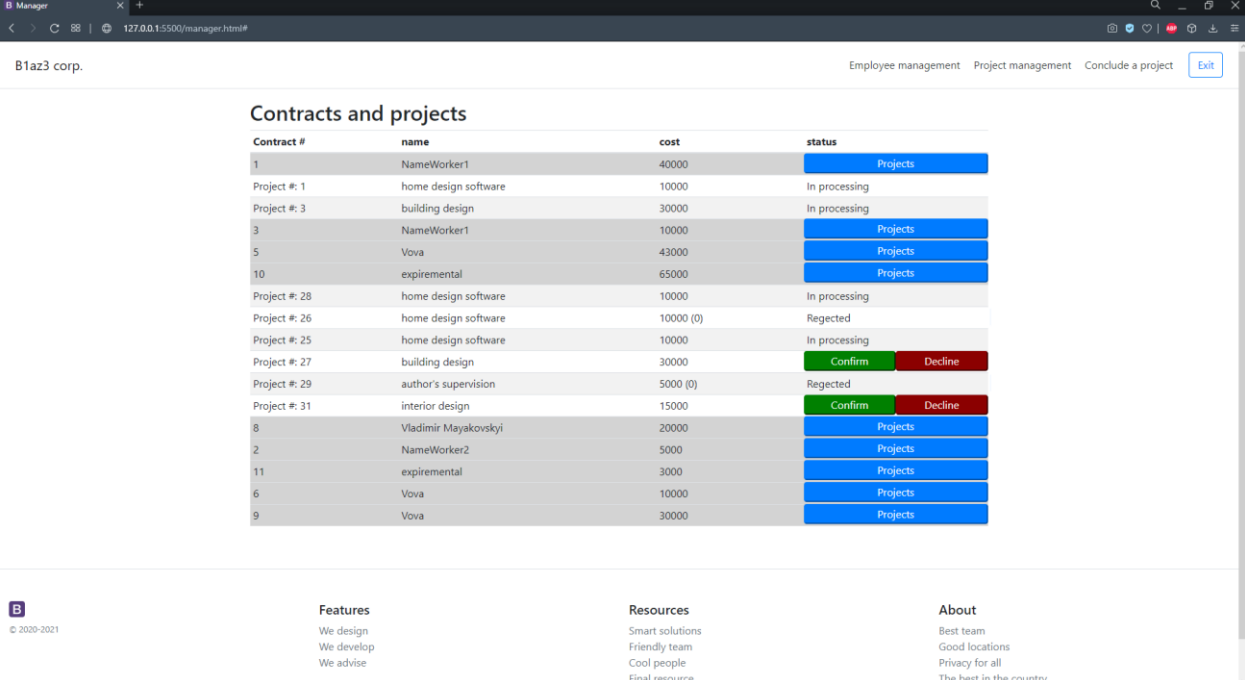


Рис. 3.14 Таблиця “Project managment”.

Ці проекти які виводяться до певного контракту мають свій статус, та якщо статус проекту на розгляді, то біля цього проекту буде дві кнопки підтвердити/відхилити. Після натиснення котрих буде мінятися статус проекту (Рис. 3.15).



B1az3 corp. Employee management Project management Conclude a project Exit

Contracts and projects

Contract #	name	cost	status
1	NameWorker1	40000	Projects
Project #: 1	home design software	10000	In processing
Project #: 3	building design	30000	In processing
3	NameWorker1	10000	Projects
5	Vova	43000	Projects
10	expiremental	65000	Projects
Project #: 28	home design software	10000	In processing
Project #: 26	home design software	10000 (0)	Rejected
Project #: 25	home design software	10000	In processing
Project #: 27	building design	30000	Confirm Decline
Project #: 29	author's supervision	5000 (0)	Rejected
Project #: 31	interior design	15000	Confirm Decline
8	Vladimir Mayakovskiy	20000	Projects
2	NameWorker2	5000	Projects
11	expiremental	3000	Projects
6	Vova	10000	Projects
9	Vova	30000	Projects

© 2020-2021 Features: We design, We develop, We advise. Resources: Smart solutions, Friendly team, Cool people, Final resource. About: Best team, Good locations, Privacy for all, The best in the country.

Рис. 3.15 Виведення проектів до контракту та можливості над ними.

Результати роботи триггеру видалення всієї історії робіт працівника (Рис. 3.17-3.18), після його звільнення (Рис. 3.16):

Рис. 3.16 Видалення робочого з таблиці “Worker”.

	id_pr_st	id_contract_project	id_worker
2	2	2	2
3	3	3	3
4	4	4	4
5	4	4	5
10	4	4	2
12	1	4	4
13	4	3	3
14	1	3	3
15	1	6	6
16	2	6	6
17	3	6	6
*	NULL	NULL	NULL

Рис. 3.17 Таблица “project_staff” до видалення робочого.

	id_pr_st	id_contract_project	id_worker
2	2	2	2
3	3	3	3
4	4	4	4
5	4	4	5
10	4	4	2
12	1	4	4
13	4	3	3
14	1	3	3
*	NULL	NULL	NULL

Рис. 3.18 Таблица “project_staff” після видалення робочого.

3.5 Опис тестової бази даних

З самого початку взаємодії з сервісом, користувач потрапляє та взаємодіє з інтерфейсом сайту, і через цей інтерфейс взаємодіє з базою даних викликаючи запити, які описані на Java фреймворці Spring Boot. Які в свою чергу вже напряму взаємодіють з БД. База даних зберігає в собі багато різних таблиць, які в залежності від вимог взаємодіють один з одним.

Для того, щоб перемістити базу даних на хостинг необхідно перемістити саму базу даних або використовувати доступ до вже існуючих загальнодоступним

ВИСНОВКИ

Під час розробки даної програми було реалізовано сайт з різними рівнями доступу та для кожної сутності реалізований інтерфейс під базу даних та в повному обсязі реалізоване технічне завдання. Дана програма дозволяє автоматизувати отримання потрібної інформації про проектну організацію. В програмі можна безпосередньо змінювати та додавати нові дані. Саме це робить її більш універсальною та доступною для користувачів.

За допомогою мови структурованих запитів вдалося реалізувати усі раніше вказані задачі, а також удосконалити пошук інформації з попередньо створених баз даних.

Основним середовищем розробки, яке використовувалося при роботі є програма MySQL Workbench.

Загальним недоліком можна вважати незначні проблеми із масштабом елементів інтерфейсу та їх доступністю.

Перевагою даного програмного продукту є можливість задавати інформацію безпосередньо під час роботи користувача. Цим досягається зрозуміла форма роботи з базою даних. В результаті можна наочно спостерігати отримання важливої інформації, що відповідає потрібним критеріям та параметрам.

У наступних версіях можна було б додати пейджінг на сайт, та реалізувати більшу кількість самих запитів, відповідно до яких можна отримати більшу кількість даних і вдосконалити зручність та ефективність роботи з БД, щоб програма стала ще універсальнішою.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Малыгина, М.П. Базы данных основы, проектирование, использование [Текст]/ М.П. Малыгина.- 2-е изд., перераб. и доп.- СПб: Петербург, 2005.- 528 с.
2. Керівництво з проектування реляційних баз даних [Електронний ресурс] – Режим доступу до ресурсу: <https://habr.com/ru/post/193380/>
3. Нормалізація відносин. Шість нормальних форм [Електронний ресурс] – Режим доступу до ресурсу: <https://habr.com/ru/post/254773/>
4. Керівництво з роботи Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: <https://bootstrap-4.ru>
5. Керівництво з роботи Spring Boot [Електронний ресурс] – Режим доступу до ресурсу: <https://spring.io/projects/spring-boot>
6. Відеоресурс з розробки web-інтерфейсу та backend на Java Spring Boot [Відео]/ І.В. Афанасьєва – Режим доступу до ресурсу: <https://drive.google.com/drive/folders/135jnBGAnPP92QkfUhJQzzCt01LzWOewb>
7. Код frontend та backend частини [Текст]/ В.В. Забельський – <https://github.com/B1az3zZz/coursework>

Додаток А Лістинг ПЗ

```
-- MySQL Script generated by MySQL Workbench
-- Fri May 21 23:34:55 2021
-- Model: New Model      Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_
O_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_EN
GINE_SUBSTITUTION';

-- -----
--
-- Schema courseworkBD
-- -----
--
-- -----
--
-- Schema courseworkBD
-- -----
--
CREATE SCHEMA IF NOT EXISTS `courseworkBD` DEFAULT
CHARACTER SET utf8 ;
USE `courseworkBD` ;

-- -----
--
-- Table `courseworkBD`.`work_position`
-- -----
--
CREATE TABLE IF NOT EXISTS
`courseworkBD`.`work_position` (
  `id_work_position` INT NOT NULL AUTO_INCREMENT,
  `work_position_name` VARCHAR(45) NOT NULL,
  `salary` INT NOT NULL,
  `premium` INT NOT NULL,
  PRIMARY KEY (`id_work_position`))
ENGINE = InnoDB;
```

```
-- -----  
--  
-- Table `courseworkBD`.`equipment_category`  
-- -----  
--
```

```
CREATE TABLE IF NOT EXISTS  
`courseworkBD`.`equipment_category` (  
  `id_equipment_category` INT NOT NULL AUTO_INCREMENT,  
  `category_name` VARCHAR(45) NULL,  
  PRIMARY KEY (`id_equipment_category`))  
ENGINE = InnoDB;
```

```
-- -----  
--  
-- Table `courseworkBD`.`user_type`  
-- -----  
--
```

```
CREATE TABLE IF NOT EXISTS `courseworkBD`.`user_type` (  
  `id_user_type` INT NOT NULL AUTO_INCREMENT,  
  `type_name` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`id_user_type`))  
ENGINE = InnoDB;
```

```
-- -----  
--  
-- Table `courseworkBD`.`client`  
-- -----  
--
```

```
CREATE TABLE IF NOT EXISTS `courseworkBD`.`client` (  
  `id_client` INT NOT NULL AUTO_INCREMENT,  
  `full_name` VARCHAR(45) NOT NULL,  
  `email` VARCHAR(45) NOT NULL,  
  `password` VARCHAR(45) NOT NULL,  
  `id_user_type` INT NOT NULL,  
  PRIMARY KEY (`id_client`),  
  INDEX `fk_clients_users_type1_idx` (`id_user_type`  
ASC) VISIBLE,  
  CONSTRAINT `fk_clients_users_type1`  
    FOREIGN KEY (`id_user_type`)  
    REFERENCES `courseworkBD`.`user_type`  
  (`id_user_type`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)
```

ENGINE = InnoDB;

```
-- -----  
--  
-- Table `courseworkBD`.`worker`  
-- -----  
--  
CREATE TABLE IF NOT EXISTS `courseworkBD`.`worker` (  
  `id_worker` INT NOT NULL AUTO_INCREMENT,  
  `id_client` INT NOT NULL,  
  `id_department` INT NULL,  
  `start_date` DATE NULL,  
  `id_work_position` INT NULL,  
  `engineer_rank` INT NULL,  
  `constructor_amount_certificates` INT NULL,  
  `specialist_equipment_category` INT NULL,  
  `age` INT NULL,  
  PRIMARY KEY (`id_worker`),  
  INDEX `fk_workers_departments1_idx` (`id_department`  
ASC) VISIBLE,  
  INDEX `fk_workers_clients1_idx` (`id_client` ASC)  
VISIBLE,  
  INDEX `fk_workers_work_positions1_idx`  
(`id_work_position` ASC) VISIBLE,  
  INDEX `fk_worker_equipment_category1_idx`  
(`specialist_equipment_category` ASC) VISIBLE,  
  CONSTRAINT `fk_workers_departments1`  
    FOREIGN KEY (`id_department`)  
    REFERENCES `courseworkBD`.`department`  
    (`id_department`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_workers_clients1`  
    FOREIGN KEY (`id_client`)  
    REFERENCES `courseworkBD`.`client` (`id_client`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_workers_work_positions1`  
    FOREIGN KEY (`id_work_position`)  
    REFERENCES `courseworkBD`.`work_position`  
    (`id_work_position`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_worker_equipment_category1`
```

```

        FOREIGN KEY (`specialist_equipment_category`)
        REFERENCES `courseworkBD`.`equipment_category`
        (`id_equipment_category`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-
-- Table `courseworkBD`.`department`
-----
-
CREATE TABLE IF NOT EXISTS `courseworkBD`.`department`
(
    `id_department` INT NOT NULL AUTO_INCREMENT,
    `id_chief` INT NOT NULL,
    PRIMARY KEY (`id_department`),
    INDEX `fk_departments_workers1_idx` (`id_chief` ASC)
    VISIBLE,
    CONSTRAINT `fk_departments_workers1`
        FOREIGN KEY (`id_chief`)
        REFERENCES `courseworkBD`.`worker` (`id_worker`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-
-- Table `courseworkBD`.`equipment`
-----
-
CREATE TABLE IF NOT EXISTS `courseworkBD`.`equipment` (
    `id_equipment` INT NOT NULL AUTO_INCREMENT,
    `id_equipment_category` INT NOT NULL,
    `equipment_name` VARCHAR(45) NOT NULL,
    `equipment_cost` INT NOT NULL,
    `id_department` INT NULL,
    `date_distribution` DATE NULL,
    PRIMARY KEY (`id_equipment`),
    INDEX `fk equipments equipment categories1_idx`
    (`id_equipment_category` ASC) VISIBLE,
    INDEX `fk equipments departments1_idx`
    (`id_department` ASC) VISIBLE,

```

```

        CONSTRAINT `fk_equipments_equipment_categories1`
        FOREIGN KEY (`id_equipment_category`)
        REFERENCES `courseworkBD`.`equipment_category`
        (`id_equipment_category`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
        CONSTRAINT `fk_equipments_departments1`
        FOREIGN KEY (`id_department`)
        REFERENCES `courseworkBD`.`department`
        (`id_department`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
--
-- Table `courseworkBD`.`project`
-- -----
--
CREATE TABLE IF NOT EXISTS `courseworkBD`.`project` (
  `id_project` INT NOT NULL AUTO_INCREMENT,
  `project_name` VARCHAR(45) NOT NULL,
  `project_cost` INT NOT NULL,
  `description` VARCHAR(100) NULL,
  PRIMARY KEY (`id_project`))
ENGINE = InnoDB;

```

```

-- -----
--
-- Table `courseworkBD`.`contract`
-- -----
--
CREATE TABLE IF NOT EXISTS `courseworkBD`.`contract` (
  `id_contract` INT NOT NULL AUTO_INCREMENT,
  `id_client` INT NOT NULL,
  `date_start` DATE NULL,
  `date_end` DATE NULL,
  PRIMARY KEY (`id_contract`),
  INDEX `fk_contracts_clients2_idx` (`id_client` ASC)
  VISIBLE,
  CONSTRAINT `fk_contracts_clients2`
    FOREIGN KEY (`id_client`)
    REFERENCES `courseworkBD`.`client` (`id_client`)

```

```

        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-
-- Table `courseworkBD`.`contract_project`
-----
-
CREATE TABLE IF NOT EXISTS
`courseworkBD`.`contract_project` (
  `id_contract_project` INT NOT NULL AUTO_INCREMENT,
  `id_project` INT NOT NULL,
  `id_contract` INT NOT NULL,
  `id_status_project` INT NULL,
  `date_start` DATE NULL,
  `date_end` DATE NULL,
  PRIMARY KEY (`id_contract_project`),
  INDEX `fk_contract_projects_contracts1_idx`
(`id_contract` ASC) VISIBLE,
  CONSTRAINT `fk_contract_projects_projects`
    FOREIGN KEY (`id_project`)
      REFERENCES `courseworkBD`.`project` (`id_project`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `fk_contract_projects_contracts1`
    FOREIGN KEY (`id_contract`)
      REFERENCES `courseworkBD`.`contract`
        (`id_contract`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-
-- Table `courseworkBD`.`status`
-----
-
CREATE TABLE IF NOT EXISTS `courseworkBD`.`status` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `scode` VARCHAR(95) NULL,
  `sname` VARCHAR(95) NULL,
  `bclosed` VARCHAR(95) NULL,

```

```
PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
-- -----
--
-- Table `courseworkBD`.`project_staff`
-- -----
--
CREATE TABLE IF NOT EXISTS
`courseworkBD`.`project_staff` (
  `id_pr_st` INT NOT NULL AUTO_INCREMENT,
  `id_contract_project` INT NOT NULL,
  `id_worker` INT NOT NULL,
  PRIMARY KEY (`id_pr_st`),
  INDEX `fk_project_staff_workers1_idx` (`id_worker`
ASC) VISIBLE,
  CONSTRAINT `fk_project_staff_contracts_projects1`
    FOREIGN KEY (`id_contract_project`)
    REFERENCES `courseworkBD`.`contract_project`
(`id_contract_project`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_project_staff_workers1`
    FOREIGN KEY (`id_worker`)
    REFERENCES `courseworkBD`.`worker` (`id_worker`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----
--
-- Table `courseworkBD`.`equipment_contract_project`
-- -----
--
CREATE TABLE IF NOT EXISTS
`courseworkBD`.`equipment_contract_project` (
  `id_equipment_contract_project` INT NOT NULL,
  `id_equipment` INT NOT NULL,
  `id_contract_project` INT NOT NULL,
  `date_usage` DATE NULL,
  INDEX
  `fk_equipment_has_contract_project_contract_project1_id
x` (`id_contract_project` ASC) VISIBLE,
```



```

INDEX
`fk_equipment_has_contract_project_equipment1_idx`
(`id_equipment` ASC) VISIBLE,
PRIMARY KEY (`id_equipment_contract_project`),
CONSTRAINT
`fk_equipment_has_contract_project_equipment1`
FOREIGN KEY (`id_equipment`)
REFERENCES `courseworkBD`.`equipment`
(`id_equipment`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT
`fk_equipment_has_contract_project_contract_project1`
FOREIGN KEY (`id_contract_project`)
REFERENCES `courseworkBD`.`contract_project`
(`id_contract_project`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
--
-- Table `courseworkBD`.`subcontracting`
-- -----
--
CREATE TABLE IF NOT EXISTS
`courseworkBD`.`subcontracting` (
  `id_subcontracting` INT NOT NULL,
  `name_subcontracting` VARCHAR(45) NULL,
  PRIMARY KEY (`id_subcontracting`))
ENGINE = InnoDB;

```

```

-- -----
--
-- Table `courseworkBD`.`subcontracting_do`
-- -----
--
CREATE TABLE IF NOT EXISTS
`courseworkBD`.`subcontracting_do` (
  `id_subcontracting_contract_project` INT NOT NULL,
  `id_subcontracting` INT NOT NULL,
  `id_contract_project` INT NOT NULL,

```

```

INDEX
`fk_subcontracting_has_contract_project_contract_project1_idx` (`id_contract_project` ASC) VISIBLE,
INDEX
`fk_subcontracting_has_contract_project_subcontracting1_idx` (`id_subcontracting` ASC) VISIBLE,
PRIMARY KEY (`id_subcontracting_contract_project`),
CONSTRAINT
`fk_subcontracting_has_contract_project_subcontracting1`
FOREIGN KEY (`id_subcontracting`)
REFERENCES `courseworkBD`.`subcontracting`
(`id_subcontracting`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT
`fk_subcontracting_has_contract_project_contract_project1`
FOREIGN KEY (`id_contract_project`)
REFERENCES `courseworkBD`.`contract_project`
(`id_contract_project`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `pr1_1`(IN
var1 int,var2 int)
BEGIN
SELECT worker.id_worker, clients.full_name from worker
join clients on worker.id_client = clients.id_client
where worker.age>var1 and worker.age<var2;
END

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `pr1_2`(IN
var1 int)
BEGIN
SELECT worker.id_worker, clients.full_name from worker
join clients on worker.id_client = clients.id_client
where worker.id_work_position=var1;
END

```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr1_3`(IN
age1 int, age2 int, department int)
BEGIN
SELECT worker.id_worker, clients.full_name from worker
join clients on worker.id_client = clients.id_client
where worker.age>age1 and worker.age<age2 and
worker.id_department=department;
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr1_4`(IN
position int, department int)
BEGIN
SELECT worker.id_worker, clients.full_name from worker
join clients on worker.id_client = clients.id_client
where worker.id_work_position=position and
worker.id_department=department;
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr2`()
BEGIN
select department.id_department, clients.full_name as
chief from department
join worker on department.id_chief=worker.id_worker
join clients on worker.id_worker=clients.id_client;
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr3_1`()
BEGIN
select contract.id_contract, contract.date_start,
contract.date_end from contract
where contract.date_start<CURRENT_DATE() and
contract.date_end>CURRENT_DATE();
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr3_2`(in
our_date date)
BEGIN
select contract.id_contract, contract.date_start,
contract.date_end from contract
where contract.date_start<our_date and
contract.date_end>our_date;
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr3_3`()
```

```
BEGIN
select contract_project.id_contract_project as project,
project.project_name from contract_project
join project on contract_project.id_project =
project.id_project
where contract_project.date_start<CURRENT_DATE() and
contract_project.date_end>CURRENT_DATE();
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr3_4`(in
our_date date)
BEGIN
select contract_project.id_contract_project as project,
project.project_name from contract_project
join project on contract_project.id_project =
project.id_project
where contract_project.date_start<our_date and
contract_project.date_end>our_date;
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr4_1`(IN
contract int)
BEGIN
select contract_project.id_contract_project,
project.project_name, contract.id_contract from
contract_project
join project on contract_project.id_project =
project.id_project
join contract on contract_project.id_contract =
contract.id_contract
where contract_project.id_contract = contract;
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr4_2`(IN
prj int)
BEGIN
select contract_project.id_contract_project,
project.project_name, contract.id_contract from
contract_project
join project on contract_project.id_project =
project.id_project
join contract on contract_project.id_contract =
contract.id_contract
where contract_project.id_project = prj;
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr5_1`(in  
our_date date)  
BEGIN  
select contract_project.id_contract_project as project,  
project.project_name, project.project_cost,  
contract_project.date_start, contract_project.date_end  
from contract_project  
join project on contract_project.id_project =  
project.id_project where  
contract_project.date_start<our_date and  
contract_project.date_end>our_date;  
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr6_1`()  
BEGIN  
select equipment.id_equipment,  
equipment.equipment_name, equipment.date_distribution  
from equipment where equipment.date_distribution =  
current_date();  
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr6_2`(in  
our_date date)  
BEGIN  
select equipment.id_equipment,  
equipment.equipment_name, equipment.date_distribution  
from equipment where equipment.date_distribution =  
our_date;  
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr7_1`(in  
cp int)  
BEGIN  
select equipment.id_equipment,  
equipment.equipment_name,  
contract_project.id_contract_project as project,  
project.project_name  
from equipment_contract_project  
join equipment on  
equipment_contract_project.id_equipment =  
equipment.id_equipment  
join contract_project on  
equipment_contract_project.id_contract_project =  
contract_project.id_contract_project
```

```
join project on contract_project.id_project =  
project.id_project  
where contract_project.id_contract_project=cp;  
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr8_1`(in  
worker int, var1 date , var2 date)  
BEGIN  
select project_staff.id_pr_st, worker.id_worker,  
clients.full_name,  
contract_project.id_contract_project,  
project.project_name, contract_project.date_start,  
contract_project.date_end  
from project_staff  
join worker on project_staff.id_worker =  
worker.id_worker  
join clients on worker.id_client = clients.id_client  
join contract_project on  
project_staff.id_contract_project=contract_project.id_c  
ontract_project  
join project on contract_project.id_project =  
project.id_project  
where worker.id_worker = worker  
and (contract_project.date_start>var1 and  
contract_project.date_start<var2) or  
(contract_project.date_end>var1 and  
contract_project.date_end<var2);  
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr8_2`(in  
pos int, var1 date , var2 date)  
BEGIN  
select project_staff.id_pr_st,  
worker.id_work_position, worker.id_worker,  
clients.full_name,  
contract_project.id_contract_project,  
project.project_name, contract_project.date_start,  
contract_project.date_end  
from project_staff  
join worker on project_staff.id_worker =  
worker.id_worker  
join clients on worker.id_client = clients.id_client  
join contract_project on  
project_staff.id_contract_project=contract_project.id_c  
ontract_project
```

```

join project on contract_project.id_project =
project.id_project
where worker.id_work_position = pos
and (contract_project.date_start>var1 and
contract_project.date_start<var2) or
(contract_project.date_end>var1 and
contract_project.date_end<var2);
END

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `pr9`()
BEGIN
select subcontracting_do.id_subcontracting,
contract_project.id_contract_project,
project.project_name, project.project_cost,
subcontracting.name_subcontracting
from subcontracting_do
join contract_project on
subcontracting_do.id_contract_project=contract_project.
id_contract_project
join project on
contract_project.id_project=project.id_project
join subcontracting on
subcontracting_do.id_subcontracting=subcontracting.id_s
ubcontracting;
END

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `pr10_1`(in
pr int)
BEGIN
select project_staff.id_pr_st, worker.id_worker,
clients.full_name, project_staff.id_contract_project,
project.project_name
from project_staff
join worker on project_staff.id_worker =
worker.id_worker
join clients on worker.id_client = clients.id_client
join contract_project on
project_staff.id_contract_project=
contract_project.id_contract_project
join project on contract_project.id_project =
project.id_project
where contract_project.id_contract_project = pr;

end

```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr10_2`(in  
pr int)  
BEGIN
```

```
select contract_project.id_contract_project,  
project.project_name, COUNT(*)  
from project_staff  
join worker on project_staff.id_worker =  
worker.id_worker  
join clients on worker.id_client = clients.id_client  
join contract_project on  
project_staff.id_contract_project=  
contract_project.id_contract_project  
join project on contract_project.id_project =  
project.id_project  
where contract_project.id_contract_project = pr;  
end
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr10_3`(in  
pr int, cat int)
```

```
BEGIN  
select project_staff.id_pr_st, worker.id_worker,  
clients.full_name, project_staff.id_contract_project,  
project.project_name  
from project_staff  
join worker on project_staff.id_worker =  
worker.id_worker  
join clients on worker.id_client = clients.id_client  
join contract_project on  
project_staff.id_contract_project=  
contract_project.id_contract_project  
join project on contract_project.id_project =  
project.id_project  
where contract_project.id_contract_project = pr and  
worker.id_work_position = cat;
```

```
end
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr13_1`(in  
st date, endt date)
```

```
BEGIN  
select contract_project.id_contract_project,  
project.project_name, contract_project.date_start,  
contract_project.date_end,  
COUNT(project_staff.id_contract_project)
```



```

from project_staff
left join worker on project_staff.id_worker =
worker.id_worker
left join clients on worker.id_client =
clients.id_client
left join contract_project on
project_staff.id_contract_project=
contract_project.id_contract_project
left join project on contract_project.id_project =
project.id_project
where (st between contract_project.date_start and
contract_project.date_end) or (endt between
contract_project.date_start and
contract_project.date_end)
or (contract_project.date_start<st and
contract_project.date_end>endt) or
(contract_project.date_start>st and
contract_project.date_end<endt)
group by project_staff.id_contract_project;
end

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `pr13_2`(in
cat int, st date, endt date)
BEGIN
select contract_project.id_contract_project,
project.project_name, contract_project.date_start,
contract_project.date_end,
COUNT(project_staff.id_contract_project)
from project_staff
left join worker on project_staff.id_worker =
worker.id_worker
left join clients on worker.id_client =
clients.id_client
left join contract_project on
project_staff.id_contract_project=
contract_project.id_contract_project
left join project on contract_project.id_project =
project.id_project
where worker.id_work_position = cat and ((st
between contract_project.date_start and
contract_project.date_end) or (endt between
contract_project.date_start and
contract_project.date_end)
or (contract_project.date_start<st and
contract_project.date_end>endt) or

```

```
(contract_project.date_start>st and  
contract_project.date_end<endt))  
group by project_staff.id_contract_project;  
end
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr13_3`(in  
st date, endt date)  
BEGIN  
select project_staff.id_pr_st, worker.id_worker,  
worker.id_work_position, clients.full_name,  
project_staff.id_contract_project,  
project.project_name, contract_project.date_start,  
contract_project.date_end  
from project_staff  
join worker on project_staff.id_worker =  
worker.id_worker  
join clients on worker.id_client = clients.id_client  
join contract_project on  
project_staff.id_contract_project=  
contract_project.id_contract_project  
join project on contract_project.id_project =  
project.id_project  
where (st between contract_project.date_start and  
contract_project.date_end) or (endt between  
contract_project.date_start and  
contract_project.date_end)  
or (contract_project.date_start<st and  
contract_project.date_end>endt) or  
(contract_project.date_start>st and  
contract_project.date_end<endt);  
  
end
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pr13_4`(in  
cat int, st date, endt date)  
BEGIN  
select project_staff.id_pr_st, worker.id_worker,  
worker.id_work_position, clients.full_name,  
project_staff.id_contract_project,  
project.project_name, contract_project.date_start,  
contract_project.date_end  
from project_staff  
left join worker on project_staff.id_worker =  
worker.id_worker
```

```

left join clients on worker.id_client =
clients.id_client
left join contract_project on
project_staff.id_contract_project=
contract_project.id_contract_project
left join project on contract_project.id_project =
project.id_project
where worker.id_work_position = cat and ((st
between contract_project.date_start and
contract_project.date_end) or (end between
contract_project.date_start and
contract_project.date_end)
or (contract_project.date_start<st and
contract_project.date_end>end) or
(contract_project.date_start>st and
contract_project.date_end<end));

```

end

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `pr14`()
BEGIN
select contract_project.id_contract_project,
project.project_name, project.project_cost,
contract_project.date_end, contract_project.date_start,
(project.project_cost/DATEDIFF(contract_project.date_en
d, contract_project.date_start)) as efficiency
from contract_project
join project on contract_project.id_project =
project.id_project;
end

```

```

CREATE DEFINER=`root`@`localhost` TRIGGER
`triggerDelWorkerStaff` BEFORE DELETE ON `worker` FOR
EACH ROW BEGIN
DELETE FROM project_staff WHERE
id_worker=old.id_worker;
END

```