

Військовий інститут телекомунікацій та інформатизації
імені Героїв Крут

Кафедра
“Комп’ютерних інформаційних технологій”

Курсовий проєкт
з дисципліни: “Web-технології та Web-дизайн”
на тему: “Київське квартирно-експлуатаційне управління МО України”

Виконав:

Курсант 292 навчальної групи

мол. сержант

Олександр САЧУК

Керівник:

Викладач кафедри № 22

пр. ЗСУ

Андрій ЗАДВОРНИЙ

Київ 2022

АНОТАЦІЯ

Курсового проєкту на тему:

“Київське квартирно-експлуатаційне управління МО України”

Курсова робота містить: 32 сторінок, 8 рисунків, 5 джерел.

Курсова робота присвячена розробці програмного модуля обліку академічної успішності курсантів кафедри з можливістю динамічного показу рейтингу особового складу. Проектування здійснюється за допомогою SpringBOOT Framework з використанням шаблонізатора thymeleaf та використанням бази даних MySQL.

У роботі проведено аналіз сучасного стану системи квартирної обліку військовослужбовців, наявних проблем в цій системі та перспективи подальшого її розвитку. Наведено особливості програмної реалізації та алгоритми роботи. Описано порядок роботи із програмою. У ході роботи розроблено особливий алгоритм обрахунку черги.

ANNOTATION

Course work

on the topic " Kyiv apartment management department"

Course work: contains 32 pages, 8 pictures, 5 sources.

The course work is devoted to the development of a software module for accounting of the academic success of the department's cadets with the possibility of dynamic display of the personnel rating. The design is carried out with the help of SpringBOOT Framework using the thymeleaf templater and using the MySQL database.

The paper analyzes the current state of the servicemen's apartment registration system, the existing problems in this system, and the prospects for its further development. Features of software implementation and work algorithms are given. The procedure for working with the program is described. During the work, a special algorithm for queue calculation was developed.

ЗМІСТ

АНОТАЦІЯ.....	2
ANNOTATION	3
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	5
ВСТУП.....	6
РОЗДІЛ 1	7
АНАЛІЗ МОЖЛИВОСТЕЙ РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ КВАРТИРНОГО ОБЛІКУ КИЇВСЬКОГО КЕУ	7
1.1 Аналіз існуючої системи квартирного обліку Київського КЕУ	7
1.2 Напрямки розвитку автоматизації облікового процесу	8
1.3 Вхідні дані	8
Висновок за розділом 1	8
РОЗДІЛ 2	9
ПРОЕКТУВАННЯ ТА ОСОБЛИВОСТІ РОЗРОБКИ WEB-ДОДАТКУ.....	9
2.1 Опис джерела даних, вимоги до програми та алгоритм її роботи.....	9
Висновок за розділом 2.....	13
РОЗДІЛ 3	14
ОСОБЛИВОСТІ ВИКОРИСТАННЯ ПРОГРАМНОГО МОДУЛЯ МОНІТОРИНГУ УСПІШНОСТІ КУРСАНТІВ	14
3.1 Порядок використання програмного додатку	14
3.2 Аналіз виконання поставленого завдання та подальші плани до розширення функціоналу програмного модуля	17
Висновок за розділом 3.....	18
ВИСНОВКИ	19
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	20
ДОДАТОК А	21
ЛІСТИНГ ПРОГРАМНОГО КОДУ	21

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – Application programming interface

UML – Unified Modeling Language

GUI – Graphical user interface

SDK – Software Development Kit

HTML - HyperText Markup Language

CSS – Cascading Style Sheets

XML – Extensible Markup Language

JVM – Java Virtual Machine

ВСТУП

Актуальність теми. Нині квартирний облік ведеться вручну з безліччю паперів, рапортів і заяв, що є досить не зручно та часозатратно. Також використовується старий програмний додаток з обмеженим функціоналом та поганою підтримкою на сучасних операційних системах

Тож, аби йти в ногу з часом, та полегшити збір статистичної інформації та даних про чергу, доцільно було б диджиталізувати, автоматизувати та оновити систему квартирнього обліку.

Мета роботи: розробити WEB-додаток квартирнього обліку військовослужбовців, з використанням бази даних MySQL.

Виходячи з мети роботи, виникають наступні завдання:

- проаналізувати сучасну систему квартирнього обліку, напрямки її розвитку та існуючі підходи до реалізації програмного модуля квартирнього обліку;
- проаналізувати вимоги до програми та особливості її реалізації, ознайомитися з джерелом даних, алгоритмом роботи додатку, представити розроблені класи;
- розробити інструкцію користувачу;

Об'єкт досліджень: квартирний облік Київського квартирно-експлуатаційного управління Міністерства оборони України.

Предмет досліджень: методи модернізації та автоматизації обліку військовослужбовців.

РОЗДІЛ 1

АНАЛІЗ МОЖЛИВОСТЕЙ РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ КВАРТИРНОГО ОБЛІКУ КИЇВСЬКОГО КЕУ

1.1 Аналіз існуючої системи квартирної обліку Київського КЕУ

Відповідно до Наказу Міністерства оборони України №380 від 31.07.2018 року «Про затвердження Інструкції з організації забезпечення військовослужбовців Збройних Сил України та членів їх сімей жилими приміщеннями» забезпечення військовослужбовців та членів їх сімей житловими приміщеннями від Міністерства оборони України здійснюється за рахунок:

- новозбудованого, вивільненого або придбаного житла;
- надання грошової компенсації за належне для отримання жиле приміщення (за згодою військовослужбовця);
- переобладнання нежилых приміщень фонду Міноборони у жилі (крім приміщень, розташованих на територіях, які використовуються за призначенням військовими частинами).

Для отримання житлового приміщення військовослужбовець повинен стати на квартирний облік в житловій комісії своєї військової частини. Житлова комісія реєструє квартирну справу і військовослужбовець стає в чергу за датою прийняття на квартирний облік.

На даний момент у відділі розквартирування використовується додаток створений у 1994 році на платформі .NET. Програма дуже застаріла та не підтримується розробником, крім того погано працює на нових операційних системах. Щодо даних то використовується база даних формату DBF, працювати з якою надзвичайно складно, а у наявних засобах роботи з цим форматом дуже обмежений функціонал, до того ж вони дорогі.

1.2 Напрямки розвитку автоматизації облікового процесу

Цифровізація стає невід'ємним елементом розвитку всіх сфер життя суспільства, в тому числі і військової справи. Реалії розвитку сучасного світу, безперечно, стали передумовами до впровадження діджиталізації у військову справу, особливо там де ведеться будь-який серйозний облік.

1.3 Вхідні дані

Стара база даних представлена базою DBase DB, що є безумовно застарілою та має специфічне кодування символів, що не презентабельно. Аби дістати дані необхідно написати скрипт який переведе їх спочатку в масив байтів, а потім вже в читаємий UTF-8 (див. рисунок 1.1).

NOM	NOM_VZ	VZ_DARK	FAM	IMA	OTCH	DATUCH	NLG
231	31	0/0.	0000000	00I0	00000000	19951227	7
0 0000000...							
00000000...	1	20071126	370	231			226
49	31	0/0.	00I0000	0000000	0000000I0...	19941026	0
0000 I0 1993	1	20071123	346	49			
1312	49	0/0.000.	0000000	00I0	0000000...	20090129	7
50	31	0/0.	0000000I0...	00000000...	00000000...	19941026	0
0000 I0 1989							
0 0000000...	1	20071126	413	50			
674	45	0/0.	00000000	00000I0	00000000...	20020430	7
5	45	0/0.000.	00000000...	0000000	00000000...	19930519	7
0000 0000...	1	20071114	185	5			109
37	31	0/0.	0000000	00000000...	00000000...	19940831	0
0000 0I000...	1	20071123	365	37			
9	31	0/0.000.	0000000	I000	00000000...	19930917	7
0 0000000...	1	20071130	768	9			232
711	46	0/0.	0I0000	00000000...	00000I000...	20021113	7
0 0I0 1999	1	20071115	257	709			48
12	31	0/0.	0000000	00I0	0I000000...	19940105	0
0 0 1986							
0 00000 1...	1	20071123	281	12			
22	35	0/0.000.	00000000...	00000000...	00000000...	19940705	7
0 0000000...							
0000 000 1...	1	20071115	261	22			4

Рисунок 1.1 – Вхідні дані DBase

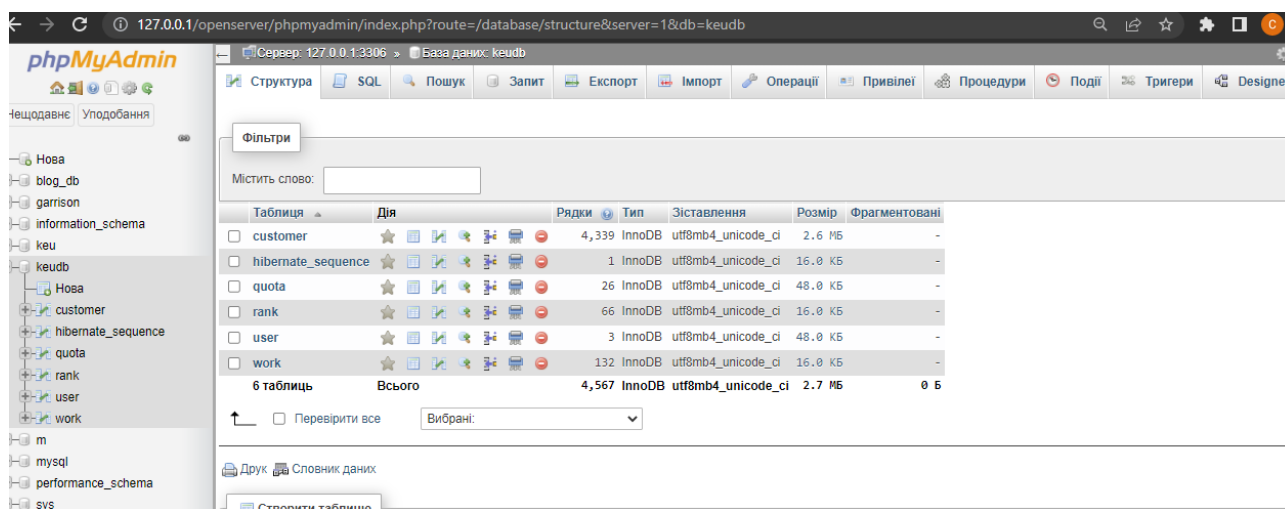
Висновок за розділом 1

Отже, зважаючи на стрімкий розвиток цифровізації в усіх сферах життя, та його підтримку відповідними розпорядженнями з боку влади, а також на потребу полегшення роботи працівників відділу розквартирування, було б доцільно інтегрувати і Збройні Сили України в інформаційно-технологічний простір.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА ОСОБЛИВОСТІ РОЗРОБКИ WEB-ДОДАТКУ

2.1 Опис джерела даних, вимоги до програми та алгоритм її роботи



Таблиця	Дія	Рядки	Тип	Зіставлення	Розмір	Фрагментовані
<input type="checkbox"/> customer	☆	4,339	InnoDB	utf8mb4_unicode_ci	2.6 МБ	-
<input type="checkbox"/> hibernate_sequence	☆	1	InnoDB	utf8mb4_unicode_ci	16.0 КБ	-
<input type="checkbox"/> quota	☆	26	InnoDB	utf8mb4_unicode_ci	48.0 КБ	-
<input type="checkbox"/> rank	☆	66	InnoDB	utf8mb4_unicode_ci	16.0 КБ	-
<input type="checkbox"/> user	☆	3	InnoDB	utf8mb4_unicode_ci	48.0 КБ	-
<input type="checkbox"/> work	☆	132	InnoDB	utf8mb4_unicode_ci	16.0 КБ	-
6 таблиць		Всього			2.7 МБ	0 Б

Рисунок 2.1 – База даних MySQL

Для додатку необхідна сучасна база даних, тому було прийнято рішення використовувати СУБД MySQL. Для цього необхідно імпортувати наявні дані з старого формату. Щоб це зробити знадобиться конвертор форматів і скрипт для конвертування тексту в масив байтів, а з нього читаємий SQL dump.

Розроблений додаток повинен містити сторінку логіну, головну сторінку зі статистикою та загальною інформацією. Серед функціоналу слід виділити можливість додавання особи, тобто її запис у чергу на забезпечення житлом, видаленням та редагуванням її даних. Також необхідно видавати довідку з інформацією про місце людини в черзі. Серед іншого слід виокремити формування самої черги за різноманітними параметрами: житлова комісія, місце служби, наявні пільги, кількість членів сім'ї і тд, також додати можливість друку такої черги за визначеним шаблоном. Доцільно буде додати пошук військовослужбовців.

Для того аби краще зрозуміти алгоритм роботи модуля, наведено UML-діаграму послідовності (див. рисунок 2.2).

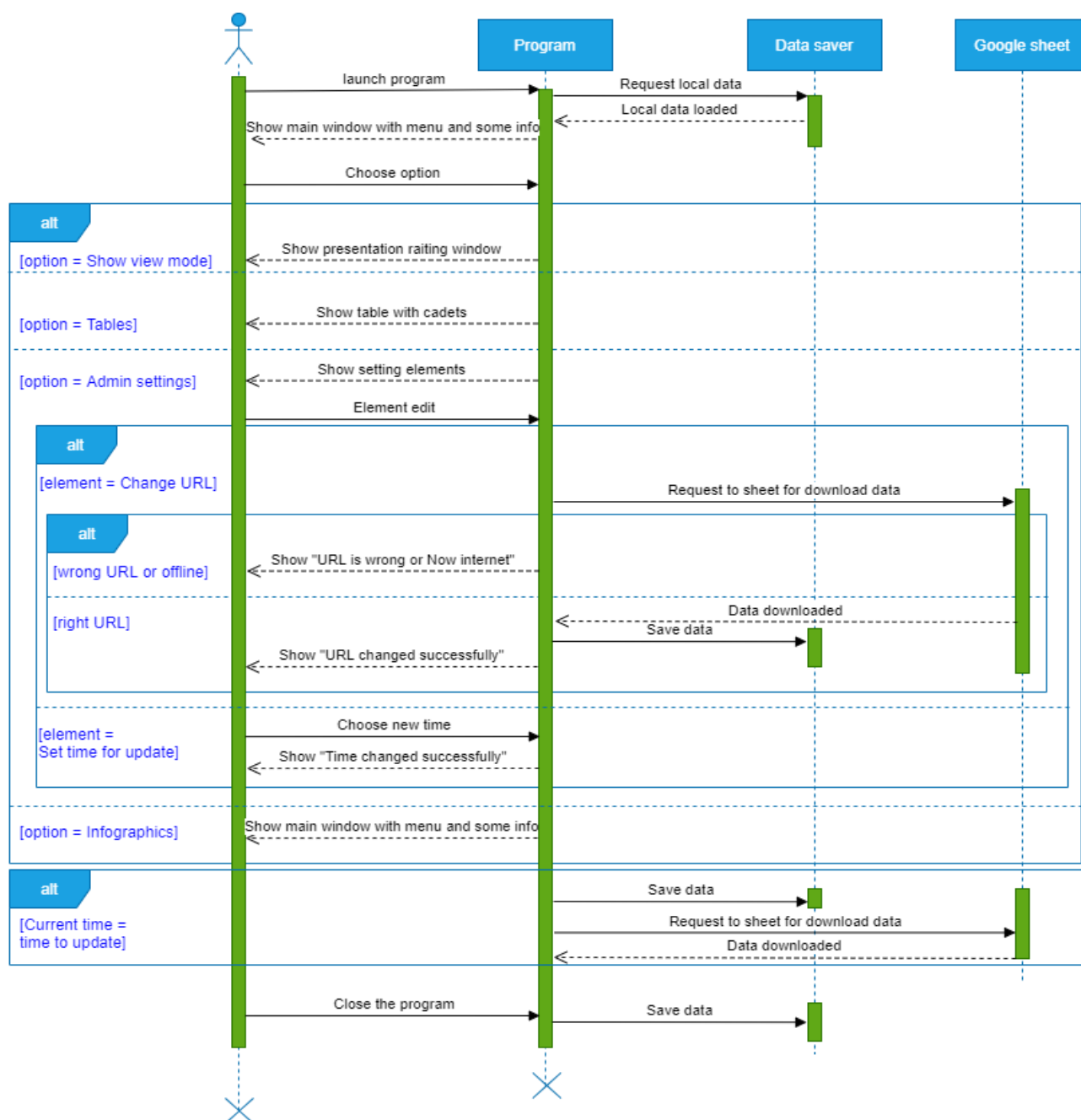


Рисунок 2.2 – Діаграма послідовності

2.3 Особливості програмної реалізації

Аби створити зрозумілий і зручний у використанні додаток необхідно реалізувати графічний інтерфейс користувача (Graphical user interface). GUI – це різновид користувацького інтерфейсу, в якому всі елементи (кнопки, меню, піктограми, списки) виконані в виді картинок, графіки та представлені користувачу на дисплеї. На відміну від інтерфейсу командної строки, в GUI у користувача є довільний доступ до видимих об'єктів за допомогою пристроїв введення. Найчастіше елементи інтерфейсу реалізовані у вигляді метафор і відображають їх властивості і призначення для полегшення сприймання користувачем.

Початковою GUI системою в Java є Abstract Window Toolkit, яка визначає базовий набір елементів керування, вікон та діалогів, які підтримують придатний до використання, але обмежений у можливостях графічний інтерфейс. Однією з причин обмеженості AWT є те, що зовнішній вигляд компонентів визначається платформою, а не закладається в Java. Оскільки компоненти AWT використовують “рідні” ресурси коду, вони називаються ваговитими. Використання “рідних” рівноправних компонентів породжує деякі проблеми. По-перше, у зв'язку із різницею, що існує між операційними системами, компонент може виглядати або навіть вести себе по-різному на різноманітних платформах. Така мінливість суперечила філософії Java: написане один раз, працює скрізь». По-друге, зовнішній вигляд кожного компонента був фіксованим (оскільки усе залежало від платформи), і це неможливо було змінити (принаймні, це важко було зробити). У зв'язку з цим в AWT на різних платформах виникали різні помилки і програмісту доводилось перевіряти працездатність програм на кожній платформі окремо. Незабаром після появи початкової версії Java, стало очевидним, що обмеження, властиві AWT, були настільки незручними, що потрібно було знайти кращий підхід. У результаті з'явилися класи Swing як частина бібліотеки базових класів Java (JFC). В 1997 році вони були включені до Java 1.1 у вигляді окремої бібліотеки. А починаючи

з версії Java 1.2, класи Swing (а також усі останні, що входили до JFC) стали повністю інтегрованими у Java. Щоправда графічні класи AWT до сих пір використовується при написанні невеликих програм та аплетів. Крім того Swing хоч і надає більше можливостей роботи з графікою, проте не заміняє їх повністю.

В той же час активно велася робота над розробкою нової мови для створення графічних інтерфейсів – F3 (Froms Follows Functions). Цим займався Кріс Олівер з компанії SeeBeyond, яку згодом в 2005 році купила Sun Microsystems, яка на той час займалася розвитком Java. Тоді ж F3 перейменувалась на JavaFX, і Олівер продовжив над нею роботу вже в рамках нової компанії. І в травні 2007 Sun Microsystems публічно анонсувала нову платформу для створення графічних програм. А 4 грудня 2008 року вийшов JavaFX 1.0 SDK.

Після покупки Sun Microsystems компанією Oracle, в 2011 році вийшла в реліз версія JavaFX 2.0. В першій версії JavaFX фактично представляв собою скриптову мову. В другій версії було повністю змінено підхід. Скриптова мова була прибрана, а платформа переписалася фактично з нуля. Тепер створювати програми можна було за допомогою будь-якої мови, яка підтримувала JVM. Були добавлені нові API, інтеграція зі Swing і багато іншого.

Наступними важливими етапами розвитку платформи стали версії JavaFX 8 і особливо JavaFX 9, яка вийшла в вересні 2017 року і додала до платформи модульність.

На даний момент JavaFX є популярним способом створення графічних додатків за допомогою мови Java, який прийшов на заміну AWT та Swing. JavaFX надає більше можливостей порівняно з рядом інших подібних платформ, зокрема, порівняно зі Swing. Це і великий вибір елементів управління, і можливості роботи з мультимедіа, двохвимірною та трьохвимірною графікою, декларативний спосіб опису інтерфейсу за допомогою мови розмітки FXML, можливість стилізації інтерфейсу CSS, інтеграція зі Swing.

Саме через оголошені вище переваги JavaFX, її і було обрано для розробки користувацького інтерфейсу програмного модуля моніторингу академічної успішності курсантів.

Так як програма повинна взаємодіяти з Google таблицями, для цього потрібно використовувати Google Sheets API. Адже тільки за допомогою цього інтерфейсу можна працювати з таблицями.

Щоб додати до проекту бібліотеки Google Sheets API, потрібно використати збірник проектів. На даний час найпопулярнішими є Gradle та Maven. Їх досить важко порівнювати, адже вони використовують різні способи збірки, і кожен в чомусь кращий, а в чомусь ні. Проте, деякі пункти все ж можна протиставити. Скрипти збірки Gradle, написані на Groovy, набагато коротші і простіші XML, використовуваних Maven. Безумовно Gradle швидше Maven збирає проект, для цього в ньому використовуються всеможливі кеши і навіть спеціальний процес – Gradle Daemon, працюючий після збірки. Якщо ж його вимкнути або робити збірку проекту не так часто, то приріст в продуктивності буде незначним. Gradle не має власної інфраструктури і завантажує всі залежності з Maven – репозиторію. Таким чином, Gradle – це просто збірник проектів.

Перечислених вище переваг достатньо аби надати перевагу у використанні Gradle.

Висновок за розділом 2

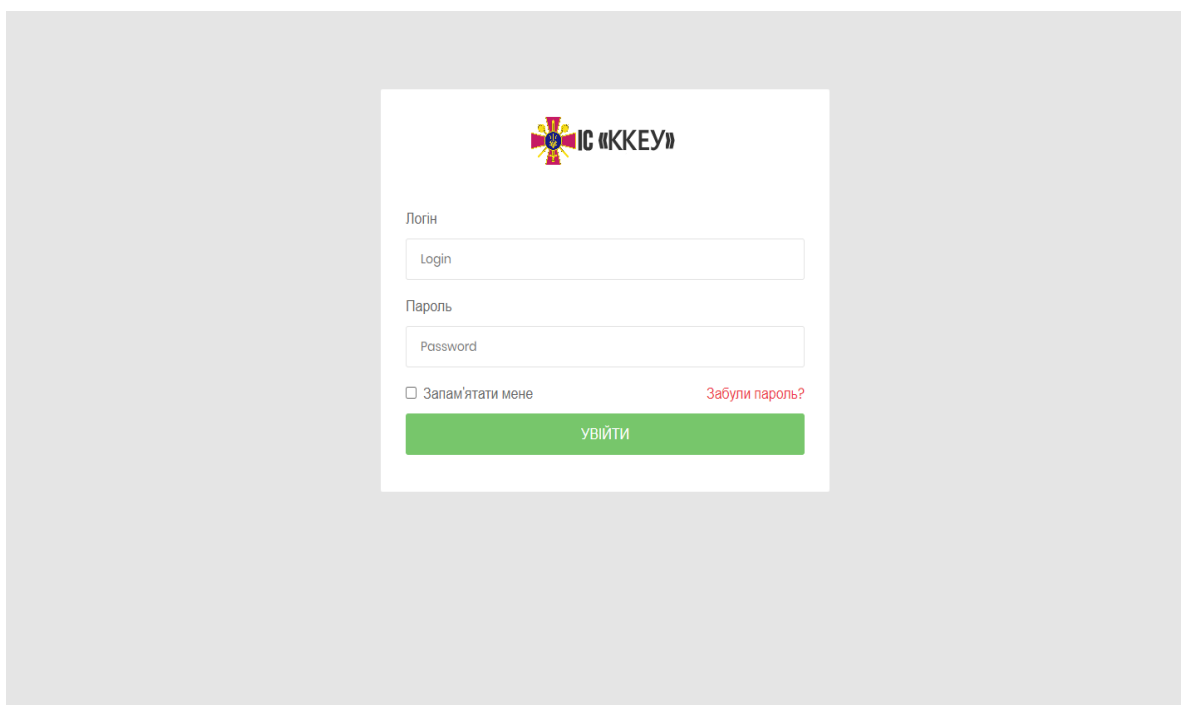
Отже, був розроблений WEB-додаток квартирної обліку Київського квартирно-експлуатаційного управління, засобами Spring Boot Framework. Функціонал повністю задовільняє потреби ККЕУ. А динамічна таблиця, наочно показує місце в черзі кожного військовослужбовця. Загалом додаток відповідає всім вимогам і відмінно працює за призначенням.

РОЗДІЛ 3

ОСОБЛИВОСТІ ВИКОРИСТАННЯ ПРОГРАМНОГО МОДУЛЯ МОНІТОРИНГУ УСПІШНОСТІ КУРСАНТІВ

3.1 Порядок використання програмного додатку

Додаток функціонує в локальній мережі ККЕУ. Щоб розпочати роботу, оператору досить перейти за посиланням у своєму браузері. Перед ним з'явиться вікно логіну, в яке необхідно внести дані зазделегідь збереженого в базі даних користувача (див. рисунок 3.1).



The image shows a login interface for the KKEU monitoring system. At the top center is a logo consisting of a red cross with a yellow circle in the center, followed by the text 'ККЕУ'. Below the logo are two input fields: the first is labeled 'Логін' (Login) and the second is labeled 'Пароль' (Password). Under the password field, there is a checkbox labeled 'Запам'ятати мене' (Remember me) and a red link labeled 'Забули пароль?' (Forgot password?). At the bottom of the form is a green button with the text 'УВІЙТИ' (Login).

Рисунок 3.1 – Сторінка логіну

Далі користувач-оператор потрапляє на головну сторінку додатку, де відразу кидається в очі загальна інформація про стан черги, кількість записів у ній, та статистика по гарнізонах та пільгах (див. рисунок 3.2). Відразу можна помітити бокове меню, за допомогою якого можна потрапити на потрібну сторінку.

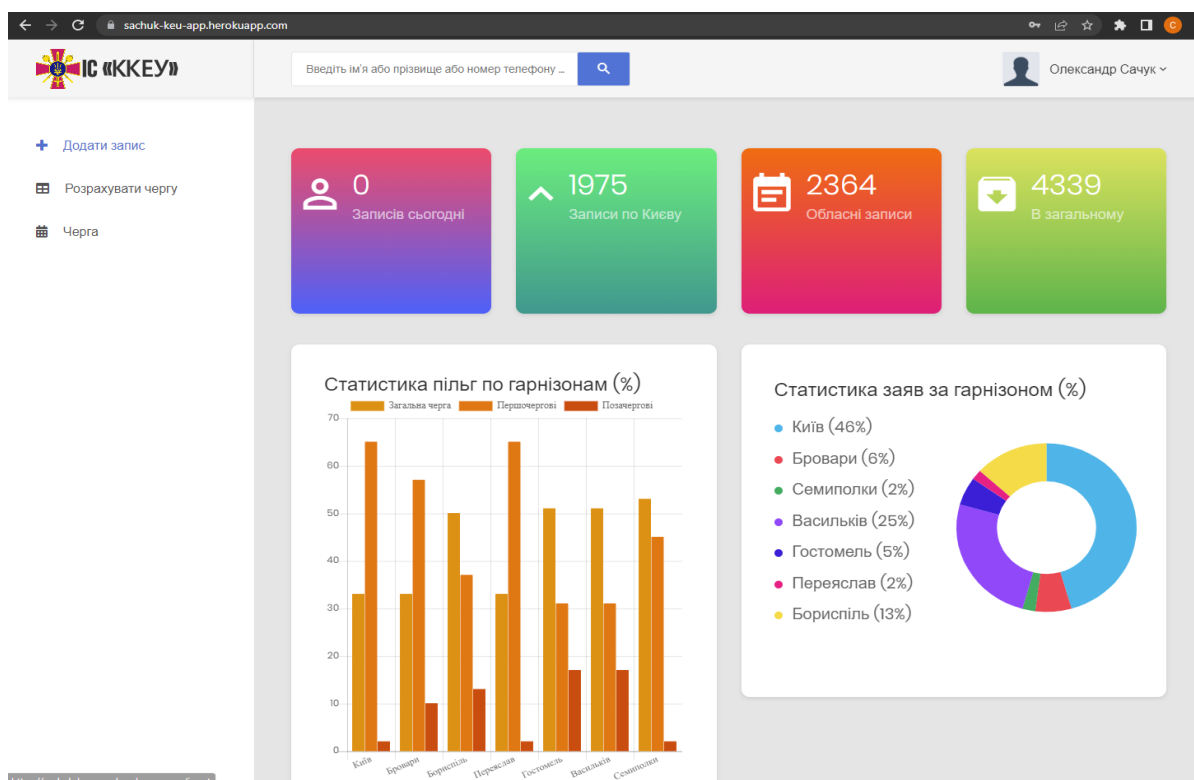


Рисунок 3.2 – Вигляд головної сторінки додатку

Перейти до запису військовослужбовця до черги можна натиснувши кнопку «Додати запис». Перед користувачем з'явиться форма для введення даних (див. рисунок 3.3).

Введіть ім'я або прізвище або номер телефону ...

Олександр Сачук

+ Додати запис

Розрахувати чергу

Черга

Основні дані про особу

Номер в загальній черзі: null

Номер в пільговій черзі: null

Прізвище

Ім'я

По батькові

Військове звання

Місце служби

Номер телефону

Дата постановки на квартирний облік

Кількість членів сім'ї

Кількість кімнат

Потребує доотримання житла?

Забезпечений?

Додаткові дані про особу

Пільга

Дата зарахування до пільгової черги

Пільга 2(за наявності)

Дата зарахування до пільгової черги

Сім'я в/сл. загиблого з часу широкомасштабного вторгнення РФ?

Пройшов реєстрацію?

В Збройних Силах з:

Дата звільнення з ЗС:

Зберегти ВІСЛАДІТИ Очищення бази Друк даних

Рисунок 3.3 – Форма введення даних

Після введення даних натиснути кнопку «Зберегти». Якщо ж потрібно редагувати дані, то знову відкриється ця сторінка, але уже із даними військовослужбовця. Тут запис можна змінити, видалити чи роздрукувати довідку.

Після натиснення кнопки меню «Черга», а в випадяючому списку вибрати потрібний гарнізон, користувач може переглянути чергу відповідного гарнізону з можливістю її відфільтрувати (див. рисунок 3.4).

Для фільтрування потрібно вибрати потрібний параметр у випадяючих списках, а потім натиснути на потрібну категорію черги: «Загальна черга», «Позачергова», «Першочергова», «Інваліди АТО/ООС».

Тут же і можна роздрукувати чергу.

Аби перерахувати чергу потрібно натиснути кнопку «Розрахувати чергу» та зачекати.

ІС «ККЕУ»

Введіть ім'я або прозвіще або номер телефону ...

🔍

Олександр Савчук

+

Додати запис

📄

Розрахувати чергу

📅

Черга

Черга гарнізону м.Київ 1932 осіб

Загальна черга

Позачергова

Першочергова

Ім/АТО/ООС

ДРУК

Організація

ЖК

Пільга

Кількість членів сім'ї

Кількість кімнат

Пройшов реєстрацію

Сортувати за висл

Всі

Всі

Всі

0

0

🗕

НІ

🗕

ТАК

🗕

Всі

🗕

НІ

🗕

ТАК

Рнд	№	Військ. звання	Прізвище, ініціали	Дата набрання	Дата пільги	Категорія пільги	К-сть чл.сім	К-сть кімн	ЖК	Організація	Висл
<div>🗕</div>	1	м-р	TestSurname7 I.M.	10.12.2014	13.02.2015	першочерговий	3	2	ЖК А0515 і ПНЧ	А0515	12
<div>🗕</div>	2	м-р	TestSurname5 С.В.	10.12.2014	21.10.2015	першочерговий	3	2	ЖК А0515 і ПНЧ	А0515	17
<div>🗕</div>	3	п-к	TestSurname2 В.Л.	10.12.2014	12.10.2015	першочерговий	1	1	ЖК ПШ ЗСУ	ПШ ЗСУ	30
<div>🗕</div>	4	п/л-к кост.	TestSurname3 Ю.Г.	10.12.2014	12.01.2017	першочерговий	4	3	ЖК КО6 Сил	А0335	29
<div>🗕</div>	5	п/л-к	TestSurname8 О.С.	10.12.2014		без пільг	4	3	ЖК НУОУ	НУОУ	20
<div>🗕</div>	6	п-к	TestSurname4 А.О.	10.12.2014		без пільг	4	3	ЖК КБЗелену	А0106	25
<div>🗕</div>	7	к-н	TestSurname1 В.А.	10.12.2014		без пільг	4	3	ЖК КСЛейстери	А2070	12
<div>🗕</div>	8	солдат	TestSurname6 І.В.	10.12.2014		без пільг	1	1	ЖК А0515 і ПНЧ	А0515	8
<div>🗕</div>	9	м-р	TestSurname9 Ю.В.	12.12.2014	06.05.2015	першочерговий	1	1	ЖК КСВ	КСВ	18
<div>🗕</div>	10	м-р м/с	TestSurname10 Н.В.	16.12.2014	15.04.2015	першочерговий	1	1	ЖК КБЗелену	А0351	9
<div>🗕</div>	11	п/л-к з/су	TestSurname11 В.М.	18.12.2014	26.10.2021	позачерговий	1	1	ЖК КСПітрини	А2330	27
<div>🗕</div>	12	м-р м/с	TestSurname12 Д.В.	19.12.2014	22.12.2015	першочерговий	3	2	ЖК КМедСил	КМедСил	14
<div>🗕</div>	13	п-к	TestSurname13 Ю.О.	19.12.2014	30.09.2019	першочерговий	4	3	ЖК КССО	А0957 ви ННБ	26
<div>🗕</div>	14	п-к	TestSurname17 І.В.	22.12.2014	22.12.2014	першочерговий	1	1	ЖК АМОУ та БПС	ГУМР	33
<div>🗕</div>	15	п-к м/с	TestSurname22 М.Ю.	22.12.2014	22.12.2014	першочерговий	3	2	ЖК АМОУ та БПС	МОУ	37

Рисунок 3.4 – Перегляд черги












Результати Пошуку														
Ред.	Заг	Пільг	В/Зв	Тип	ПІБ	На Обліку З	Пільга	Пільга З	Тип Пільги	Склад Сім'ї	Кількість Кімнат	Місце Роботи	ЖК	Гарнізон
	3	0	капітан	в/сл.	TestSurname1 ВІКТОР АНДРІЙОВИЧ	10.12.2014	Без пільг		без пільг	4	3	A2070	ЖК КСПолітики	м.Київ
	7	268	полковник	в/сл.	TestSurname2 ВАДІМ ЛЬВОВИЧ	10.12.2014	учасник БД АТО	12.10.2016	першочерговий	1	1	ПШ ЗСУ	ЖК ПШ ЗСУ	м.Київ
	8	349	підполковник юст.	в/сл.	TestSurname3 ЮРІЙ ГРИГОРОВИЧ	10.12.2014	учасник БД АТО	12.01.2017	першочерговий	4	3	A0135	ЖК КОБ.Сип	м.Київ
	2	0	полковник	в/сл.	TestSurname4 АНДРІЙ ОЛЕКСАНДРОВИЧ	10.12.2014	Без пільг		без пільг	4	3	A0106	ЖК КВЗв'язку	м.Київ
	4	67	майор	в/сл.	TestSurname5 ЄВГЕНІЙ ВАСИЛЬОВИЧ	10.12.2014	Учасник б/д	21.10.2015	першочерговий	3	2	A0515	ЖК А0515 і ПБЧ	м.Київ
	5	0	солдат за контрактом	в/сл.	TestSurname6 ІРИНА ВАСИЛІВНА	10.12.2014	Без пільг		без пільг	1	1	A0515	ЖК А0515 і ПБЧ	м.Київ
	6	6	майор	в/сл.	TestSurname7 ІГОР МИКОЛАЙОВИЧ	10.12.2014	учасник БД АТО	13.02.2015	першочерговий	3	2	A0515	ЖК А0515 і ПБЧ	м.Київ
	1	0	підполковник	в/сл.	TestSurname8 ОЛЕКСАНДР СЕРПІЙОВИЧ	10.12.2014	Без пільг		без пільг	4	3	НУОУ	ЖК НУОУ	м.Київ
	9	199	майор	в/сл.	TestSurname9 ЮРІЙ ВОЛОДИМИРОВИЧ	12.12.2014	учасник БД АТО	06.06.2016	першочерговий	1	1	КСВ	ЖК КСВ	м.Київ
	10	18	майор м/с	в/сл.	TestSurname10 НАДІЯ ВОЛОДИМИРІВНА	16.12.2014	учасник БД АТО	15.04.2015	першочерговий	1	1	A0351	ЖК КВЗв'язку	м.Київ
	11	37	п/п-к запасу	в/сл.	TestSurname11 ВОЛОДИМИР	18.12.2014	інвалід війни	26.10.2021	позачерговий	1	1	A2330	ЖК КСВ	м.Київ

Рисунок 3.5 – Результати пошуку

Також у додатку реалізовано пошук. Для цього у відповідному полі потрібно ввести прізвище або номер телефону. З'явиться таблиця з подібними результатами (див. рисунок 3.5).

3.2 Аналіз виконання поставленого завдання та подальші плани до розширення функціоналу програмного модуля

Розроблений додаток квартирному обліку дозволяє полегшити роботу з обліку військовослужбовців. Додаток повністю відповідає вимогам та є зручним у використанні. В подальшому планується додати функцію друкування різноманітних звітів та можливість редагування житлових комісій.

Висновок за розділом 3

Отже, в розділі було наведено інструкцію та порядок використання програмного модуля, з представленням всього його функціоналу. Розглянуто ступінь виконання завдань та подальші перспективи розвитку проекту, який можна зробити більш функціональним і корисним.

ВИСНОВКИ

Проводячи аналіз системи квартирної обліку військовослужбовців стало зрозуміло, що вона потребує модернізації із залученням новітніх цифрових технологій. Саме тому виникла ідея створення додатку квартирної обліку військовослужбовців для інтеграції в інформаційно-технологічний процес та оптимізації роботи інспекторів Київського квартирно-експлуатаційного управління.

Під час розробки було вивчено особливості такого фреймворку як SpringBOOT, освоєно MySQL та збірник проектів Maven, в результаті чого розроблено WEB-додаток квартирної обліку військовослужбовців з можливістю фільтрації та друком черги.

Загалом інтерфейс користувача вийшов досить ергономічним та зручним. Програма чудово виконує свої функції. Представлено інструкцію користувача, з порядком використання. В подальшому слід продовжити роботу над проектом, адже його модернізація тільки покращить і облегшить ведення обліку військовослужбовців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Oracle. Getting started [Електронний ресурс] / Oracle. – 2013. – Режим доступу до ресурсу: https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm
2. Наказ Міністерства оборони України Про затвердження Інструкції з організації забезпечення військовослужбовців Збройних Сил України та членів їх сімей жилими приміщеннями [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://zakon.rada.gov.ua/laws/show/z1020-18#Text>.
3. Сієрра К. Вивчаємо Java / К. Сієрра, Б. Бейтс.
4. Діаграми онлайн [Електронний ресурс] – Режим доступу до ресурсу: <https://app.diagrams.net>.
5. Розпорядження Про схвалення Концепції розвитку цифрової економіки та суспільства України на 2018-2020 роки та затвердження плану заходів щодо її реалізації [Електронний ресурс] – Режим доступу до ресурсу: <https://zakon.rada.gov.ua/laws/show/67-2018-p#Text>.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМНОГО КОДУ

```

package com.sachuk.keu.controllers;

import com.sachuk.keu.database.service.CustomerService;
import com.sachuk.keu.database.service.UserService;
import com.sachuk.keu.entities.SearchQuery;
import com.sachuk.keu.entities.security.User;
import lombok.AllArgsConstructor;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import java.time.LocalDate;
import java.util.List;

@Controller
@AllArgsConstructor
public class IndexController {

    //public EnrolleeService enrolleeService;
    public CustomerService customerService;
    private final UserService databaseUserService;

    @ModelAttribute("query")
    public SearchQuery createQuery() {
        return new SearchQuery();
    }

    @ModelAttribute("uri")
    public String getUri(HttpServletRequest request){ return request.getRequestURI().substring(request.getContextPath().length()); }

    @RequestMapping(value = {"/", "/cabinet"}, method = {RequestMethod.GET})
    public String show(@RequestParam(value = "error", required = false, defaultValue = "false") boolean error,
        @RequestParam(value = "success", required = false, defaultValue = "false") boolean success,
        Model modelAndView) {
        User user = databaseUserService.getByEmail(SecurityContextHolder.getContext().getAuthentication().getName());
        System.out.println(user);
        long kyivCount = (long) customerService.findByGarrison("м.Київ").size();
        modelAndView.addAttribute("success", success);
    }

```

```

        modelAndView.addAttribute("error", error);
        modelAndView.addAttribute("user", user);
        modelAndView.addAttribute("kyivCount", kyivCount);
        modelAndView.addAttribute("todayCount", customerService.countAllAfterDate(LocalDate.now().atStartOfDay()));
        modelAndView.addAttribute("oblastCount", customerService.count() - kyivCount);
        modelAndView.addAttribute("allCount", customerService.count());
        return "cabinet";

    }

    @RequestMapping(value = {"/cabinet/delete/{id}"}, method = {RequestMethod.GET})
    public String delete(@PathVariable("id") String id,
        ModelAndView modelAndView) {
        User user = databaseUserService.getByEmail(SecurityContextHolder.getContext().getAuthentication().getName());
        if (customerService.existsById(Long.valueOf(id))) {
            customerService.deleteById(Long.valueOf(id));
            return "redirect:/cabinet?success=true";
        }
        return "redirect:/cabinet?success=false";
    }

    @RequestMapping(value = "/redirect", method = { RequestMethod.GET, RequestMethod.POST })
    public ModelAndView showMain(ModelAndView modelAndView) {

        modelAndView = new ModelAndView();
        modelAndView.setViewName("input");
        return modelAndView;
    }
}

```

```

package com.sachuk.keu.controllers;

```

```

import com.sachuk.keu.database.service.CustomerService;
import com.sachuk.keu.database.service.QuotaService;
import com.sachuk.keu.database.service.UserService;
import com.sachuk.keu.database.service.WorkService;
import com.sachuk.keu.entities.Customer;
import com.sachuk.keu.entities.Quota;
import com.sachuk.keu.entities.SearchQuery;
import com.sachuk.keu.entities.Work;
import com.sachuk.keu.entities.enums.QuotaType;
import com.sachuk.keu.entities.enums.Registrated;

```

```

import com.sachuk.keu.entities.security.User;
import com.sachuk.keu.services.rating.RatingXLSWeb;
import com.sachuk.keu.services.rating.RatingXlsCreateService;
import lombok.AllArgsConstructor;

import org.springframework.core.io.InputStreamResource;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

```

```

import javax.servlet.http.HttpServletResponse;
import javax.transaction.Transactional;
import java.awt.print.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;
import java.util.stream.Collectors;

```

```
@Controller
```

```
@AllArgsConstructor
```

```

public class RatingController {
    private UserService databaseUserService;
    private RatingXlsCreateService ratingXlsCreateService;
    public QuotaService quotaService;
    public WorkService workService;
    public CustomerService customerService;

```

```

    public static List<String> garrisons =
Arrays.asList("м.Київ","Бориспіль","Семиполки","Переяславський","Бровари","Гостомель","Васильків");

```

```
@ModelAttribute("quotas")
```

```
@Transactional
```

```

public List<Quota> getQuota(){
    return quotaService.findAll().stream()
        .sorted(Comparator.comparing(Quota::getNameQuota)).collect(Collectors.toList());
}

```

```

public Collection<String> getAllWorkPlaces(String garrison) {
    switch (garrison) {
        case "KYIV":
            garrison = "м.Київ";
            break;
        case "BORI":
            garrison = "Бориспіль";
            break;
        case "SEMI":
            garrison = "Семиполки";
            break;
        case "PERE":
            garrison = "Переяславський";
            break;
        case "BROV":
            garrison = "Бровари";
            break;
        case "GOST":
            garrison = "Гостомель";
            break;
        case "VASY":
            garrison = "Васильків";
            break;
    }
    String garr = garrison;
    return workService.findAll().stream().filter(w -> w.getGarrison().equals(garr))
        .sorted(Comparator.comparing(Work::getWorkPlace)).map(Work::getWorkPlace).distinct().collect(Collectors.toList());
}

```

```

public Collection<String> getAllAccountings(String garrison) {
    switch (garrison) {
        case "KYIV":
            garrison = "м.Київ";
            break;
        case "BORI":
            garrison = "Бориспіль";
            break;
        case "SEMI":
            garrison = "Семиполки";
            break;
        case "PERE":
            garrison = "Переяславський";
            break;
        case "BROV":
            garrison = "Бровари";
            break;
        case "GOST":

```



```

        garrison = "Гостомель";
        break;
    case "VASY":
        garrison = "Васильків";
        break;
    }
    String garr = garrison;
    return workService.findAll().stream().filter(w -> w.getGarrison().equals(garr))

.sorted(Comparator.comparing(Work::getWorkPlace)).map(Work::getAccountingPlace).distinct().collect(Collectors.toList());
}
private static List<Customer> staticCustomers = new ArrayList<>();

@PostMapping("/getRating/{garrison}")
public String getRating(@PathVariable String garrison,
                        @ModelAttribute("query") SearchQuery query,
                        @ModelAttribute("customer") Customer customer,
                        Model model) {

    User user = databaseUserService.getByEmail(SecurityContextHolder.getContext().getAuthentication().getName());
    System.out.println(customer);

    staticCustomers = RatingXLSWeb.getCustomers(garrison, customer);

    model.addAttribute("customer", customer);
    model.addAttribute("workPlaces", getAllWorkPlaces(garrison.substring(0,4)));
    model.addAttribute("accountings", getAllAccountings(garrison.substring(0,4)));
    model.addAttribute("user", user);
    model.addAttribute("sort", garrison.substring(4));
    model.addAttribute("search", false);
    model.addAttribute("garrison", garrison.substring(0,4));
    model.addAttribute("customers", staticCustomers);
    return "rating";
}

@PostMapping("/getRating/{garrison}/print")
public void printToFile(@PathVariable String garrison,
                        @ModelAttribute("query") SearchQuery query,
                        @ModelAttribute("customer") Customer customer,
                        HttpServletResponse response,
                        Model model) {

    System.out.println(customer);
    ratingXlsCreateService.createXls(System.getProperty("user.home") + File.separator + garrison + ".xls", staticCustomers,
customer);

    Path file = Paths.get(System.getProperty("user.home") + File.separator + garrison + ".xls");

```

```

response.setHeader("Content-disposition", "attachment; filename=" + garrison + ".xls");
try {
    response.getOutputStream().write(Files.readAllBytes(file));
} catch (IOException e) {
    throw new RuntimeException(e);
}
}

@GetMapping("/calculate")
public String calculateQueue(Model model) {
    List<Customer> customers;
    List<Customer> customersPersho;
    List<Customer> customersPoza;

    Customer customer = new Customer();
    for (String gar:garrisons) {
        customers = customerService.findByGarrison(gar);
        customers = customers.stream().filter(c->c.getRegistered().equals(Registered.YES)).sorted(Comparator.comparing(Customer::getAccountingDate)).collect(Collectors.toList());
        for (int i = 0; i<customers.size(); i++) {
            customers.get(i).setZagalna(String.valueOf(i+1));
        }
        customerService.saveAll(customers);
        customersPersho = new ArrayList<>(customers);

        customersPoza = customers.stream()
            .filter(c->c.getQuotaType().equals("позачерговий"))
            .sorted(Comparator.comparing(Customer::getQuotaDate)
                .thenComparing(c -> c.getAccountingDate()))
            .collect(Collectors.toList());
        for (int i = 0; i<customersPoza.size(); i++) {
            customersPoza.get(i).setPilgova(String.valueOf(i+1));
        }
        customerService.saveAll(customersPoza);

        customersPersho = customersPersho.stream()
            .filter(c->c.getQuotaType().equals("першочерговий") || (c.getQuotaType2() != null && c.getQuotaType2().equals("першочерговий")))
            .sorted(Comparator.comparing(Customer::getQuotaDate)
                .thenComparing(c -> c.getQuotaDate2()!=null ? c.getQuotaDate2() : c.getAccountingDate()))
            .collect(Collectors.toList());
        for (int i = 0; i<customersPersho.size(); i++) {
            customersPersho.get(i).setPilgova(String.valueOf(i+1));
        }
    }
}

```

```

        customerService.saveAll(customersPersho);
    }

    return "redirect:/cabinet?success=true";
}

@GetMapping("/report")
public String generateReport() {

    return "redirect:/cabinet?success=true";
}

@GetMapping("/generateAndShowRating/{garrison}")
public String generateAndShowRating(@PathVariable String garrison,
                                     @ModelAttribute("query") SearchQuery query,
                                     Model model) {
    User user = databaseUserService.getByEmail(SecurityContextHolder.getContext().getAuthentication().getName());

    Customer customer = new Customer();
    customer.setWork(new Work("Bci", "Bci", ""));
    customer.setQuota(new Quota("Bci", "Bci", QuotaType.NONE));
    customer.setRoomCount(0);
    customer.setFamilyCount(0);
    customer.setRegistrated(Registrated.YES);
    customer.setExperience("100");

    staticCustomers = RatingXLSWeb.getCustomers(garrison + "ZAGALNA", customer);
    //staticCustomers = customerService.findByGarrison(garrison);

    model.addAttribute("customer", customer);
    model.addAttribute("workPlaces", getAllWorkPlaces(garrison));
    model.addAttribute("accountings", getAllAccountings(garrison));
    model.addAttribute("user", user);
    model.addAttribute("sort", "ZAGALNA");
    model.addAttribute("garrison", garrison);
    model.addAttribute("search", false);
    model.addAttribute("customers", staticCustomers);

    return "rating";
}
}

```

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4">
<meta charset="UTF-8">
<title>Рейтинг</title>
<head>
  <base href="/">
  <link th:href="@{../css/font-face.css}" rel="stylesheet" media="all">
  <link th:href="@{../vendor/font-awesome-4.7/css/font-awesome.min.css}" rel="stylesheet" media="all">
  <link th:href="@{../vendor/font-awesome-5/css/fontawesome-all.min.css}" rel="stylesheet" media="all">
  <link th:href="@{../vendor/mdi-font/css/material-design-iconic-font.min.css}" rel="stylesheet" media="all">
  <!-- Bootstrap CSS-->
  <link th:href="@{../vendor/bootstrap-4.1/bootstrap.min.css}" rel="stylesheet" media="all">
  <!-- Vendor CSS-->
  <link th:href="@{../vendor/animsiton/animsiton.min.css}" rel="stylesheet" media="all">
  <link th:href="@{../vendor/bootstrap-progressbar/bootstrap-progressbar-3.3.4.min.css}" rel="stylesheet" media="all">
  <link th:href="@{../vendor/wow/animate.css}" rel="stylesheet" media="all">
  <link th:href="@{../vendor/css-hamburgers/hamburgers.min.css}" rel="stylesheet" media="all">
  <link th:href="@{../vendor/slick/slick.css}" rel="stylesheet" media="all">
  <link th:href="@{../vendor/select2/select2.min.css}" rel="stylesheet" media="all">
  <link th:href="@{../vendor/perfect-scrollbar/perfect-scrollbar.css}" rel="stylesheet" media="all">
  <link th:href="@{../vendor/vector-map/jqvmap.min.css}" rel="stylesheet" media="all">
  <!-- Main CSS-->
  <link th:href="@{../css/theme.css}" rel="stylesheet" media="all">

  <!-- Fontfaces CSS-->

</head>

<body id="dynamicContent">
<div th:replace="fragments/skeleton :: navbar(${user}, ${query})"></div>
<div class="main-content" >
  <form class="col-md-12" th:object="${customer}" th:action="/getRating/"+${garrison}+"ZAGALNA" method="post">
    <h2 class="title-1 text-center m-b-15 rating-title" th:text="'Чепра гарнізону '+${customers.get(0).work.garrison}+'
'+${customers.size()}+' оциб'"></h2>
    <div class="text-center m-b-15">
      <button type="submit" class="btn" th:classappend="${sort.equals('ZAGALNA')}?'btn-success':'btn-dark'" >Загальна
чепра</button>
      <button type="submit" class="btn" th:classappend="${sort.equals('POZA')}?'btn-success':'btn-dark'"
th:formation="/getRating/"+${garrison}+"POZA">Позачепрова</button>
      <button type="submit" class="btn" th:classappend="${sort.equals('PERSHO')}?'btn-success':'btn-dark'"
th:formation="/getRating/"+${garrison}+"PERSHO">Першочепрова</button>
      <button type="submit" class="btn" th:classappend="${sort.equals('ATO')}?'btn-success':'btn-dark'"
th:formation="/getRating/"+${garrison}+"ATO">ІНВ./ATO/OOC</button>

```

```

        <button type="submit" class="btn" th:classappend="btn-dark" style="float: right"
th:formaction="/getRating/"+${garrison}+${sort}+'/print'">ДРУГ</button>
<!-- th:href="/getRating/"+${garrison}+${sort}+'/print"-->
</div>
<div class="text-center m-b-15">
    <div style="display: inline-block">
        <label class=" form-control-label"><h4>Організація</h4></label>
        <select class="form-control select" style="max-width: 200px" th:field="*{ work.workPlace}">
            <option th:each="var : ${ workPlaces}"
                th:value="${ var}" th:text="${ var}" ></option>
        </select>
    </div>
    <div style="display: inline-block">
        <label class=" form-control-label"><h4>ЖК</h4></label>
        <select class="form-control select" style="max-width: 200px" th:field="*{ work.accountingPlace}">
            <option th:each="var : ${ accountings}"
                th:value="${ var}" th:text="${ var}" ></option>
        </select>
    </div>
    <div style="display: inline-block">
        <label class=" form-control-label"><h4>Пільга</h4></label>
        <select class="form-control select" style="max-width: 200px" th:field="*{ quota}">
            <option th:each="var : ${ quotas}"
                th:value="${ var.id}" th:text="${ var.getNameQuota()}" ></option>
        </select>
    </div>
    <div style="display: inline-block">
        <label class=" form-control-label"><h4>Кількість членів сім'ї</h4></label>
        <input type="number" th:field="*{ familyCount}"
            th:value="${ customer?.familyCount}" name="text-input"
            placeholder="Кількість членів сім'ї" class="form-control">
    </div>
    <div style="display: inline-block">
        <label class=" form-control-label"><h4>Кількість кімнат</h4></label>
        <input type="number" th:field="*{ roomCount}"
            th:value="${ customer?.roomCount}" name="text-input"
            placeholder="Кількість кімнат" class="form-control">
    </div>
    <div style="display: inline-block">
        <label class=" form-control-label" style="margin-bottom: 14px"><h4>Пройшов реєстрацію</h4></label>
        <div class="form-check">
            <div class="radio" style="display: inline-block; margin-left: 25px"
                th:each="ty : ${ T(com.sachuk.keu.entities.enums.Registrated).values()}">
                <input th:field="*{ registrated}" th:value="${ ty}"
                    th:text="${ ty.getName()}" type="radio"

                name="radios"

```

```

        class="form-check-input"/>
        <label th:for="{#ids.prev('registrated')}}"
        class="form-check-label"/>
    </div>
</div>
<div style="display: inline-block; margin-left: 10px">
    <label class=" form-control-label" style="margin-bottom: 14px"><h4>Сортувати за висл</h4></label>
    <div class="form-check">
        <div class="radio" style="display: inline-block; margin-right: 25px">
            <input th:field="*{experience}" th:value="{100}"
                th:text="НІ" type="radio"
                name="radios"
                class="form-check-input"/>
            <label th:for="{#ids.prev('registrated')}}"
                class="form-check-label"/>
        </div>
        <div class="radio" style="display: inline-block; margin-right: 25px">
            <input th:field="*{experience}" th:value="{200}"
                th:text="ТАК" type="radio"
                name="radios"
                class="form-check-input"/>
            <label th:for="{#ids.prev('registrated')}}"
                class="form-check-label"/>
        </div>
    </div>
</div>
<div>
<div class="table-responsive m-b-40">
    <table class="table table-borderless mx-auto">
        <thead class="thead-dark">
            <tr>
                <th>Ред</th>
                <th>№</th>
                <!-- <th>Заг. черга.</th>-->
                <th>Військ. звання</th> <!--sec:authorize="hasAnyAuthority('OPERATOR')"-->
                <th>Прізвище, ініціали</th>
                <th>Дата кв/обліку</th>
                <th>Дата пільги</th>
                <th>Категорія пільг</th>
                <th>К-сть чл.сім</th>
                <th>К-сть кімн.</th>
                <th>ЖК</th>
                <th>Організація</th>
                <th>Висл</th>
            </tr>
        </thead>

```

```

<tbody>
<tr th:each="item,iter : ${customers}" th:class="table-success">
  <td> <!--sec:authorize="hasAnyAuthority('OPERATOR')"-->
    <a class="btn btn-light" th:href="@{'edit/'+${item.getId()}}">
      <i class="fa fa-edit" aria-hidden="true"></i>
    </a>
  </td>
  <td th:utext="${iter.index+1}" class="table-danger"></td>
<!--
  <td th:utext="${customers.indexOf(item)+1}"></td-->
  <td th:utext="${item.getRank().getShortNameRank()}"></td>
  <td th:utext="${item.getSurname()+ ' '+item.getName().charAt(0)+' '+item.getThirdName().charAt(0)+''}"></td>
  <td th:utext="${item.getFormatAccounting()}"></td>
  <td th:utext="${(sort.equals('ATO'))?item.getFormatQuota2():item.getFormatQuota()}"></td>
  <td th:utext="${(sort.equals('ATO'))?item.getQuotaType2():item.getQuotaType()}"></td>
  <td th:utext="${item.getFamilyCount()}"></td>
  <td th:utext="${item.getRoomCount()}"></td>
  <td th:utext="${item.getWork().getAccountingPlace()}"></td>
  <td th:utext="${item.getWork().getWorkPlace()}"></td>
  <td th:utext="${item.getExperience()}"></td>
</tr>
</tbody>
</table>
</div>
</form>
<div class="row mt-5">
  <div class="col-md-12">
    <div class="copyright">
      <p>Military institute of
        telecommunications and information technologies named after Heroes of Kruty</p>
      <p>2022 -
        <script language="JavaScript" type="text/javascript">
          now = new Date;
          theYear = now.getYear();
          if (theYear < 1900)
            theYear = theYear + 1900;
          document.write(theYear)
        </script>. Powered by Sachuk</p>
    </div>
  </div>
</div>
</div>
</body>

<script th:src="@{../vendor/jquery-3.2.1.min.js}"></script>
<!-- Booth:tstra@{ / JS-->
<script th:src="@{../vendor/bootstrap-4.1/popper.min.js}"></script>

```

```

<script th:src="@ { ../vendor/bootstrap-4.1/bootstrap.min.js }"></script>
<!-- Venth:dor J@ { ../ -->
<script th:src="@ { ../vendor/slick/slick.min.js }"></script>
<script th:src="@ { ../vendor/wow/wow.min.js }"></script>
<script th:src="@ { ../vendor/ansition/ansition.min.js }"></script>
<script th:src="@ { ../vendor/bootstrap-progressbar/bootstrap-progressbar.min.js }"></script>
<script th:src="@ { ../vendor/counter-up/jquery.waypoints.min.js }"></script>
<script th:src="@ { ../vendor/counter-up/jquery.counterup.min.js }"></script>
<script th:src="@ { ../vendor/circle-progress/circle-progress.min.js }"></script>
<script th:src="@ { ../vendor/perfect-scrollbar/perfect-scrollbar.js }"></script>
<script th:src="@ { ../vendor/chartjs/Chart.bundle.min.js }"></script>
<script th:src="@ { ../vendor/select2/select2.min.js }"></script>
<script th:src="@ { ../vendor/vector-map/jquery.vmap.js }"></script>
<script th:src="@ { ../vendor/vector-map/jquery.vmap.min.js }"></script>
<script th:src="@ { ../vendor/vector-map/jquery.vmap.sampledata.js }"></script>
<script th:src="@ { ../vendor/vector-map/jquery.vmap.world.js }"></script>
<!--<script th:src="@ { ../js/sidebarOpen.js }"></script>-->
<script th:src="@ { ../js/main.js }"></script>
</html>
<!-- end document-->

```