МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут автоматики, кібернетики та
обчислювальної техніки
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри комп'ютерних
технологій та економічної кібернетики
д. е. н., проф. П. М. Грицюк

_____
«____» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітньо-кваліфікаційного рівня «бакалавр»

«АВТОМАТИЗОВАНА СИСТЕМА КОНТРОЛЮ ВИКОНАННЯ ЗАВДАНЬ ІНФОРМАЦІЙНО - ОБЧИСЛЮВАЛЬНОГО ЦЕНТРУ ЗВО»

Виконав:
здобувач вищої освіти, групи ІСТ-21інт
Приходчук Василь Олександрович

Керівник:
к.т.н., доцент Барановський С.В.
Рецензент:
к.т.н., доцент Гладка О.М.

Добавлено примечание ([h1]): Титульна сторінка НЕ відповідає вимогам!!. Див. МетодВказівки, Додаток З!!!

Рівне – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних.

СУБД – система управління базами даних.

СКБД – Система керування базами даних .

ІОЦ – інформаційно обчислювальний центр.

IDE – Інтегроване середовище розробки.

ПЗ – Програмне забезпечення.

ВСТУП

Успішне виконання завдань є ключовим фактором для досягнення успіху в будь-якій сфері діяльності. Проте, без ефективної системи контролю виконання завдань, навіть найкращі проекти можуть зазнавати затримок та проблем з якістю. У зв'язку з цим, створення системи контролю виконання завдань є важливим етапом в управлінні проектами.

Неодмінною складовою ефективного управління є функція контроль. Контроль дає змогу суб'єкту управління регулярно отримувати інформацію про стан справ у керованій системі, на основі чого він приймає управлінські рішення, спрямовані або на закріплення досягнутого успіху, або на усунення причин, що перешкоджають його досягненню.

Контроль є необхідним в силу тих обставин, що на керований об’єкт завжди діють збурювальні фактори, і виникає потреба мати оцінну інформацію, щоб адекватним чином відредагувати на нову ситуацію. Контроль здійснюється з метою досягнення узгодженості і синхронізації зусиль виконавців, виявлення суперечливих тенденцій і протиріч у їх діяльності.

Система контролю виконання завдань — це логічна структура формальних та неформальних процедур, що передбачена для аналізу та оцінки ходу виконання завдання та оцінки ефективності управління ресурсами, витратами, зобов’язаннями протягом всього терміну його реалізації (періодичний моніторинг поточної діяльності, порівняння обсягів та витрат матеріалів людський ресурсів, виявлення відхилень з метою усунення додаткових витрат).

Це також процес, в якому керівник проекту встановлює - чи досягаються поставлені цілі, виявляє причини, які дестабілізують хід роботи й обґрунтовує прийняття управлінських рішень, що коригують виконання робіту. Основними задачами контролю є перевірка фактичних даних, зіставлення їх із плановими і виявлення відхилень.

Предметом контролю є факти і події, перевірка виконання конкретних рішень, з'ясування причин відхилення, оцінка ситуації, прогнозування наслідків. Контроль передбачає постійне спостереження за просуванням проекту.

Контроль як завершальна стадія процесу управління безпосередньо впливає на ефективність здійснення інших управлінських функцій — планування, організування, мотивування. Так, навіть найкращі плани не будуть здійснені, якщо не забезпечити контроль за їх реалізацією.

Для вирішення проблеми контролю виконання завдань необхідно створити автоматизовану систему контролю виконання завдань, з її допомогою можливо буде ефективно здійснювати контроль, хід виконання та затримки у виконанні завдань.

Метою даної дипломної роботи створення автоматизованої системи контролю виконання завдань для Інформаційно–обчислювального центру ЗВО, її важливості, основних функціональних можливостей та етапів впровадження. Також будуть проаналізовані результати роботи системи та її вплив на ефективність управління проектами..

Об’єктом дипломної роботи є автоматизована система контролю виконання завдань для Інформаційно–обчислювального центру ЗВО.

Предметом дослідження є проектування бази даних та створення інформаційної системи.

Для досягнення вказаної вище мети роботи визначені такі **завдання**:

Проаналізувати функції ІОЦ.

- Дослідити існуючу в НУВГП процедуру призначення і контролю за виконанням завдань.
- Створити модель (схему) бази даних для ІС.
- Виконати проектування та розробити ІС.
- розробити проект (план) впровадження.

Дослідження області можна провести за допомогою різних методів, таких як:

– Аналіз документів: дослідження документів.

– Аналіз вже існуючих систем.

– Анкетування: опитування різних груп людей, пов’язаних з виконанням завдань тобто співробітників.

– Статистичний аналіз: використання статистичних методів може допомогти виявити проблеми в сфері виконання завдань та забезпечити науково обґрунтоване прийняття рішень у цій області.

– Моделювання: використання моделей та імітаційних систем може допомогти вивчити різні варіанти розвитку подій при виконанні завдань.

Ці методи можна поєднувати між собою для отримання комплексної інформації та належного аналізу контролю виконання завдань.

РОЗДІЛ 1. АНАЛІЗ ДІЯЛЬНОСТІ ІНФОРМАЦІЙНО - ОБЧИСЛЮВАЛЬНОГО ЦЕНТРУ НУВГП

ІОЦ є структурним підрозділом Національного університету водного господарства та природокористування (далі – Університет). Створений відповідно до рішення вченої ради Університету від 2019 року та затверджено наказом ректора.

ІОЦ підпорядковується ректору Університету, а в порядку оперативного управління – проректору з науково-педагогічної, методичної та виховної роботи. Робота центру ґрунтується на основі чинного законодавства України, локально нормативних актів Університету, стратегії розвитку університету і відповідно до затверджених річних планів роботи ІОЦ.

Відділ забезпечує впровадження та функціонування сучасних інформаційних технологій для автоматизованої обробки, зберігання, систематизованої інформації, здійснення в установленому порядку інформаційного обміну в інтересах діяльності Університету.

Діяльність Інформаційно-обчислювального центру здійснюється за двома основними напрямами:

- експлуатація та розвиток інформаційних систем (технічне обслуговування, ремонт і заміна комп’ютерної техніки, серверного та периферійного обладнання; технічне обслуговування, модернізація та налаштування комп’ютерних мереж)

- підтримка та розвиток існуючих, впровадження нових інформаційних технологій (сайт університету, службова електронна пошта, електронний документообіг, навчальна платформа Moodle, електронний деканат, цифровий репозиторій, мобільний додаток тощо).

1.1. Організаційна структура функції ІОЦ НУВГП

До складу інформаційно-обчислювальний центру входять 3 відділи, а саме:

- відділ інформаційних технологій;
- відділ експлуатації комп'ютерних систем;
- відділ забезпечення інноваційних технологій навчання.

Відділ інформаційних технологій:

- відповідає за розробку, інтегрування, адміністрування та технічну підтримку WEB-сайтів університету;
- забезпечує працівників та студентів корпоративними обліковими записами;
- здійснює діяльність щодо забезпечення кібербезпеки інформаційних технологій та систем університету.

Відділ експлуатації комп'ютерних систем відповідає за комп'ютерне забезпечення навчального процесу, впровадження нової обчислювальної техніки, установка, тестування і оновлення операційних систем та прикладних програм, проведення інвентаризації комп'ютерної техніки, експлуатація та розвиток комп'ютерної мережі університету.

Відділ забезпечення інноваційних технологій навчання відповідає за:

- підтримку та розвиток навчальної системи Moodle;
- організацію системи Unichek для перевірки навчально-наукових робіт на предмет схожості;
- синхронізацію даних між локальною інформаційною системою на основі програм пакету "Деканат" та "Єдиною державною електронною базою з питань освіти" (ЄДЕБО)

ІОЦ очолює директор. На посаду директора центру призначається фахівець з повною вищою освітою. Директор ІОЦ здійснює загальне керівництво центру, організує його роботу забезпечує виконання покладених на нього завдань, а саме:

- приймає самостійні рішення з оперативного управління роботою ІОЦ;
- здійснює взаємодію з керівниками всіх структурних підрозділ Університету;
- розподіляє обов’язки, встановлює коло відповідальності працівників ІОЦ та контролює їх роботу;

- керує підготовкою та розробкою стратегічного розвитку, планів роботи ІОЦ відповідно до планів роботи Університету;
- Організовує та контролює своєчасне та якісне виконання працівниками їх зобов’язань;
- Забезпечує функціонування та розвиток інформаційної мережі університету.

Кількісний і якісний склад працівників та структура ІОЦ визначається штатним розписом, затвердженим ректором Університету за поданням директора центру. До складу ІОЦ входять відділи визначені штатним розписом. Права та обов’язки працівників визначається посадовими інструкціями, затвердженими ректором Університету.

Кожний відділ очолює начальник, які керують та організовують роботу відділів інформаційно-обчислювальний центру. У відділах працюють провідні спеціалісти відповідно до якого відділу вони відносяться.

Базою практики було обрано ІОЦ, тому за моїм бажанням було обрано відділ експлуатації комп'ютерних систем, його склад я розберу докладніше.

Відділ експлуатації комп'ютерних систем відповідає за комп'ютерне забезпечення навчального процесу, впровадження нової обчислювальної техніки, установка, тестування і оновлення операційних систем та прикладних програм, проведення інвентаризації комп'ютерної техніки, експлуатація та розвиток комп'ютерної мережі університету.

Склад відділу складається із:

- Начальник відділу;
- Провідного інженера із обліку та списання комп'ютерних систем;
- Провідний інженер з сервісу периферійного обладнання;
- Оператор електронно-обчислювальних машин;
- Оператор електронно-обчислювальних машин.

Технічна підтримка програмного забезпечення та комп'ютерних систем:

- Провідний інженер з експлуатації комп'ютерних систем;

Технічна підтримка комп'ютерних мереж:

- Провідний інженер з експлуатації комп'ютерних мереж;
- Провідний фахівець із комунікаційних технологій.

Схема організаційної структури інформаційно-обчислювальний центру зображено на рис 1.1.

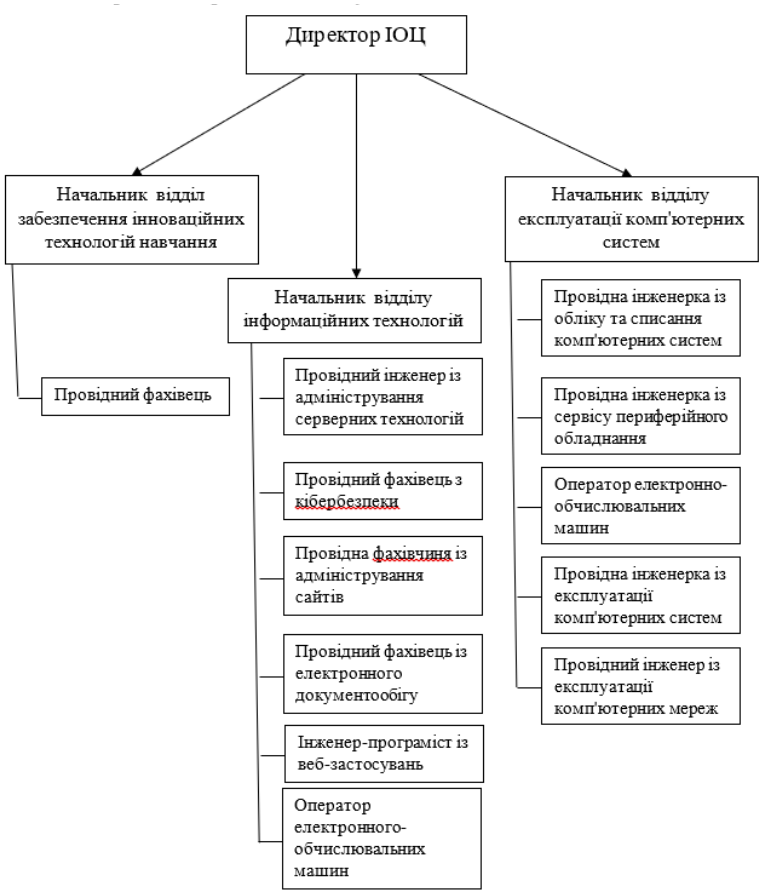


Рис 1.1. Організаційна структура ІОЦ

1.2. Завдання та функції ІОЦ НУВГП

Основними функціями ІОЦ є:

- Реалізація стратегії розвитку інформаційних технологій Університету; забезпечення інформаційних потреб та інформаційної підтримки навчальної, науково-технічної, господарської та соціальної діяльності Університету.
- Експлуатація, обслуговування та розвиток комп'ютерної мережі, комп'ютерних систем, серверного та периферійного комп'ютерного обладнання Університету.
- Впровадження та функціонування сучасних інформаційних технологій для автоматизованої обробки, зберігання, систематизації та аналізу інформації, здійснення в установленому порядку інформаційного обміну в інтересах діяльності Університету.
- Організація системи захисту інформаційних ресурсів Університету та інформації, вимога щодо захисту якої встановлена законом.

Інформаційно-обчислювальний центр відповідно до покладених на нього завдань:

- Забезпечує функціонування комп'ютерної мережі НУВГП, здійснює аналітичну роботу з її експлуатації та модернізації.
- Проводить контроль та адміністрування комп'ютерних систем Університету, встановлення операційних систем, встановлення та видалення програмного забезпечення.
- Забезпечує технічне обслуговування і налагодження периферійного комп'ютерного обладнання та устаткування.
- Проводить аналітичну роботу щодо технічного стану існуючого комп'ютерного обладнання. Здійснює його модернізацію.

- Координує списання комп'ютерної техніки, непридатної для подальшого використання.
- Здійснює технічне обслуговування, встановлення, адміністрування та системне супроводження серверів комп'ютерної мережі Університету.
- Забезпечує раціональну організацію накопичення, зберігання і ведення баз даних та інформаційних масивів, фонду електронних документів.
- Здійснює адміністрування корпоративних інформаційних систем Університету.
- Забезпечує стабільне функціонування, адміністрування та безпеку веб-ресурсів Університету.

ІОЦ має право:

- Надавати пропозиції щодо визначення - напряму розвитку інформатизації Університету та впровадження сучасних інформаційних технологій для забезпечення діяльності Університету.
- Готувати документацію з питань, що стосується інформаційних технологій, кібербезпеки, публічності інформації, використання інформаційної бази даних, а також з питань забезпечення доступу до веб-ресурсів Університету.
- Взаємодіяти з іншими структурними підрозділами Університету, підрозділами органів, установ та закладів у межах наданих повноважень.
- За дорученням керівництва Університету представляти Університет на переговорах з питань впровадження та розвитку інформаційних систем і технологій.

- Забезпечуватись приміщеннями, матеріалами й обладнанням, необхідними для функціонування центру та доступом до всіх приміщень Університету з метою обслуговування, профілактики і ремонту обладнання комп'ютерних систем і мереж.

1.3. Аналіз існуючої в ІОЦ системи планування та контролю виконання завдань

Згідно положення про ІОЦ служба підтримки призначена для надання допомоги та технічної підтримки кінцевим користувачам. Її основна мета полягає у вирішенні проблем, які виникають у користувачів під час використання обчислювальних ресурсів, програмного забезпечення або послуг, що надаються ІОЦ.

Як правило, служба підтримки укомплектована командою кваліфікованих фахівців, які відповідають за відповіді на питання, вирішення технічних проблем та усунення несправностей, з якими можуть зіткнутися користувачі. Співробітники служби підтримки часто використовують спеціалізоване програмне забезпечення для відстеження та управління проблемами кінцевих користувачів, і вони можуть нести відповідальність за передачу складних проблем групам підтримки вищого рівня або розробникам продуктів.

Служби підтримки можуть надаватися по різних каналах, включаючи телефон, електронну пошту, чат або портал самообслуговування. Вони відіграють вирішальну роль у забезпеченні того, щоб користувачі мали позитивний досвід роботи з продуктом або послугою, та можуть допомогти підвищити задоволеність та лояльність користувачів.

Рішення використовувати систему Service Desk пов'язане з досить типовими очікуваннями від користувачів системи, фахівців, які забезпечують технічної підтримки, а також власників бізнесу.

Давайте розбиратися в цих очікуваннях і проблемах, що виникають під час їхнього перетину.

При організації зручної служби технічної підтримки користувач повинен чітко знати, де і як залишати заявку. Часто користувачеві простіше взяти телефон і зателефонувати до того фахівця, якого він знає або до якого звертався раніше. У такому разі може виникнути втрата заявок або перевантаженість якогось фахівця першої лінії. Очевидним вирішенням цієї проблеми є розробка інструкції користувача. Інструкція повинна включати:

Очевидно, що основним критерієм якості роботи служби технічної підтримки є вирішення завдань. І тут дуже важливу роль відіграє автоматизація зворотного зв'язку з користувачем. Адже одна річ, коли заявка вирушає в надра якоїсь «чорної скриньки» і користувач змушений перебувати в тяжкому очікуванні. І зовсім інше, коли користувач отримує негайний відгук і його повідомляють при будь-яких діях з його проблеми. Однак не завжди ресурси технічних фахівців дозволяють приділяти увагу ввічливому спілкуванню та й просто вчасно брати заявки на роботу в період великого завантаження.

Оператору першої лінії дуже хотілося б зібрати максимум інформації від користувача, поставивши мінімум запитань. А також уникнути помилок у відповідях користувачів на питання, в яких вони можуть бути некомпетентними.

Часто виникає така ситуація, коли заявка, яка пішла далі з першої лінії технічної підтримки, просто втрачається. Таке може статися, якщо перша лінія, передаючи заявку, перестає бути відповідальною за неї, а спеціалісти другої лінії, виконавши свою частину роботи, не переводять завдання назад або ескалують її далі, але без призначення іншого відповідального. Таким чином, завдання «гуляє» між фахівцями, оператор першої лінії про неї давно забув, а замовник не дочекається вирішення питання.

В процесі дослідження існуючої системи служби підтримки були підкресленні наступні проблеми: застарілий інтерфейс – це перше що бачить користувач. Звісно це досить суб’єктивна річ, але може негативно вплинути на нього.

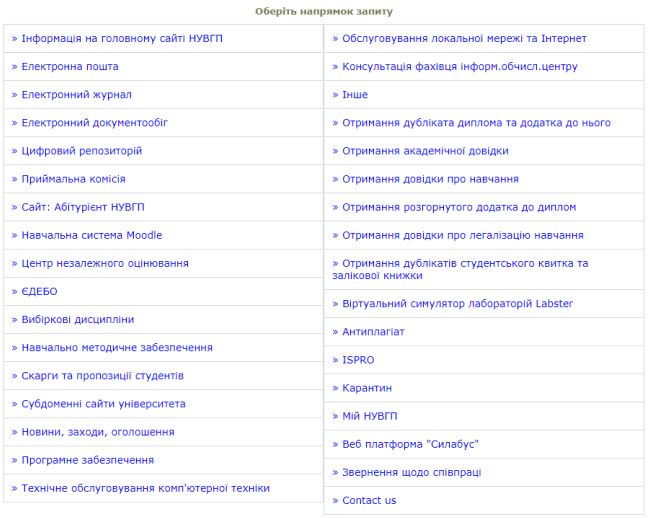


Рис. 1.2. Напрямок теми запиту

Можна побачити, що інтерфейс досить не зручний в плані пошуку теми запиту, велика кількість тем для запиту може розгубити користувача.

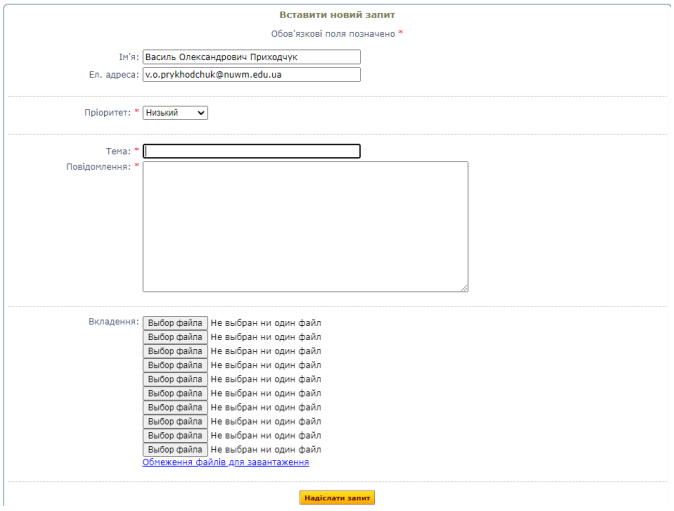


Рис 1.3. Шаблон заповнення запиту

Як можна побачити процес створення запиту складається з двох етапів це вибір теми запиту та заповнення шаблону запиту.

1.4. Аналіз проблеми інформаційного забезпечення системи планування та контролю виконання завдань в ІОЦ НУВГП

У рамках ІОЦ системи планування та контролю виконання завдань проводиться контроль над різними аспектами роботи. Нижче наведено інформацію про завдання, які підлягають контролю в ІОЦ:

- налагодження периферійного обладнання;
- встановлення, оновлення та налагодження програмного забезпечення;
- монтаж інтернет мережі;
- усунення аварій в інтернет мережі;
- налагодження БФП;
- діагностика ПК для подальшого ремонту.

Завдання, які виконують працівники відділу експлуатації комп'ютерних систем, складають значну частину їх робочих обов'язків. Важливість цих завдань та відповідність термінам їх виконання часто стають проблемою. Однак головною причиною розробки системи планування та контролю виконання завдань є необхідність збереження бази знань щодо виконаних завдань. Кожне завдання повинно містити в собі повну інформацію про виконані роботи, включаючи використані пристрої, матеріали, приміщення, терміни та статус завдання. Однак, існуюча система контролю не має необхідного функціоналу для зберігання цієї бази знань.

Тому розробка системи планування та контролю виконання завдань стає важливим завданням. Ця система буде забезпечувати можливість документування і зберігання повної інформації про виконані завдання, а також надавати засоби для відстеження статусу завдань. Це дозволить зберігати цінні знання про виконані роботи, сприятиме ефективному плануванню та контролю над виконанням завдань у майбутньому.

Доцільно використовувати багаторівневу системи технічної підтримки часто організована за наступним багаторівневим принципом:

- Користувач — звертається з питанням в службу підтримки за телефоном або за допомогою електронної заявки (електронна пошта, або спеціальні сервіси подачі заявок).
- Оператор (1-а лінія підтримки, так званий cell-centr) — реєструє звернення, при можливості допомагає користувачеві самостійно, або ескалує (передає та контролює виконання) заявку на другу лінію підтримки.
- Друга лінія підтримки — отримує заявки від першої лінії, опрацьовує їх, за необхідності залучаючи до вирішення проблеми фахівців за напрямом робіт.

РОЗДІЛ 2. МОДЕЛІ ТА МЕТОДИ АВТОМАТИЗАЦІЇ СИСТЕМ КОНТРОЛЮ ВИКОНАННЯ ЗАВДАНЬ

2.1. Характеристика існуючих автоматизованих систем контролю виконання завдань

Система контролю виконання завдань - це інструмент, методологія або процес, що допомагає відстежувати, організовувати і контролювати виконання завдань. Ось кілька систем контролю виконання завдань:

1. Kanban-системи: Такі системи, як Trello, KanbanFlow або JIRA, використовують дошку з колонками, що представляють різні стадії виконання завдань (наприклад, "В очікуванні", "У процесі", "Завершено"). Користувачі можуть створювати завдання, переміщувати їх по колонках та відстежувати прогрес виконання.
2. GTD-системи (Getting Things Done): Такі системи, як Nirvana, Todoist або Evernote, базуються на методології "Getting Things Done" Девіда Аллена. Вони надають можливість створювати список завдань, визначати пріоритети, додавати теги та контексти, планувати та відстежувати виконання завдань.
3. Agile-системи: Ці системи, наприклад, Scrum або Kanban-інструменти для розробки програмного забезпечення, дозволяють організовувати та відстежувати виконання завдань у команді. Вони надають засоби для планування спринтів, створення та призначення завдань, відстежування прогресу та спільної роботи.
4. Системи управління проектами: Ці системи, такі як Microsoft Project, Asana або Basecamp, дозволяють керувати проектами в цілому і відстежувати виконання завдань. Вони надають інструменти для розподілу завдань, призначення відповідальних осіб, встановлення термінів та відстежування прогресу.

До інформаційної системи контролю виконання завдань для стабільної і зручної роботи користувачів та самої системи пред'являється певний ряд вимог. Проаналізувавши вже існуючі проекти можна сформувати вимоги до системи:

- зручну систему введення даних користувача;
- виведення на екран дисплея необхідної інформації по запиту користувача в зручному для перегляду вигляді;
- формування звітної інформації
- обробку інформаційних запитів користувача;
- захист інформації від несанкціонованого доступу шляхом введення системи паролів;

Перш ніж приступити до проектування бази даних, необхідно визначити основні функціональні вимоги, які висуваються до системи, тобто визначити коло завдань системи і програми бази даних і склад її користувачів.

2.2. Функціональні вимоги до Інформаційної системи автоматизованого контролю виконання завдань

Для коректної роботи користувачів в системі контролю виконання заявок, доцільно використати розмежування прав доступу в інформаційній системі, здійснюється шляхом дозволу / заборони на редагування та перегляд певної інформації певної групи користувачів.

В інформаційній системі контролю виконання заявок функціонують три групи користувачів:

- Користувач - виконавець.
- Користувач - обліку пристроїв.
- Користувач - модератор.

Отже кожен з наведених типів користувачів має свої певні права доступу в системі. Одні користувачі мають доступ до додавання, редагування та видалення інформації, а інші мають лише можливість створювати завдання та перегляд інформації.

Базовий тип користувача – «Користувач – виконавець» має доступ лише до створення завдань їх наповнення та завершення, створення зв'язків між пристроями (формування топології мережі) .Також цей тип користувача має доступ до перегляду бази пристрої

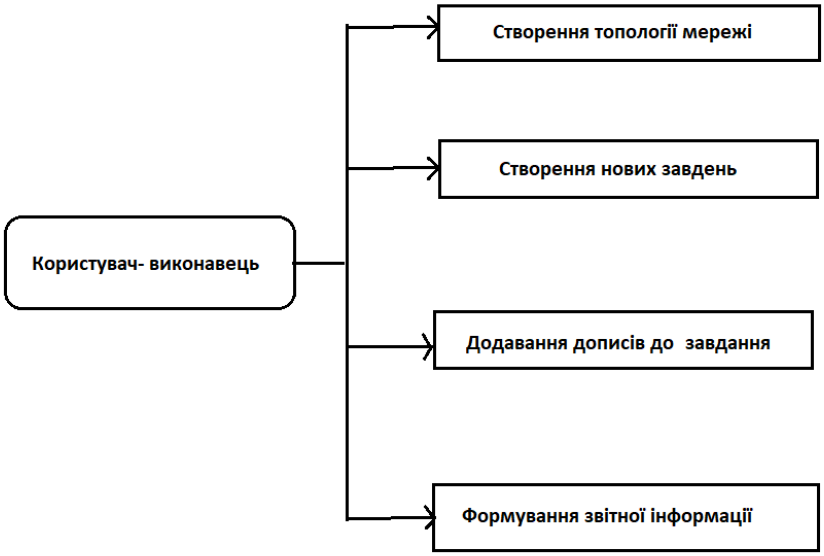


Рис. 2.1 Функціональні можливості користувача виконавець.

Наступний тип користувача інформаційної системи контролю виконання завдань - «користувач - обліку пристроїв», даний користувач має функціональні можливості «користувача-виконавець», та правами доступу до додавання, редагуванн та видалення даних по активному обладнанню (персональні комп'ютери, принтери, БФП, комутатори, роутери та інше).

Рис. 2.2 Функціональні можливості користувача обліку пристроїв.

Наступний і останній тип користувачів, який має адміністраторські права та може впливати на інших користувачів – «користувач-модератор». Даний тип користувачів мають права модератора, тобто даний користувач володіє усіма вище перерахованими можливостями та з можливістю створювати нових користувачів, редагувати їх інформацію. Модератор це також користувач який здійснює контролю виконаних завдань, в разі якщо завдання наприклад не правильно сформоване перед закриттям то модератор може повернути до виконання.

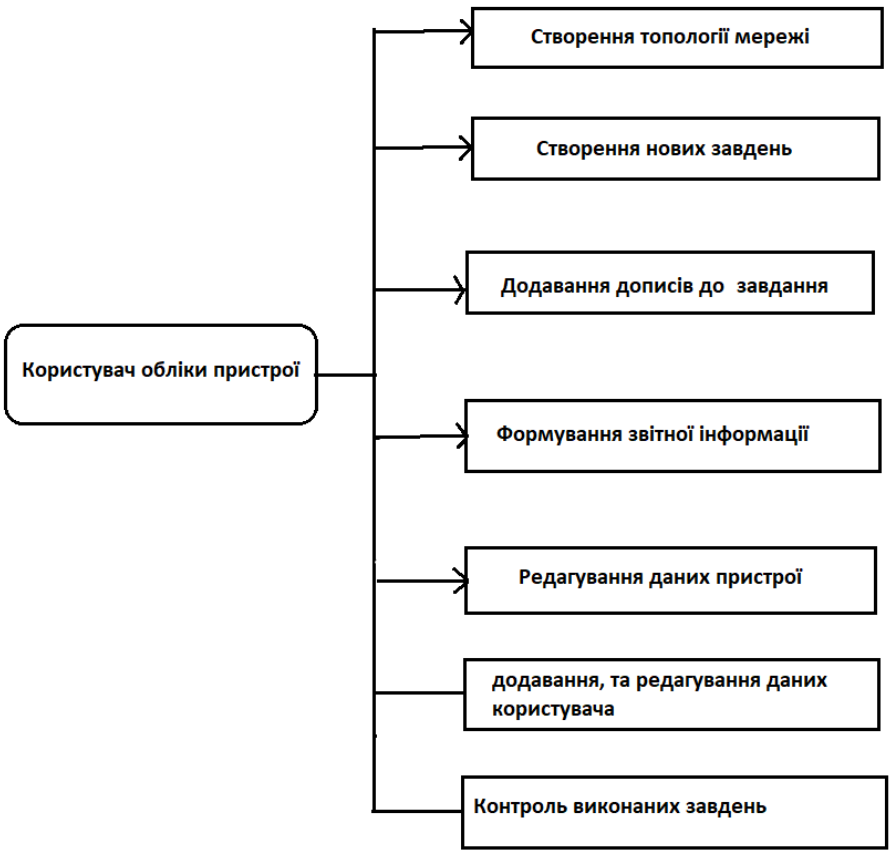


Рис. 2.3 Функціональні можливості користувача модератора.

2.3. Модель інформаційних потоків та схема бази даних

Модель інформаційних потоків системи управління заявками є важливою складовою для ефективного та організованого управління заявками в організації. Вона дозволяє відстежувати рух інформації в системі, зокрема збір, обробку та передачу даних, пов'язаних з заявками.

Модель інформаційних потоків включає опис структури системи, зокрема, складу та взаємодії її компонентів, а також регламентує обмін інформацією між ними. Така модель визначає процеси обробки заявок, відслідковування їх статусу та зв'язку з користувачами, що звернулися зі заявкою.

Нижче наведений загальний опис моделі інформаційних потоків системи управління заявками, який можна використовувати як загальну базу для розробки власної моделі, враховуючи конкретні особливості вашої системи.

1. Вхідні дані: Це інформація, яка надходить до системи управління заявками з зовнішніх джерел або внутрішніх користувачів. Вхідні дані можуть включати деталі заявки, такі як тип заявки, прізвище та контактна інформація заявника, опис проблеми тощо.
2. Збір і реєстрація: Цей етап включає збір інформації заявок і реєстрацію їх в системі. На цьому етапі можуть бути встановлені відповідні поля для введення даних, а також присвоєння унікального ідентифікатора кожній заявці.
3. Обробка: Після реєстрації заявки в системі вона переходить на етап обробки, де виконується аналіз і класифікація заявки. Залежно від типу заявки та її пріоритету, вона може бути направлена до відповідного відділу або спеціаліста для подальшої обробки.
4. Призначення і розподіл: На цьому етапі заявка призначається відповідальній особі або команді для подальшого вирішення проблеми. Це може бути зроблено автоматично на основі правил або ручним вибором адміністратора системи.
5. Виконання: Після призначення заявки відповідальної особи, заявка переходить в статус виконується.

Для наочного відображення поточної роботи системи контролю виконання заявок використано - Діаграми потоків даних (Data Flow Diagram). Діаграми потоків даних DFD - це графічний інструмент для моделювання потоків даних в

інформаційній системі. Вони використовуються для візуалізації процесів обробки даних, потоків інформації та взаємодії між компонентами системи. Діаграми потоків даних є популярними серед аналітиків систем та розробників програмного забезпечення, оскільки дозволяють зрозуміти структуру і функціонування системи, а також виявити потенційні проблеми або недоліки.

Основними елементами діаграм потоків даних є:

- Процеси: Представлені у вигляді прямокутників і вказують на операції або дії, які виконуються над даними. Кожен процес має вхідні та вихідні потоки даних.
- Потоки даних: Представлені у вигляді стрілок і показують напрямок переміщення даних в системі. Вони вказують на дані, які вводяться або виводяться з процесів.
- Сховища даних: Представлені у вигляді прямокутників з подвійним контуром і представляють місце, де зберігаються дані. Наприклад, бази даних, файли або сховища в пам'яті.
- Зовнішні сутності: Представлені у вигляді прямокутників з нижньою пунктирною лінією і вказують на джерела і приймачі даних, що взаємодіють з системою. Це можуть бути користувачі, інші системи або зовнішні сервіси.

Діаграми потоків даних надають візуальну модель потоку даних в системі, допомагаючи зрозуміти, як дані обробляються і перетворюються в процесах.

Діаграми потоків даних інформаційної системи контролю виконання завдань представлено наступному рис. 2.4.1.

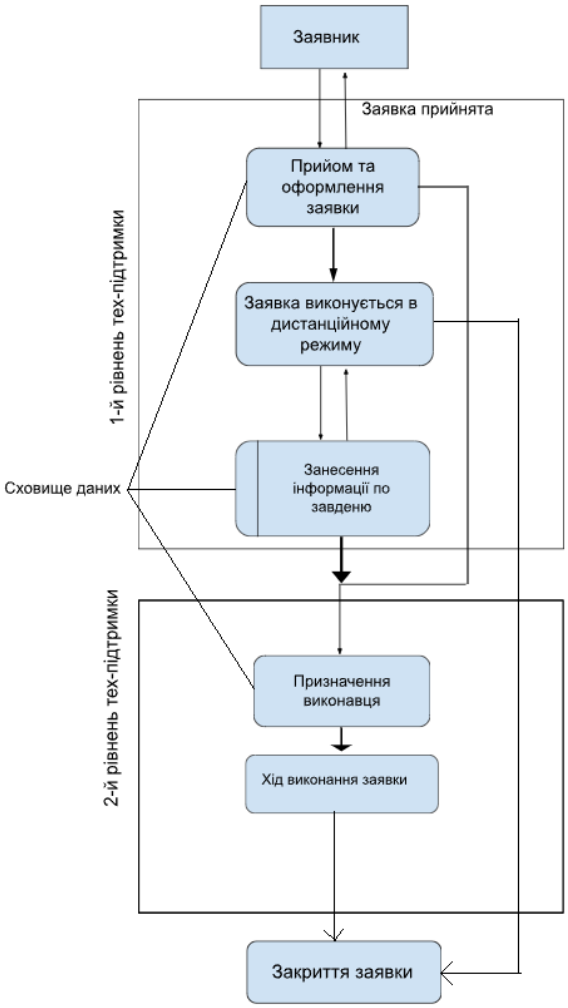


Рис. 2.4. Діаграма Потоків даних системи контролю виконання завдань

2.4. Проектування бази-даних інформаційної системи

Обрана архітектура інформаційної системи, що проектується, реалізує архітектуру «клієнт-сервер» та передбачає, що БД може бути розміщена як на сервері локальної комп'ютерної мережі так і на віддаленому (remote) сервері. На

робочих станціях встановлюють клієнтську програму, що формує і надсилає запити віддаленого серверу з БД, який забезпечує виконання запиту і видачу клієнтові результату. Вся обробка інформації відбувається на віддаленому сервері.

Для реалізації бази даних вибрано реляційну структури БД. Реляційна база даних (РБД) - це тип бази даних, яка зберігає дані у вигляді таблиць з рядками і стовпцями. У реляційній базі даних дані організовані у вигляді таблиць, які складаються з рядків і стовпців. Кожна таблиця має назву і складається зі стовпців, які визначають типи даних, що зберігаються в цій таблиці. Кожен рядок таблиці представляє запис з конкретними значеннями для кожного стовпця. Кожна таблиця має первинний ключ, який є унікальним ідентифікатором для кожного запису в таблиці.

Один з основних принципів реляційної бази даних - це можливість встановлення зв'язків між таблицями за допомогою зовнішніх ключів. Зовнішній ключ у таблиці посилається на первинний ключ іншої таблиці, утворюючи таким чином зв'язок між двома таблицями.

Реляційні бази даних використовують мову структурованих запитів SQL (Structured Query Language) для роботи з даними. За допомогою SQL можна створювати, змінювати і видаляти дані з таблиць, а також виконувати складні запити для отримання необхідної інформації.

Реляційні бази даних широко використовуються в різних галузях, включаючи бізнес, наукові дослідження, фінанси, логістику та інші сфери, де потрібно зберігати та організовувати великі обсяги даних.

Отже Реляційна структура бази даних має декілька важливих переваг:

1. Структурованість і організація даних: Реляційна база даних зберігає дані у вигляді таблиць з рядками і стовпцями, що дозволяє структурувати і організовувати дані у логічний спосіб. Це полегшує розуміння структури бази даних і спрощує роботу з даними.

2. Інтегрітет даних: Реляційна модель бази даних використовує ключі для встановлення зв'язків між таблицями і забезпечення цілісності даних. Це означає, що дані можуть бути зв'язані із використанням унікальних ідентифікаторів, а також встановлюються правила, які гарантують, що дані будуть зберігатися у консистентному стані.
3. Ефективний доступ до даних: Реляційна база даних використовує мову структурованих запитів SQL, яка дозволяє виконувати різноманітні операції з даними, такі як вибірка, сортування, фільтрація і з'єднання. Це дозволяє швидко та ефективно отримувати необхідну інформацію з бази даних.
4. Гнучкість і розширюваність: Реляційні бази даних можуть бути легко змінені і розширені, додаванням нових таблиць або зміною вже існуючих структур. Це дає можливість адаптувати базу даних до змінних потреб бізнесу або додавати нові функціональні можливості.
5. Можливість роботи з великими обсягами даних: Реляційні бази даних можуть обробляти і зберігати великі обсяги даних ефективно і безпечно.

Перший етап процесу проектування бази даних називається концептуальним проектуванням бази даних. Даний етап полягає у створенні концептуальної моделі даних предметної області. Дана модель даних створюється на основі функціональних вимог користувачів системи. Концептуальне проектування бази даних не залежить від подробиць її реалізації, як тип обраної СУБД, набір створюваних прикладних програм, використовувані мови програмування, тип обраної обчислювальної платформи, а також від будь-яких інших особливостей фізичної реалізації. Концептуальне проектування – створення концептуального уявлення бази даних, що включає визначення типів найважливіших сутностей та існуючих між ними зв'язків і атрибутів. Послідовність етапів проектування концептуальної моделі даних: визначення сутностей; визначення взаємозв'язків між сутностями; визначення атрибутів сутностей; завдання первинних і альтернативних ключів.

Для бази даних системи управління контролю виконання заявок визначені наступні 9 сутностей:

1) користувачі;

2) посади користувачів;

3) завдання;

4) статус завдання;

5) події завдання;

6) пристрої;

7) тип пристрою;

8) приміщення;

9) матеріали;

10) топологія.

У кожної сутності існує унікальний первинний ключ, який служить для її ідентифікації. При створенні сутності необхідно визначити групу атрибутів, які можуть стати її первинним ключем, і вибрати підмножину атрибутів для включення до цього ключа. Первинний ключ повинен бути сконструйований таким чином, щоб за значеннями цих атрибутів можна було однозначно ідентифікувати конкретний екземпляр сутності. Крім того, жоден з атрибутів, що входять до складу первинного ключа, не повинен мати значення "нуль". Значення атрибутів, що складають первинний ключ, мають залишатися постійними. Якщо значення змінюється, це означає, що ми маємо справу з іншим екземпляром сутності.

Щоб вибрати первинний ключ, можна додати до сутності додатковий атрибут і зробити його ключем. Це означає, що для визначення первинного ключа часто використовують унікальні номери, які можуть бути автоматично згенеровані системою під час додавання екземпляра сутності до бази даних. Використання унікальних номерів сприяє полегшенню процесу індексації та пошуку в базі даних.

Список всіх сутностей бази даних системи контролю виконання завдань подані у вигляді табл. 2.1.

Таблиця 2.1 – Список всіх сутностей БД.

user	Користувачі
posadu_user	Посади користувачі
zayavka	Завдання
status	Статус завдання
podii	Події завдання
pristroii	Пристрої
typ_pristrou	Тип пристрою
kab	Приміщення
materialu	Матеріали
topologii	Топологія

Далі визначимо атрибути для кожної сутності, а також поставимо для всіх сутностей первинні ключі та тип даних кожного поля сутностей. Відомості про таблиці бази даних представлені нижче у вигляді таблиць (табл. 2.2, 2.11)

Табл 2.2 – Користувачі системи.

№	Назва поля	Атрибут	Тип
1	id_user	Унікальний ідентифікатор	int(11)
2	id_posadu	Посада користувача	int(11)
3	login	Логін користувача для авторизації	varchar(30)
4	pass	Пароль користувача для авторизації	varchar(20)
5	pib	ПІБ користувача	varchar(100)
6	telefon	Телефон користувача	varchar(15)
7	poshta	Пошта користувача	varchar(30)
8	rolb	Роль користувача	varchar(20)

Таблиця 2.3 – Посади користувачів системи.

№	Назва поля	Атрибут	Тип
1	id	Унікальний ідентифікатор	int(11)
2	posada	Посада користувача	varchar(255)

Таблиця 2.4 – Завдання системи.

№	Назва поля	Атрибут	Тип
1	id_zayavka	Унікальний ідентифікатор	int(11)
2	id_status	Статус завдання	int(11)
3	nazva	Назва завдання	varchar(255)
4	opus	Опис завдання	varchar(255)
5	iniciator	Ініціатор завдання	varchar(50)
6	data_stvorenya	Дата створення завдання	date
7	data_zakrutya	Дата закриття завдання	date

Таблиця 2.5 – Статус завдання системи.

№	Назва поля	Атрибут	Тип
1	id_status	Унікальний ідентифікатор	int(11)
2	status	Статус завдання	varchar(20)

Таблиця 2.6 – Події завдання системи.

№	Назва поля	Атрибут	Тип
1	id_podii	Унікальний ідентифікатор	int(11)
2	id_user	Автор події	int(11)

3	id_zayavku	Завдання	int(11)

Продовження табл 2.6

4	nazva_podii	Назва подіїї	varchar(255)
5	opus_podii	Опис події	varchar(255)
6	data_stvorenya	Дата створення подіїї	date

Таблиця 2.7 – Пристрої системи.

№	Назва поля	Атрибут	Тип
1	id_prustrii	Унікальний ідентифікатор	int(11)
2	nazva_prustoii	Назва пристрою	varchar(255)
3	id_typ_pristroyu	Тип пристрою	int(11)
4	id_kab	Розміщення пристрою	int(11)
5	data_dobavlenya	Дата додати в систему	date
6	MVO	МВО	varchar(255)
7	inv	Інвентарний номер	int(20)

Таблиця 2.8 – Тип пристрою системи.

№	Назва поля	Атрибут	Тип
1	id_typ_pristroyu	Унікальний ідентифікатор	int(11)
2	typ	Тип пристрою	int(11)

Таблиця 2.9 – Приміщення

№	Назва поля	Атрибут	Тип
1	id_kab	Унікальний ідентифікатор	int(11)
2	nomer	Номер кабінету	varchar(10)

3	korpus	Номер корпусу	varchar(10)

Таблиця 2.10 – Матеріали для виконання завдань

№	Назва поля	Атрибут	Тип
1	id_materialy	Унікальний ідентифікатор	int(11)
2	nazva_materialy	Назва матеріалу	varchar(255)
3	kilkistb	Кількість	int(11)
4	vumir	Одиниця виміру	varchar(10)
5	data_dobavlenya	Дата додати в систему	date

Таблиця 2.11 – Топологія мережі

№	Назва поля	Атрибут	Тип
1	id_topolygii	Унікальний ідентифікатор	int(11)
2	id_master	Пристрій верхнього рівня	int(11)
3	id_slayer	Пристрій нижнього рівня	int(11)
4	kabelbna_linia	Кабель лінія торології	int(11)

Розроблена база даних містить 6 таблиць зі своїми сутностями та зв’язками.

Між таблицями присутні такі зв’язки:

- status-zayavka: зв’язок «один до багатьох»;
- zayavka-podii: зв’язок «один до багатьох»;
- kab-pristroii: зв’язок «один до багатьох»;
- posada-user: зв’язок «один до багатьох»;
- typ_prystroyu-pristroii: зв’язок «один до багатьох»;
- prustroii-topolugii: зв’язок «один до багатьох»;
- user-podii: зв’язок «один до багатьох».

У базах даних зв'язок «один до багатьох» означає, що кожен запис в одній таблиці може мати багато відповідних записів в іншій таблиці, але кожен запис в другій таблиці відповідає лише одному запису в першій таблиці. Цей тип зв'язку також називається "зв'язок один-багато".

Приклад зв’язоку «один до багатьох наведено в рисунку 2.5.

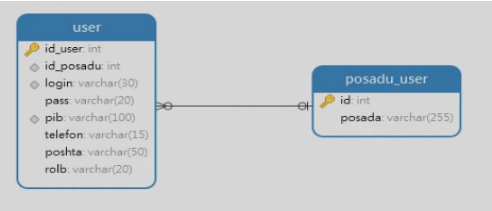


Рис. 2.5. зв’язок «один до багатьох».

Також в системі присутній зв'язок «багато до багатьох». У базах даних зв'язок "багато до багатьох" відображається в ситуаціях, коли кожен запис в одній таблиці може мати декілька відповідних записів в іншій таблиці, і навпаки. Це досягається за допомогою проміжної таблиці, яка встановлює зв'язок між записами двох таблиць.

Для представлення цього зв'язку створюється проміжна таблиця, часто називається "Таблиця зв'язків" або "Таблиця перетинів". Вона містить зв'язки між записами. У цій таблиці можуть бути стовпці, які вказують на ідентифікатори, що встановлюють зв'язок між ними.

Для прикладу розглянемо зв'язок між таблицями Користувачі та завдання. У нас є дві таблиці: "Користувачі" і "завдання". Кожне завдання може мати багатьох виконавців (користувачі системи), і кожен користувач може бути виконавцем в багатьох завданнях. Таким чином, у нас є зв'язок "багато до багатьох" між цими таблицями.

Цей підхід дозволяє нам ефективно управляти зв'язком "багато до багатьох" між даними. Ми можемо легко визначити всіх користувачів, пов'язаних з певним

завданням, або всі завдання, в яких бере участь користувач, шляхом звернення до відповідних записів у проміжній таблиці.

Перелік "Таблиця перетинів" в БД контролю виконання завдань:

- user-zayavka: зв’язок «багато до багатьох»;
- zayavka-pristroii: зв’язок «багато до багатьох»;
- zayavka-materialu: зв’язок «багато до багатьох»;

Приклад зв'язку «багато до багатьох» наведено на рисунку 2.6.

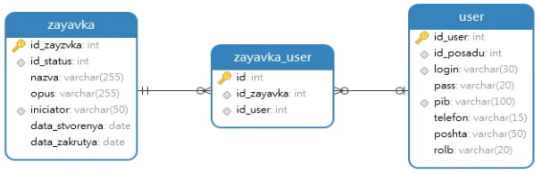


Рис. 2.6. зв’язок «багато до багатьох».

На основі проведеного аналізу сутностей, зв’язків та атрибутів в роботі розроблено проект бази даних, схему якої представлено на рис. 2.7

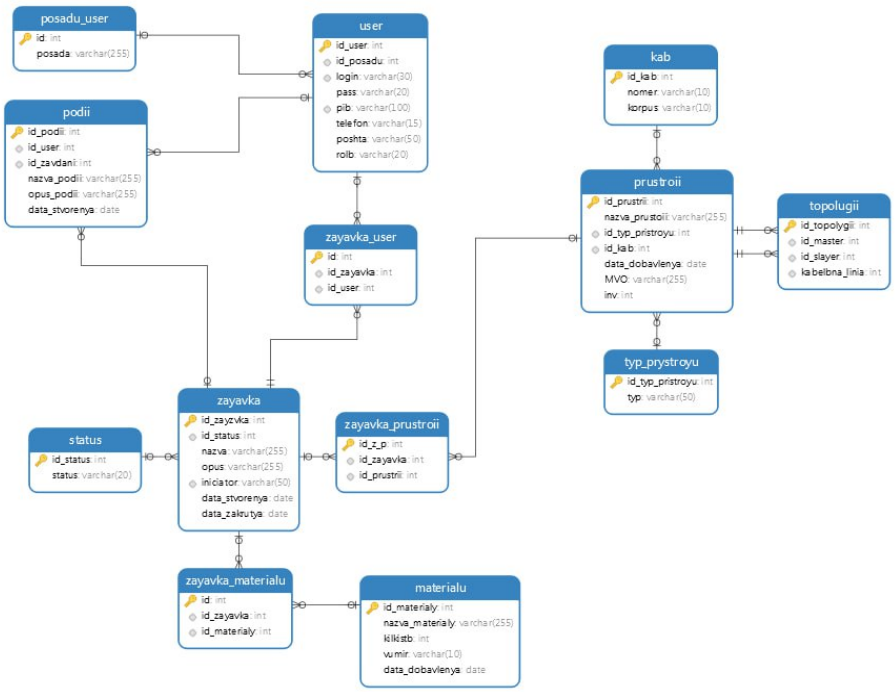


Рис 2.7. Схема бази даних системи контролю виконання завдань.

2.5. Обґрунтування вибору мови програмування та середовище розробки.

У початковій стадії розробки будь-якого програмного забезпечення важливо докласти великих зусиль для вибору мови програмування, одночасно заслуховуючись у виборі засобів для розробки. Це можна зробити, визначивши чіткі завдання та вимоги до написання проекту. При виборі мови програмування необхідно ознайомитися з її особливостями. Популярність мови програмування залежить від функцій та утиліт, які вона надає розробникам. Особливості, які повинна мати мова програмування з метою виділитися, полягають у наступному:

- Мова програмування повинна бути простою, легкою для вивчення та використання, добре читабельною та впізнаваною людиною.

- Абстракція є обов’язковими характеристиками для мови програмування, в якій з’являється здатність визначати складну структуру, а потім і ступінь її зручності.
- Перевага завжди надається портативному мові програмування.
- Ефективність мови програмування повинна бути високою, щоб її можна
- було легко конвертувати в машинний код і виконувати мало місця в пам’яті.
- Мова програмування повинна бути добре структурована та задокументована, щоб вона придатна для розробки додатків.
- Необхідні інструменти для розробки, налагодження, тестування, обслуговування програми повинні бути забезпечені мовою програмування.
- Мова програмування повинна забезпечувати єдине середовище, відоме як інтегроване середовище розробки (IDE).
- Мова програмування повинна бути послідовною з точки зору синтаксису та семантики.

Отож, розглянемо найбільш популярні мови програмування, які використовуються для створення баз даних, а саме: C#, Java і Python - це три популярні мови програмування, які використовуються для розробки різноманітних програм та додатків. Вони мають свої особливості і використовуються у різних сферах. Ось порівняння цих мов:

1. Синтаксис: Кожна з цих мов має свій унікальний синтаксис. C# та Java базуються на синтаксисі C/C++, тому їхня синтаксична структура подібна. Python, з іншого боку, має більш простий і зрозумілий синтаксис, що робить його легшим для вивчення та використання.
2. Виконання: C# та Java є компільованими мовами, що означає, що код перетворюється на машинний код перед виконанням. Python є

інтерпретованою мовою, що означає, що код виконується покроково без попередньої компіляції.

3. Використання: C# широко використовується для розробки програмного забезпечення для платформи Microsoft, зокрема для створення Windows-додатків і ігор. Java використовується для розробки різноманітних програм, включаючи веб-додатки, мобільні додатки та підприємницькі рішення. Python знаходить своє застосування в багатьох галузях, включаючи науку, веб-розробку, обробку даних, штучний інтелект і т. д.
4. Екосистема: Усі три мови мають розгалужену екосистему з великою кількістю бібліотек та фреймворків, які полегшують розробку програмного забезпечення. C# має .NET Framework і ASP.NET для розробки різноманітних додатків. Java має велику кількість фреймворків, таких як Spring і Hibernate, що полегшують створення підприємницьких додатків та веб-серверів.

Таблиця 2.12 – Порівняння мов програмування

Критерії	Python	Java	C#
Простота синтаксису	0	0	1
Об’єктно-орієнтованість	1	1	1
Легкість роботи із базами даних	0	1	1
Інструменти для легкого аналізу даних	1	0	1
Обширність та доступність документації	0	1	1
Різноманітність наявних фреймворків для створення бази даних	1	1	1
Підсумковий результат	3	4	6

За підсумками таблиці 2.12 можна дійти до висновку, що мова програмування C# найкраще підходить для створення системи контролю виконання завдань, оскільки має зручні та необхідні інструментарії, а також має зручний синтаксис.

Важливим аспектом у розробці системи контролю виконання завдань є вибір середовища розробки. Інтегроване середовище розробки (Integrated Development Environment або IDE) — це програмне забезпечення, що надає зручне середовище для розробки, налагодження та тестування програмного забезпечення. IDE об'єднує різні інструменти та функції, які допомагають програмістам писати код більш ефективно.

Visual Studio визнана найкращою IDE для C#. Справа в тому, що обидва продукти належать корпорації Microsoft. Тому вони найкраще підходять для роботи в парі.

Розглянемо основні переваги Visual Studio:

1. Підтримка різних мов програмування.
2. Багатий функціонал, включаючи потужний редактор коду, відлагоджувач, систему керування версіями та інструменти для тестування та профілювання.
3. Інтеграція з екосистемою Microsoft, зокрема з .NET Framework, Azure, SQL Server та іншими технологіями.
4. Велика спільнота розробників та активна підтримка.
5. Кросплатформеність, що дозволяє розробляти на різних платформах.

Ці переваги роблять Visual Studio популярним вибором для багатьох програмістів.

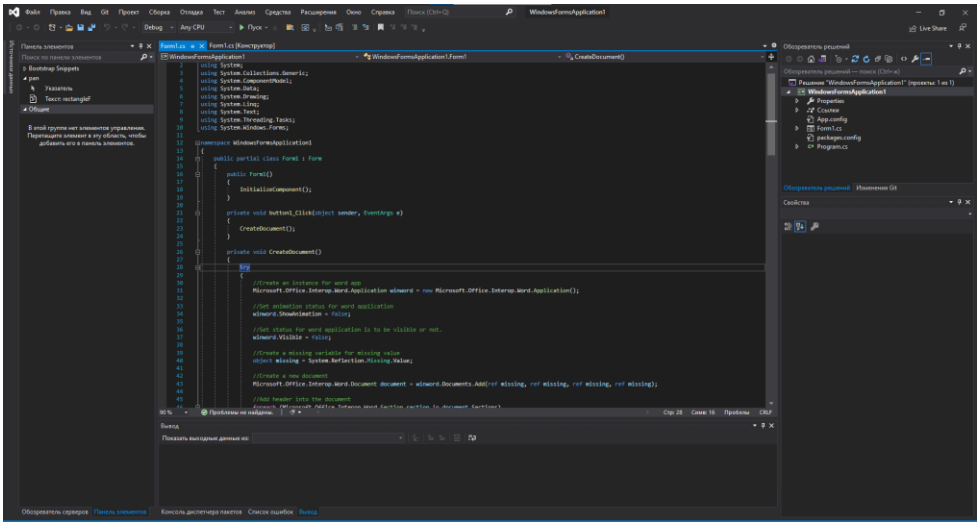


Рис. 2.8. Приклад Інтерфейсу Visual Studio.

2.6. Обґрунтування вибору бази даних та СКБД для реалізації

Для створення БД для системи контролю виконання завдань обрано реляційна база даних (РБД) є типом бази даних, який організований за допомогою реляційної моделі даних.

У реляційній базі даних дані організовані у вигляді таблиць, відомих як реляційні таблиці. Кожна таблиця складається зі стовпців (атрибутів) і рядків (кортежів). Кожен стовпець має ім'я і тип даних, який відображає значення, що зберігається у цьому стовпці. Кожен рядок таблиці представляє запис даних, який містить значення для кожного атрибуту. Одна з головних переваг реляційних баз даних полягає в тому, що вони дозволяють здійснювати складні запити до даних за допомогою мови структурованого запиту (SQL). Запити SQL дозволяють виконувати операції вибірки, вставки, оновлення та видалення даних з бази даних. Реляційні бази даних також підтримують зв'язки між таблицями за допомогою

первинного та зовнішнього ключів, що дозволяє виконувати з'єднання між таблицями для отримання потрібної інформації.

Крім цього, реляційні бази даних забезпечують незалежність даних від програмного забезпечення, що означає, що дані можна зберігати та звертатися до них незалежно від конкретного програмного забезпечення, яке використовується для доступу до бази даних.

Вибір бази даних для розробки системи контролю виконання завдань залежить від різних факторів. Однак, MySQL може бути гарним вибором з таких причин:

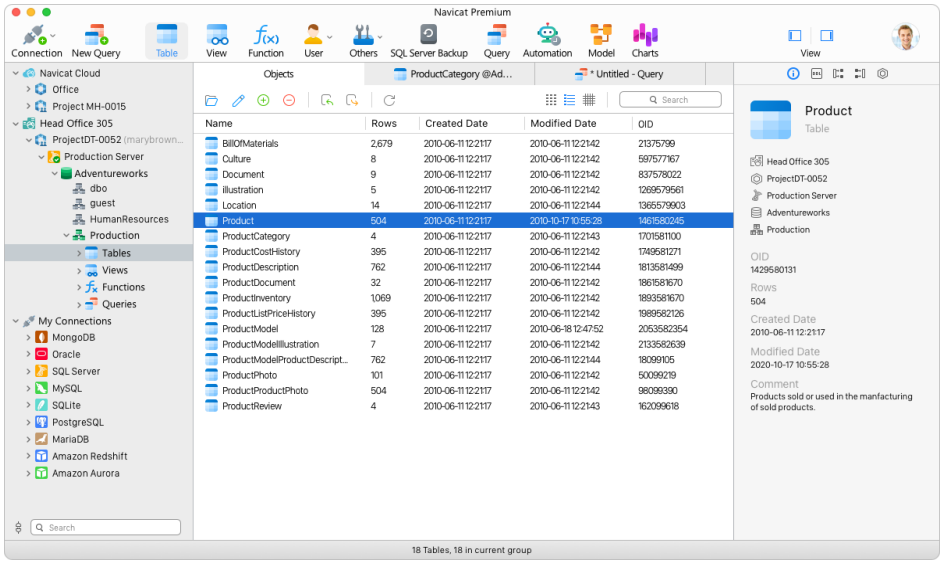
1. Відкритий джерело: MySQL є системою управління базами даних з відкритим кодом (Open Source). Це означає, що ви можете отримати доступ до вихідного коду, а також активну спільноту користувачів, яка надає підтримку, оновлення та вирішення проблем.
2. Надійність та швидкодія: MySQL відомий своєю надійністю та швидкодією. Він може обробляти великий обсяг даних та навантаження, що є важливим для систем контролю виконання завдань зі значним обсягом інформації.
3. Простота використання: MySQL має простий синтаксис мови запитів SQL, що спрощує розробку та підтримку системи контролю виконання завдань. Також існують різноманітні інструменти та інтерфейси для адміністрування та взаємодії з базою даних.
4. Розширюваність: MySQL підтримує розширення, такі як збільшення розміру бази даних, підтримка класу майстер-слуга (master-slave replication) для реплікації даних та кластеризація для забезпечення високої доступності та масштабованості.
5. Широке використання: MySQL є однією з найпопулярніших баз даних у світі. Велика кількість розробників та організацій вже використовують MySQL, що забезпечує наявність багато документації, підручників та інших ресурсів для підтримки та навчання.

Рис. 2.9. логотип MySQL.

Для розробки системи контролю виконання завдань обрано програмний продукт Navicat. Navicat є програмним продуктом, який надає графічний інтерфейс для адміністрування та керування базами даних. Особливості Navicat включають наступні:

1. Мультиплатформеність: Navicat підтримує різні операційні системи, включаючи Windows, mac OS і Linux. Це дозволяє розробникам працювати з базами даних на різних платформах.
2. Підтримка різних типів баз даних: Navicat підтримує широкий спектр баз даних, включаючи MySQL, PostgreSQL, Oracle, SQL Server, SQLite і багато інших. Це дозволяє легко працювати з різними системами управління базами даних з одного інтерфейсу.
3. Графічний інтерфейс: Navicat пропонує інтуїтивно зрозумілий графічний інтерфейс, який спрощує створення, зміну та видалення таблиць, запитів та інших об'єктів бази даних. Інтерфейс дозволяє виконувати завдання швидше і з меншими зусиллями.
4. Редактор запитів: Navicat має вбудований редактор запитів, який надає можливість створювати складні запити до бази даних. Редактор має функції автодоповнення, синтаксичного виділення та інші корисні можливості для полегшення розробки запитів.

5. Управління даними: Navicat дозволяє легко імпортувати, експортувати та синхронізувати дані між різними базами даних. Це дозволяє зручно керувати даними і переміщувати їх між різними середовищами.
6. Розширені можливості безпеки: Navicat надає можливості для налаштування безпеки бази даних, включаючи управління правами доступу користувачів, шифру.



2.10. Інтерфейс Navicat.

РОЗДІЛ 3. Програмна реалізація

3.1. Опис інтерфейсу та функціональних можливостей програмної реалізації.

При вході в додаток користувач системи контролю виконання завдань повинен пройти авторизацію. Процес авторизації користувача в ПЗ зазвичай передбачає додаткові заходи для забезпечення безпеки та обмеження доступу до конфіденційної інформації. Опис процесу авторизації користувача в системі:

1. Введення ідентифікаційних даних: Користувач вводить свої ідентифікаційні дані для отримання доступу до ПЗ. В даній системі це логін та пароль.
2. Перевірка ідентифікаційних даних: ПЗ перевіряє введені ідентифікаційні дані, шляхом перевірки збігу збережених даних у базі даних. Це допомагає перевірити правильність введених даних та підтвердити ідентичність користувача.
3. Аутентифікація: Після перевірки ідентифікаційних даних ПЗ здійснює процес аутентифікації, щоб підтвердити, що користувач є дійсним і має право на доступ. Це може включати перевірку правильності пароля.
4. Авторизація: Після успішної аутентифікації ПЗ визначає, які, дані чи функції доступні для користувача. Це залежить від ролі або привілеїв користувача в системі.

Вікно авторизації в системі контролю виконання завдань наведено в рис. 3.1.

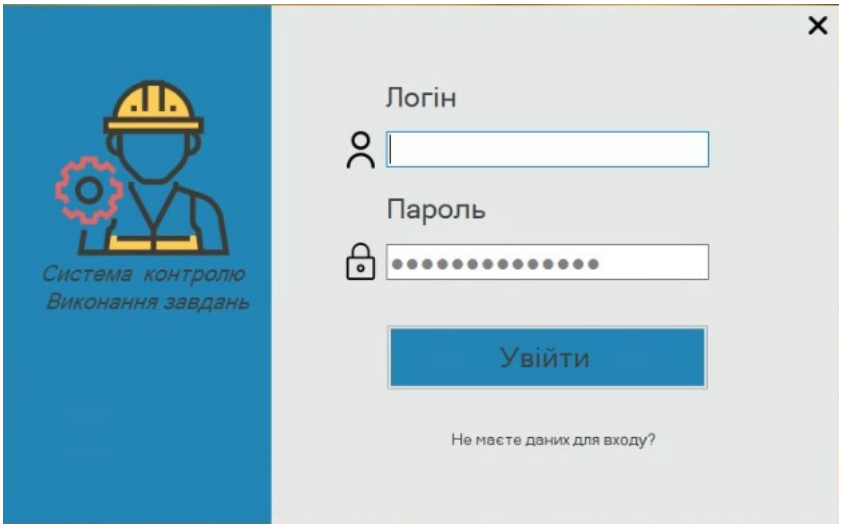


Рис. 3.1. Вікно авторизації користувача.

Якщо в користувача (працівника) не має даних для входу йому потрібно звернутися до керівника або іншого працівника з правами адміністратора для створення облікового запису для входу в систему.

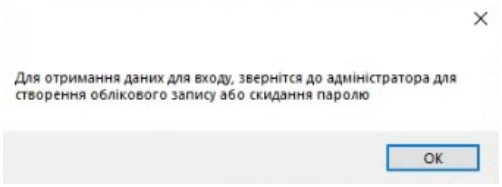


Рис. 3.2. Модальне вікно з інформацією про створення облікового запису.

Вигляд головного вікна ПЗ показано на рисунку 3.3.. На даном вікні розміщені наступні області: головне меню, робоча область, область імені користувача.

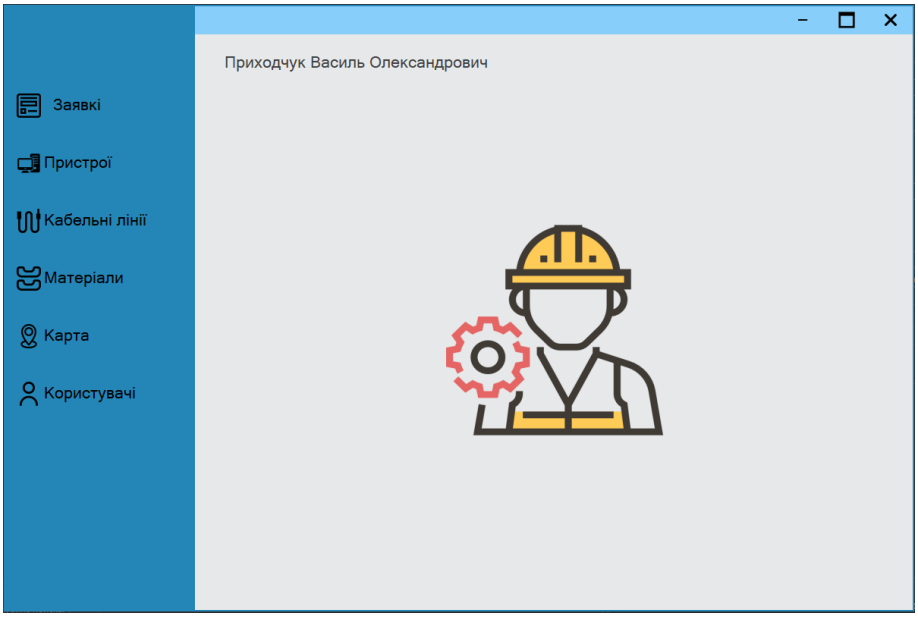


Рис. 3.3. Головне вікно системи контролю та виконання завдань

Меню розташоване у лівій частині вікна і містять список кнопок для відкриття сторінок.

1. Завдання - відкриває сторінку обліку завдань.
2. Пристрої - облік пристроїв системи.
3. Кабельні лінії - облік кабельних ліній мережі інтернет НУВГП.
4. Матеріали - містить доступ до обліку витратних матеріалів для виконання завдань.
5. Карти - вікно карти корпусів НУВГП.
6. Користувачі - дана кнопку доступна лише користувачам з правами доступу адміністратор (admin), та містить доступ до обліку користувачів системи.

При відкриті вікна «обліку завдань» користувач бачать наступні елементи: область переліку завдань, кнопку створення нового завдання, кнопку сформувати звіт та елементи сортування завдань.

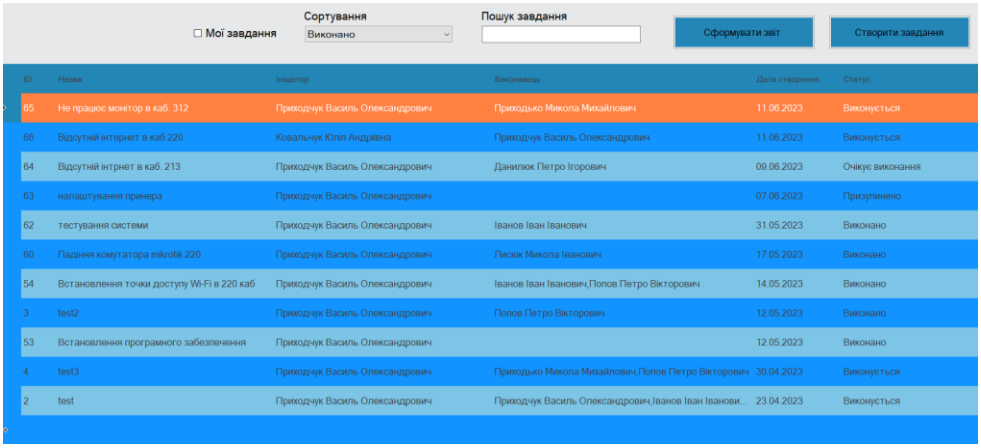


Рис. 3.4. Вікно «облік завдань».

Для створення завдання потрібно натиснути на кнопку «Створення завдання» після чого відкривається вікно «Створення завдання».

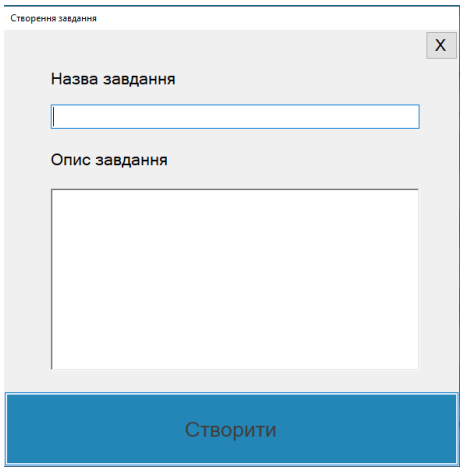


Рис. 3.5. Вікно «створення завдання».

При натисканні кнопки на головному меню пункту «Пристрої» відкриваються вікно обліку пристроїв. В даному вікні є наступні елементи: область списку пристроїв, сортування пристроїв та кнопку «Додати пристрій», пристрої можуть додавати всі користувачі.

Рис. 3.6. Вікно «обліку пристроїв».

При натисненні кнопки перед користувачем відкривається вікно «додавання», в якому користувач вводить інформацію про пристрій та створює новий пристрій в систему.

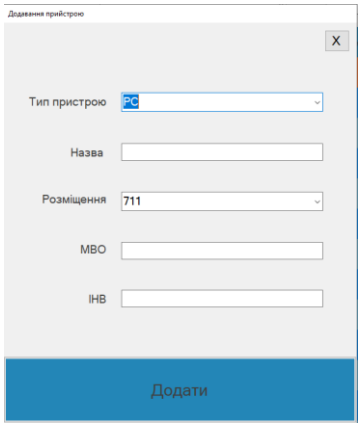


Рис. 3.7. Вікно «додавання пристрою».

При подвійному натисканні на пристрій у вікні «облік пристроїв» відкривається вікно «Картка пристрою», в якій знаходиться вся інформація про пристрій: id в системі, назва, тип пристрою, розміщення, MBO, інвентарний номер. В вкладці «завдання» показано історію завдань, в які було добавлено даний пристрій. Вкладка «топологія» містить в собі дані про те куди цей пристрій підключено та які пристрою підключені до нього.

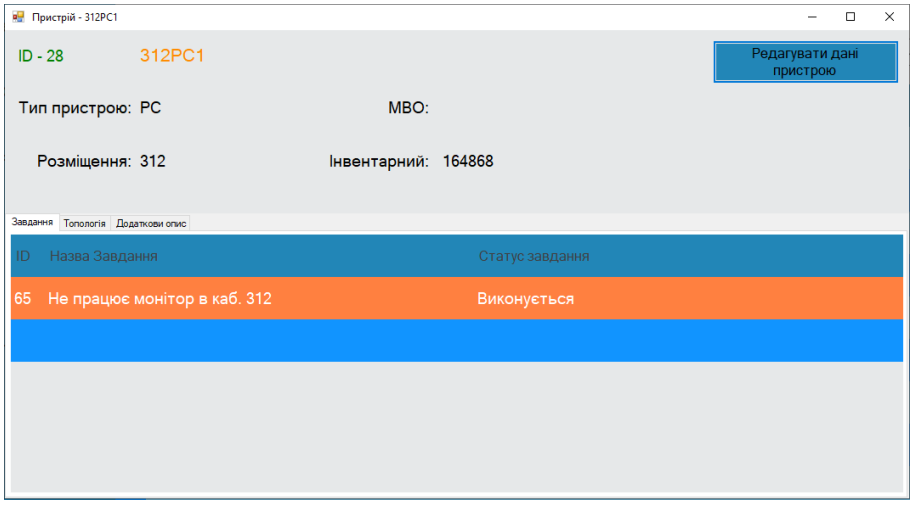


Рис. 3.8. «Картка пристрою».

Для формування топології мережі в системі контролю виконання завдань розроблено вікно, в якому користувачу (працівнику) виведено зв'язки між пристроями (кабельні лінії) в мережі. Тобто, користувач має змогу формувати топологію мережі, створювати нові зв'язки та редагувати вже існуючі зв'язки.

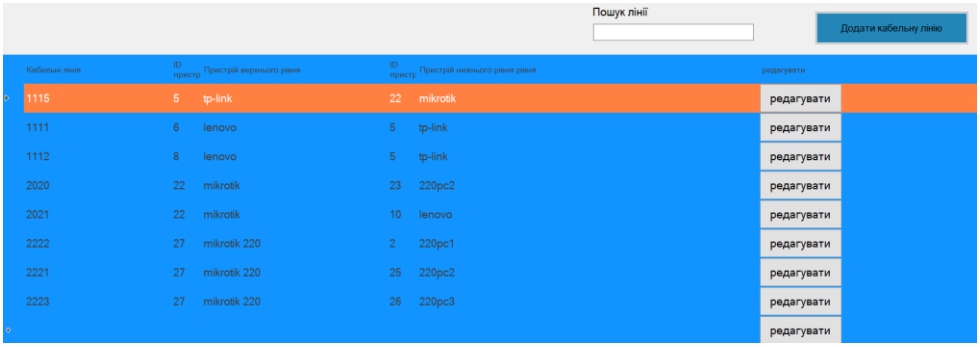


Рис. 3.9. Вікно «кабельні лінії між пристроями».

Для створення нової кабельної лінії користувачу потрібно натиснути на кнопку «Додати кабельну лінію», після чого відкривається вікно, в якому потрібно ввести номер кабельної лінії та вибрати пристрої.

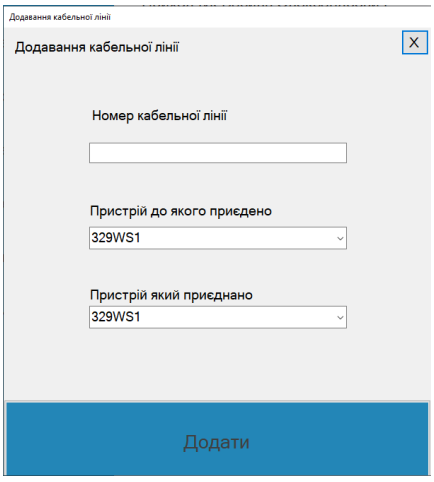


Рис. 3.10. Вікно «додавання кабельної лінії».

Натиснувши на пункт головного меню «Карта» для користувача відкривається вікно з картою корпусів НУВГП для зручної орієнтацій в приміщеннях.

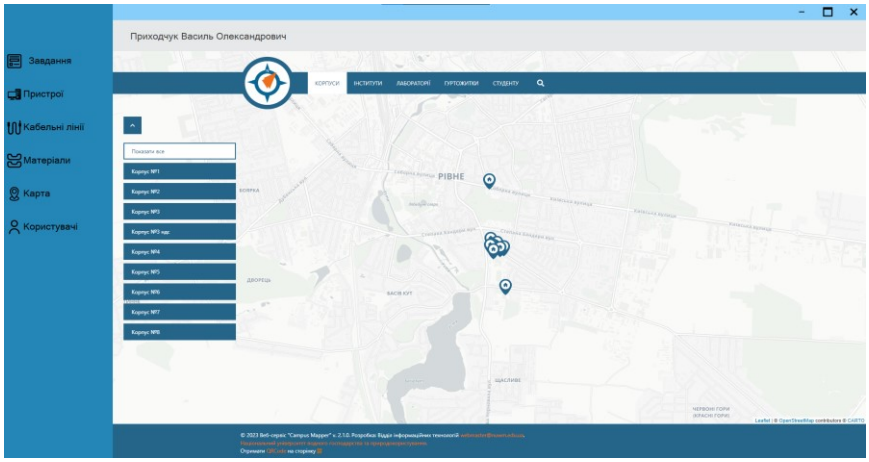
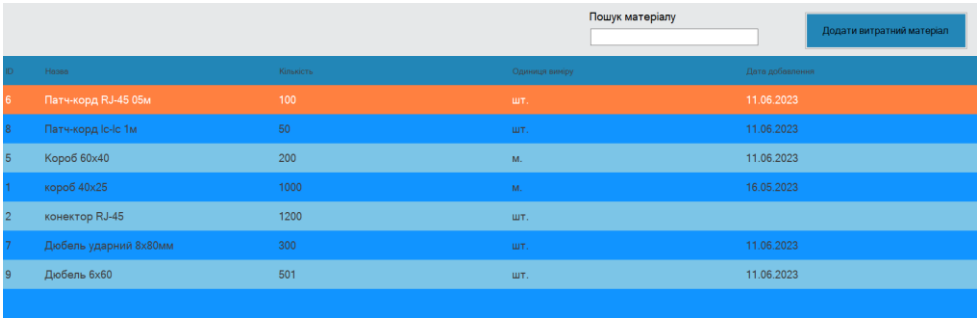


Рис. 3.11. Вікно «карта корпусів НУВГП».

Для обліку матеріалів, які будуть використані в ході виконання завдань, користувачу потрібно відкрити вікно «Матеріали», в якому наведено перелік матеріалів, також можна побачити залишок матеріалів.



ID	Назва	Кількість	Одиниця виміру	Дата добавлення
6	Патч-корд RJ-45 05м	100	шт.	11.06.2023
8	Патч-корд lc-lc 1м	50	шт.	11.06.2023
5	Короб 60х40	200	м.	11.06.2023
1	короб 40х25	1000	м.	16.05.2023
2	конектор RJ-45	1200	шт.	
7	Дюбель ударний 8х80мм	300	шт.	11.06.2023
9	Дюбель 6х60	501	шт.	11.06.2023

Рис. 3.12. Вікно «обліку витратних матеріалів».

Додати витратний матеріал можуть користувачі з правами доступу адміністратор та фахівець з обліку. Щоб додати матеріал потрібно натиснути на кнопку «Додати витратний матеріал».

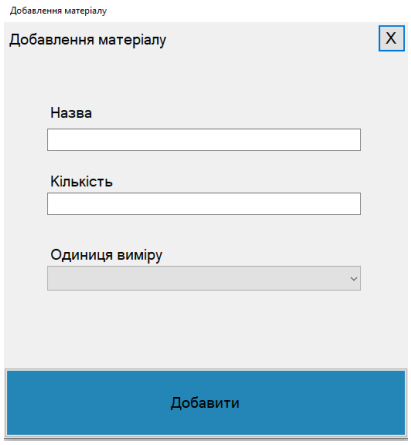


Рис. 3.13. Вікно «додавання витратного матеріалу».

При подвійному натисканні на матеріал в списку матеріалів користувачу відкривається «Картка матеріалу», в даному вікні знаходиться вся інформація про витратний матеріал: назва, кількість, одиниця виміру та дата додати. Також в даному вікно можна переглянути де та скільки використано даного матеріалу.

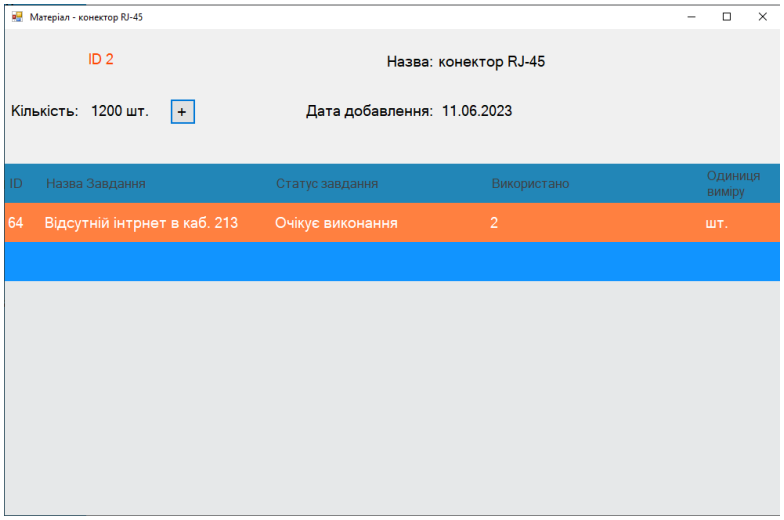


Рис.3.14. Вікно «Картка матеріалу».

Щоб до даного матеріалу добавити певну кількість матеріалу потрібно натиснути на кнопку яка зображена на рис. 3.16.

Рис. 3.15. Кнопка додати матеріалу.

Дана функція доступна користувачам з правами доступу адміністратор та фахівець з обліку.

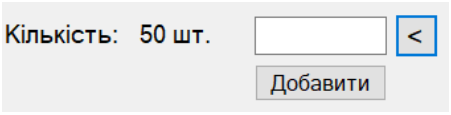


Рис. 3.16. Кнопка додати матеріалу.

При натисненні на кнопку додати матеріали відкривається поле для вводу кількості додати матеріалу.

Рис. 3.17 Вікно списку користувачів.

В кожного користувача є свій обліковий запис для входу в систему, редагувати та створювати ці дані мають право лише користувачі з правами доступу адміністратор (admin). Для створення нового користувача потрібно натиснути на кнопку «Додати» після чого відкривається відповідне вікно.

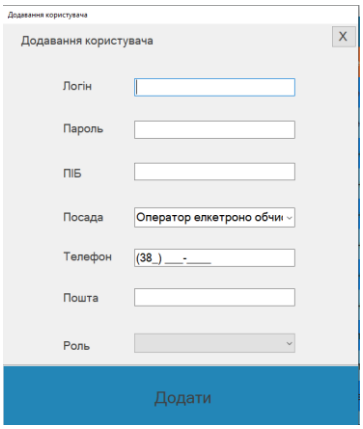


Рис. 3.18. Вікно додавання нового користувача.

Для редагування даних користувача адміністратору необхідно відкрити вікно редагування шляхом подвійного натискання на запис цього користувача у відповідному списку після чого відривається вікно редагування.

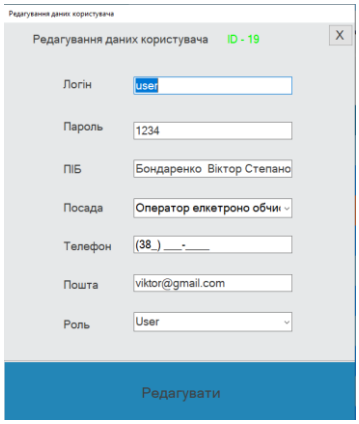


Рис. 3.19. Вікно редагування даних користувача.

3.2. Облік завдань та розподіл їх за виконавцями

Після створення завдання відкривається вікно з створеним завданням, також воно отримує статус «Очікує виконання». На даному вікні завдання розміщено наступний функціонал:

- події завдання - для реєстрація виконаних етапів завдання, тобто формування ходу виконання завдань.
- Пристрої - для додати пристрої до завдання, виконує формування історії завдань по пристрою.
- Учасники - користувачі (працівники) які приймають участь в виконання завдання.
- Використанні матеріали - витратні матеріали які були використані в ході виконання завдань.

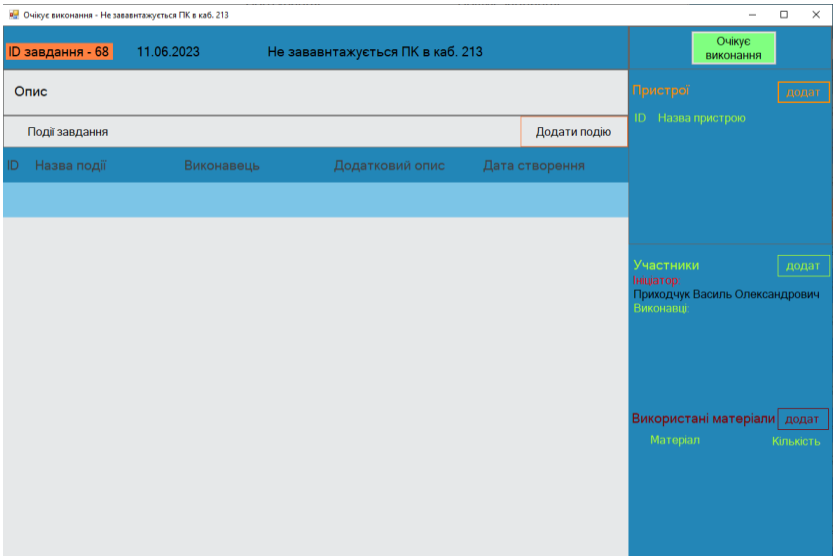


Рис. 3.20. Вікно завдання.

Користувач який створив завдання є «Ініціатором» тобто особою яка є ініціатором завдання, наступним етапом є додавання виконавців які безпосередньо будуть виконувати завдання. Для завершення завдання ініціатору або виконавцям потрібно натиснути на кнопку «Завершити», після чого завдання отримує статус «Виконано».

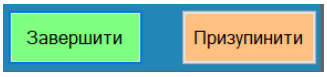


Рис. 3.21. Кнопки завершити та призупинити.

Також завдання можна призупинити, для цього потрібно натиснути на відповідну кнопку «Призупинити».

Якщо при завершенні завдання керівник не задоволений виконанням завдання або в разі не правильного оформлення завдання наприклад: не добавленні

пристрої, не добавленні витратні матеріали і т.п., керівник може повернути завдання у виконання натиснувши на «Призупинити».

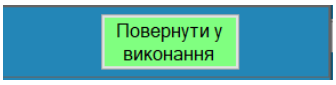


Рис. 3.21. Кнопка повернення у завдання.

3.3. Формування Звіту за певний період.

Для формування звіту виконаних завдань за певний період користувачу потрібно у вікні обліку завдань натиснути на кнопку «Сформувати звіт».

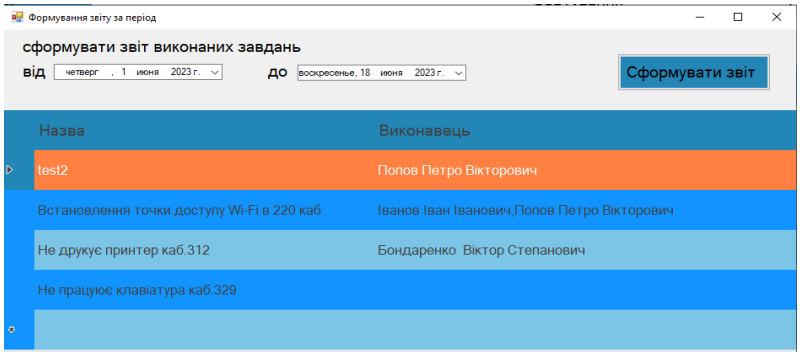


Рис. 3.21. Вікно «формування звітності».

Для формування звіту потрібно вибрати період, після обрання дати в нижній частині вікна формується попередній перегляд, які завдання виконані впродовж вибраного терміну. Для збереження звіту в форматі docx. потрібно натиснути на «Сформувати звіт», далі ввести назву та вибрати місце збереження файлу.

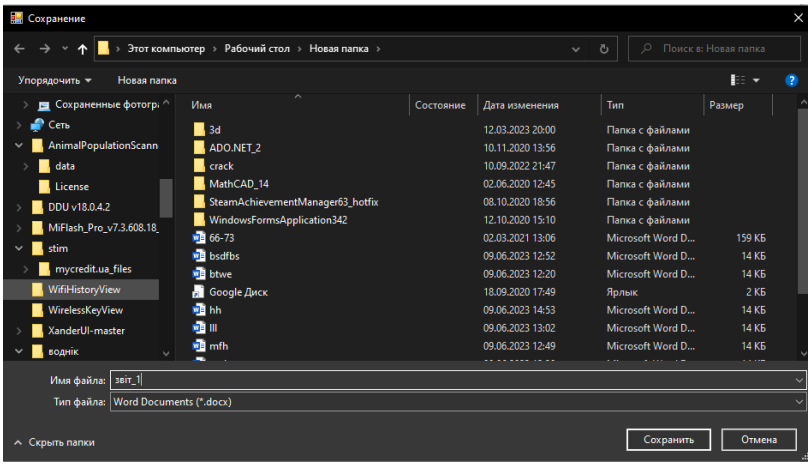


Рис. 3.22. Вікно «збереження звіту».

Приклад оформленого звіту наведено в додатку А.

3.4. Технічні вимоги та рекомендації щодо впровадження автоматизованої системи.

Технічні вимоги для додатку, створеного на базі WinForms, включають наступні аспекти:

1. Операційна система: Windows 7 або новіша версія (Windows 8, Windows 10 тощо).
2. Процесор: Мінімум 1-ядерний процесор з тактовою частотою не менше 1 ГГц. Рекомендовано більш потужний процесор для кращої продуктивності.
3. Оперативна пам'ять (RAM): Мінімум 2 ГБ оперативної пам'яті. Рекомендовано 4 ГБ або більше для покращеної продуктивності та обробки великих обсягів даних.
4. Відеокарта: Підтримка DirectX 9 або новішої версії. Рекомендовано використовувати відеокарту з підтримкою апаратного прискорення 3D-графіки для кращої візуалізації.

5. Місце на жорсткому диску: Мінімум 100 МБ вільного місця на жорсткому диску для встановлення додатку. Розмір може збільшуватись залежно від обсягу даних, що обробляються.
6. Монітор: Роздільна здатність екрану 1024x768 або вища рекомендується для належного відображення інтерфейсу додатку.
7. .NET Framework: Додаток, створений на базі WinForms, вимагатиме встановлення відповідної версії .NET Framework (наприклад, .NET Framework 4.7.2 або новіша).
8. Стабільне Інтернет-з'єднання зі швидкістю не менше 10 мб в секунду.

Нижче наведені загальні системні вимоги для сервера бази даних:

1. Операційна система: MySQL може підтримувати різні операційні системи, включаючи Windows, Linux або macOS.
2. Процесор: Рекомендовано використовувати мінімум 64-бітний процесор з підтримкою мультитрединг і високою швидкодією. Більш потужний процесор з більшою кількістю ядер буде корисним для обробки більшого обсягу даних та високої завантаженості.
3. Оперативна пам'ять (RAM): Мінімум 4 ГБ оперативної пам'яті рекомендується для невеликих баз даних. Однак для великих баз даних і/або підвищеної продуктивності може знадобитись значно більша кількість оперативної пам'яті, наприклад, 8 ГБ, 16 ГБ або навіть більше.
4. Місце на жорсткому диску: Розмір дискового простору буде залежати від розміру бази даних і очікуваного обсягу даних. Врахуйте, що бази даних з часом можуть зростати, тому слід виділити достатньо місця на жорсткому диску для забезпечення масштабованості. Рекомендовано мати достатньо місця для зберігання резервних копій та журналів транзакцій.
5. Жорсткий диск (тип): Рекомендовано використовувати швидкий жорсткий диск або SSD для поліпшення продуктивності бази даних. RAID-масиви або

інші методи забезпечення високої доступності та надійності також можуть бути розглянуті.

6. Мережеве з'єднання: Надайте стабільне мережеве з'єднання для доступу до бази даних з клієнтських додатків. Рекомендується використовувати гігабітну мережу для високої швидкодії і масштабованості.

ВИСНОВКИ

У даній дипломній роботі була розроблена автоматизована система контролю виконання завдань за допомогою технологій WinForms і MySQL. Проект має на меті автоматизувати процес контролю виконання завдань інформаційно-обчислювального центру ЗВО.

Було проаналізовано стан даної проблеми на сучасному етапі. Розглянуто основні аналоги та існуючої системи планування та контролю виконання завдань, визначено їхні особливості та недоліки та розроблено основні вимоги для власної ІС. Розроблено функціональні вимоги до ІС автоматизованого контролю виконання завдань, блок схему потоків даних, спроектовано та розроблено базу даних.

Під час розробки системи були використані можливості WinForms для створення графічного інтерфейсу користувача. Це дозволило створити зручний та інтуїтивно зрозумілий інтерфейс, що сприяє ефективному взаємодії користувачів з системою.

У результаті аналізу обрано мову програмування C#, вона дозволяє розробляти програми для різних операційних систем. Вона має широкі можливості інтеграції з платформою .NET, а також має велику спільноту розробників та ресурсів підтримки.

Однією з ключових складових розробленої системи є база даних, система управління якою реалізовано в MySQL. Таки вибір обґрунтовано тим, що дана система забезпечує зберігання інформації про завдання, користувачів, стан виконання завдань та іншу необхідну інформацію. Дана БД дозволяє зберігати дані в структурованому форматі та забезпечує швидкий доступ до них.

Завдяки використанню технологій WinForms і MySQL, розроблена система є гнучкою та розширювальною. Вона може бути адаптована до потреб конкретної організації або команди, шляхом внесення змін та налаштувань.

Система контролю виконання завдань, розроблена в рамках цієї роботи, демонструє високу ефективність та надійність. Вона дозволяє керувати завданнями, встановлювати, відстежувати прогрес та оцінювати результати. Крім того, система підтримує можливість розподілу завдань між користувачами та забезпечує комунікацію між ними.

У результаті проведених досліджень та розробки, була створена функціональна та ефективна система контролю виконання завдань, яка може бути успішно використано в різних сферах діяльності. Розроблена система дозволяє покращити організацію робочих процесів, збільшити продуктивність та забезпечити більш точне виконання завдань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архітектура та проектування програмного забезпечення URL: https://learn.ztu.edu.ua/mod/book/view.php?id=278.

2. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура правила оформлення / Нац. стандарт України. – Вид. офіц. – [Чинний від 1995–02–23]. – Київ : Держстандарт України, 1995. – 39 с.

3. Репозиторії та інформаційні системи URL: http://ep3.nuwm.edu.ua/9129/.

4. Рихтер Д. CLR via C# Программирование на платформе Microsoft .NET Framework 2.0 на языке C# / Джеффри Рихтер. – Киев: Питер, 2008. – 656 с. – (2).

5. Троелсен Э. Язык программирования C# и платформа .NET 4.5 / Эндрю Троелсен. – Киев: вильямс, 2014. – 1312 с. – (6).

6. Шарп Д. Microsoft Visual C#. Подробное руководство / Дж. Шарп. – СПб: Питер, 2017. – 848 с. – (8-е издание).

7. Архітектура та проектування програмного забезпечення URL: https://learn.ztu.edu.ua/mod/book/view.php?id=278.

8. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура правила оформлення / Нац. стандарт України. – Вид. офіц. – [Чинний від 1995–02–23]. – Київ : Держстандарт України, 1995. – 39 с.

9. Орлов С. Технологии разработки программного обеспечения: Учебник. – СПб.: Питер, 2002. – 464 с.

10. Платформа Microsoft . NET Framework [Електронний ресурс]. – Режим доступу: http://ukrefs.com.ua/169804-Platforma-Microsoft-NET-Framework.html

11.Microsoft Visual Studio URL: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022.

12. Документація C#. [Інтернет ресурс]: https://docs.microsoft.com/.

13. Э. Троелсен. C# и платф орма .NET. Библиотека программиста. – СПб. : Питер, 2007.

14. Основы проектирования реляционных БД [Електронний ресурс]. - Режим

доступу: http://citforum.ru/database/sqllg/index.shtml - Заголовок з екрану.

15.MySQL URL: https://dev.mysql.com/doc/.

16.Jon Skeet. C# in Depth. Second Edition / Jon Skeet, - NY : Manning Publications, 2008. - 554 с.

17. Сайт о программировании. C# / .NET / Entity Framework Core/ Е.Попов – URL: https://metanit.com/sharp/entityframeworkcore/.

18. Microsoft [Інтернет ресурс]: https://www.microsoft.com/.

ДОДАТКИ

ДОДАТОК А

ПРИКЛАД ОФОРМЛЕНОГО ЗВІТУ ЗА ПЕРІОД

Звіт виконаних робіт Інформаційно обчилювального центру

з 2023-06-01 по 2023-06-18

Назва	Виконавці
Тестування мережі	Попов Петро Вікторович
Встановлення точки доступу Wi-Fi в 220 каб	Іванов Іван Іванович,Попов Петро Вікторович
Не друкує принтер каб.312	Бондаренко Віктор Степанович
Не працює клавіатура каб.329	Попов Петро Вікторович

ДОДАТОК Б
ЛІСТИНГ ПРОГРАМНИХ МОДУЛІВ СИСТЕМИ

```
namespace zayavki
{
   public partial class login : Form
   {
      public main form;

      public login()
      {
         InitializeComponent();
         textlogin.Text = "Введіть логін";
         textpassword.Text = "Введіть пароль";
         textlogin.ForeColor = Color.Gray;
         textpassword.ForeColor = Color.Gray;
      }
      private void labelhelp_Click(object sender, EventArgs e)
      {
         MessageBox.Show("Для отримання даних для входу, зверніться до адміністратора для створення облікового запису або скидання паролю");
      }
      private void button1_Click(object sender, EventArgs e)
      {
         if (textlogin.Text != "Введіть логін" && textpassword.Text != "Введіть пароль")
          {
            string loginUser = textlogin.Text;
            string passUser = textpassword.Text;
```

```
            db db = new db();
            DataTable table = new DataTable();
            MySqlDataAdapter adapter = new MySqlDataAdapter();
            MySqlCommand command = new
MySqlCommand("SELECT`user`.login,`user`.pass, `user`.pib, `user`.rolb, `user`.id_user
" +
                "FROM `user` WHERE `user`.login = @ul AND`user`.pass = @uP",
db.getConnection());
            command.Parameters.Add("@ul", MySqlDbType.VarChar).Value =
loginUser;
            command.Parameters.Add("@uP", MySqlDbType.VarChar).Value =
passUser;
            db.openConection();
            adapter.SelectCommand = command;
            adapter.Fill(table);
            if (table.Rows.Count > 0)
            {
               string login = table.Rows[0][2].ToString();
               string roll = table.Rows[0][3].ToString();
               MessageBox.Show("Авторизація пройшла успішно");
               data_trans.login = login;
               data_trans.roll = roll;
               data_trans.id_user = table.Rows[0][4].ToString();
               this.Hide();
               main newForm = new main();
               newForm.Show();
            }
            else
```

```
        {
            MessageBox.Show("Користувача з вказаними даними не існує");
        }
        db.closeConection();
    }
}

private void textlogin_Enter(object sender, EventArgs e)
{
    if (textlogin.Text == "Введіть логін")
    {
        textlogin.Text = "";
        textlogin.ForeColor = Color.FromArgb(64, 64, 64);
    }
}

private void textpassword_Enter(object sender, EventArgs e)
{
    if (textpassword.Text == "Введіть пароль")
    {
        textpassword.Text = "";
        textpassword.ForeColor = Color.FromArgb(64, 64, 64);
    }
}

private void textlogin_Leave(object sender, EventArgs e)
{
    if (textlogin.Text == "")
```

```
        {
            textlogin.Text = "Введіть логін";
            textlogin.ForeColor = Color.Gray;
        }
    }

    private void textpassword_Leave(object sender, EventArgs e)
    {
        if (textpassword.Text == "")
        {
            textpassword.Text = "Введіть пароль";
            textpassword.ForeColor = Color.Gray;
        }
    }

    private void panel1_MouseDown(object sender, MouseEventArgs e)
    {
        ReleaseCapture();
        SendMessage(this.Handle, 0x112, 0xf012, 0);
    }

    [DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
    private extern static void ReleaseCapture();
    [DllImport("user32.DLL", EntryPoint = "SendMessage")]
    private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int wParam,
int lParam);
```

```
#region button close
private void pictureBox2_Click(object sender, EventArgs e)
{
   Application.Exit();
}

private void pictureBox2_MouseHover(object sender, EventArgs e)
{
   pictureBox_close2.BackColor = Color.Red;
}

private void pictureBox2_MouseLeave(object sender, EventArgs e)
{
   pictureBox_close2.BackColor = Color.FromArgb(230, 232, 233);
}

private void pictureBox2_MouseDown(object sender, MouseEventArgs e)
{
   pictureBox_close2.BackColor = Color.FromArgb(192, 0, 0);
}
#endregion

#region labalhelp
private void labelhelp_MouseHover(object sender, EventArgs e)
{
   labelhelp.ForeColor = Color.Red;
}
```

```
        private void labelhelp_MouseLeave_1(object sender, EventArgs e)
        {
            labelhelp.ForeColor = Color.FromArgb(64, 64, 64);
        }
        #endregion
    }
}

namespace Zayavki
{
    public partial class main : Form
    {
        private int borderSize = 2;

        public main()
        {
            InitializeComponent();
            this.Padding = new Padding(borderSize);
            this.BackColor =  Color.FromArgb(35,134, 183);

            btnUser.Visible = false;
            label1.Text = data_trans.login;
            if(data_trans.roll == "admin")
            {
                btnUser.Visible = true;
            }
        }
```

```
private void showSubMenu(Panel subMenu)
{
   if (subMenu.Visible == false)
   {

      subMenu.Visible = true;
   }
   else
      subMenu.Visible = false;
}

private void btnMedia_Click(object sender, EventArgs e)
{
   openChildForm(new zayavki());

}
private void btnTools_Click(object sender, EventArgs e)
{
   openChildForm(new materialu());
}
private void btnEqualizer_Click(object sender, EventArgs e)
{
   openChildForm(new topologii());
}
private void btnHelp_Click(object sender, EventArgs e)
{
   openChildForm(new maps());
}
```

```
private void btnExit_Click(object sender, EventArgs e)
{
   Application.Exit();
}
private Form activeForm = null;
private void openChildForm(Form childForm)
{
   if (activeForm != null) activeForm.Close();
   activeForm = childForm;
   childForm.TopLevel = false;
   childForm.FormBorderStyle = FormBorderStyle.None;
   childForm.Dock = DockStyle.Fill;
   panelChildForm.Controls.Add(childForm);
   panelChildForm.Tag = childForm;
   childForm.BringToFront();
   childForm.Show();
}
protected override void WndProc(ref Message m)
{
   const int WM_NCCALCSIZE = 0x0083;
   if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
   {
      return;
   }
   base.WndProc(ref m);
}
[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
private extern static void ReleaseCapture();
```

```
    [DllImport("user32.DLL", EntryPoint = "SendMessage")]
    private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int wParam,
int lParam);

    private void panelbard_MouseDown(object sender, MouseEventArgs e)
    {
       ReleaseCapture();
       SendMessage(this.Handle, 0x112, 0xf012, 0);
    }
    #region button close min max
    private void pictureBox_close2_Click(object sender, EventArgs e)
    {
       Application.Exit();
    }
    private void pictureBox_close2_MouseHover(object sender, EventArgs e)
    {
       pictureBox_close2.BackColor = Color.Red;
    }
    private void pictureBox_close2_MouseLeave(object sender, EventArgs e)
    {
       pictureBox_close2.BackColor = Color.LightSkyBlue;
    }
    private void pictureBox_close2_MouseDown(object sender, MouseEventArgs e)
    {
       pictureBox_close2.BackColor = Color.FromArgb(192, 0, 0);
    }
    private void pictureBox2_Click(object sender, EventArgs e)
    {
```

```
    this.WindowState = FormWindowState.Minimized;
}
private void pictureBox1_Click(object sender, EventArgs e)
{
    if (this.WindowState == FormWindowState.Normal)
    {this.WindowState = FormWindowState.Maximized; }
    else
    {this.WindowState = FormWindowState.Normal; }
}
private void pictureBox1_MouseHover(object sender, EventArgs e)
{
    pictureBox1.BackColor = Color.Gray;
}
private void pictureBox1_MouseLeave(object sender, EventArgs e)
{
    pictureBox1.BackColor = Color.LightSkyBlue;
}
private void bntminimiza_MouseHover(object sender, EventArgs e)
{
    bntminimiza.BackColor = Color.Gray;
}
private void bntminimiza_MouseLeave(object sender, EventArgs e)
{
    bntminimiza.BackColor = Color.LightSkyBlue;
}
#endregion
private void btnUser_Click(object sender, EventArgs e)
{
```

```
            openChildForm(new user());
        }
        private void main_Resize(object sender, EventArgs e)
        {
            switch (this.WindowState)
            {
                case FormWindowState.Maximized:
                    this.Padding = new Padding(0, 8, 8, 0);
                    break;
                case FormWindowState.Normal:
                    if (this.Padding.Top != borderSize)
                        this.Padding = new Padding(borderSize);
                    break;
            }
        }
        private void btnPrustroii_Click(object sender, EventArgs e)
        {
            openChildForm(new pristroii());
        }
        private void button1_Click(object sender, EventArgs e)
        {
            openChildForm(new dashboard());
        }
    }
}
public partial class add_zavdanya : Form
    {
        public add_zavdanya()
```

```
{
    InitializeComponent();
}
private void Close_Click(object sender, EventArgs e)
{
    Close();
}
private void button1_Click(object sender, EventArgs e)
{
    string textLogin = textBox1.Text;
    string textPass = richTextBox1.Text;
    string id_zav = "error";
    data_trans.nazva_zavdanya = textLogin;
    db db = new db();
    db.openConection();
    try
    {
        string data_p;
        DateTime date1 = dateTimePicker1.Value;
        data_p = date1.ToString("yyyy-MM-dd");

        DataTable table = new DataTable();
        DataTable table2 = new DataTable();
        DataTable table3 = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        MySqlCommand command = new MySqlCommand("INSERT INTO " +
            "`custema_zayavok`.`zayavka` " +
            "(`nazva`, " +
```

```
            "`opus`," +
            "`data_stvorenya`," +
            "iniciator, " +
            "`id_status`) " +
            "VALUES (@unazva, @uopus, @uData , @uIniciator, 4)",
db.getConnection());
        command.Parameters.Add("@uData", MySqlDbType.VarChar).Value =
data_p;
        command.Parameters.Add("@unazva", MySqlDbType.VarChar).Value =
textLogin;
        command.Parameters.Add("@uopus", MySqlDbType.VarChar).Value =
textPass;
        command.Parameters.Add("@uIniciator", MySqlDbType.VarChar).Value =
data_trans.login;

        adapter.SelectCommand = command;
        adapter.Fill(table);
        db.closeConection();
        db.openConection();

        MySqlDataAdapter adapter2 = new MySqlDataAdapter();
        MySqlCommand command2 = new MySqlCommand("SELECT
zayavka.id_zayzvka FROM zayavka WHERE zayavka.nazva = @unazva",
db.getConnection());
        command2.Parameters.Add("@unazva", MySqlDbType.VarChar).Value =
textBox1.Text;
        adapter2.SelectCommand = command2;
        adapter2.Fill(table2);
```

```
            if (table2.Rows.Count > 0)
            {
               id_zav = table2.Rows[0][0].ToString();
               data_trans.id_zayavka = table2.Rows[0][0].ToString();
            }
            MySqlDataAdapter adapter3 = new MySqlDataAdapter();
            MySqlCommand command3 = new MySqlCommand("INSERT INTO
`custema_zayavok`.`zayavka_user` " +
               "(`id_zayavka`, `id_user`) " +
               "VALUES(@uid_zav, 10)", db.getConnection());
            command3.Parameters.Add("@uid_zav",  MySqlDbType.VarChar).Value =
id_zav;
            adapter3.SelectCommand = command3;
            adapter3.Fill(table3);
            zavdanya newForm = new zavdanya();
            newForm.Show();
            db.closeConection();
            this.Close();
         }
         catch
         {
            MessageBox.Show("Не вдалося створити завдання");
         }
      }
   }
}
namespace Zavdanya
{
```

```
public partial class updata_pristrii : Form
{
    [DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
    private extern static void ReleaseCapture();
    [DllImport("user32.DLL", EntryPoint = "SendMessage")]
    private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int wParam,
int lParam);
    public updata_pristrii()
    {
        InitializeComponent();
        string query1 = @"SELECT
            typ_prystroyu.id_typ_pristroyu,
        typ_prystroyu.typ
        FROM
            typ_prystroyu";// типи пристрою
       string query2 = @"SELECT
            kab.id_kab,
        kab.nomer
        FROM
            kab";// типи розмыщеня
        db db = new db();
        db.openConection();
        using (MySqlDataAdapter da_bud = new MySqlDataAdapter(query1,
db.getConnection()))
        {
            DataTable table = new DataTable();
            table.Locale = System.Globalization.CultureInfo.InvariantCulture;
            da_bud.Fill(table);
```

```
            comboBox_typ.DataSource = table;
            comboBox_typ.DisplayMember = table.Columns["typ"].ColumnName;
            comboBox_typ.ValueMember =
table.Columns["id_typ_pristroyu"].ColumnName;
            comboBox_typ.Text = "";
        }
        using (MySqlDataAdapter da_bud2 = new MySqlDataAdapter(query2,
db.getConnection()))
        {
            DataTable table = new DataTable();
            table.Locale = System.Globalization.CultureInfo.InvariantCulture;
            da_bud2.Fill(table);
            comboBox_kab.DataSource = table;
            comboBox_kab.DisplayMember = table.Columns["nomer"].ColumnName;
            comboBox_kab.ValueMember = table.Columns["id_kab"].ColumnName;
            comboBox_kab.Text = "";
        }
        label1.Text = data_trans.id_pristroii_updata;
        DataTable table2 = new DataTable();
        string query3 = @"SELECT prustroii.id_prustrii, " +
            "typ_prystroyu.typ, prustroii.nazva_prustoii, kab.nomer, " +
            "prustroii.data_dobavlenya, prustroii.MVO, prustroii.inv " +
            "FROM prustroii " +
            "INNER JOIN typ_prystroyu " +
            "ON prustroii.id_typ_pristroyu = typ_prystroyu.id_typ_pristroyu " +
            "INNER JOIN kab " +
            "ON prustroii.id_kab = kab.id_kab " +
            "WHERE prustroii.id_prustrii = '" + data_trans.id_pristroii_updata + "'";
```

```
    MySqlDataAdapter adapter2 = new MySqlDataAdapter();
    MySqlCommand command = new MySqlCommand(query3, db.getConnection());
    adapter2.SelectCommand = command;
    adapter2.Fill(table2);
    if (table2.Rows.Count > 0)
    {
        comboBox_typ.DisplayMember = table2.Rows[0][1].ToString();
        textBox_nazva.Text = table2.Rows[0][2].ToString();
        comboBox_kab.DisplayMember = table2.Rows[0][3].ToString();
        textBox_mvo.Text = table2.Rows[0][5].ToString();
        textBox_inv.Text = table2.Rows[0][6].ToString();
    }
    else
    {  MessageBox.Show("error");}
    db.closeConection();
}
private void Close_Click(object sender, EventArgs e)
{
    this.Close();
}
private void btnUpdata_Click(object sender, EventArgs e)
{
    string data_p;
    DateTime date1 = dateTimePicker1.Value;
    data_p = date1.ToString("yyyy-MM-dd");
    string pri = @"UPDATE `custema_zayavok`.`prustroii`
                SET `nazva_prustoii` = '" + textBox_nazva.Text + "', " +
                "`id_typ_pristroyu` = '" + comboBox_typ.SelectedValue + "', " +
```

```
                "`id_kab` = '" + comboBox_kab.SelectedValue + "', " +
                "`data_dobavlenya` = '" + data_p + "', " +
                "`MVO` = '" + textBox_mvo.Text + "', " +
                "`inv` ='" + textBox_inv.Text + "' " +
                "WHERE prustroii.id_prustrii = '" + data_trans.id_pristroii_updata + "'";
            db db = new db();
            db.openConection();
            try
            {
                DataTable table = new DataTable();
                MySqlDataAdapter adapter = new MySqlDataAdapter();
                MySqlCommand command = new MySqlCommand(pri, db.getConnection());
                adapter.SelectCommand = command;
                adapter.Fill(table);
                db.closeConection();
                MessageBox.Show("Редагування виконано успушно");
                this.Close();
            }
            catch
            {MessageBox.Show("При редагувані виникла помилка");}
        }
        private void panel2_MouseDown(object sender, MouseEventArgs e)
        {
            ReleaseCapture();
            SendMessage(this.Handle, 0x112, 0xf012, 0);
        }
    }
}
```