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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК 004.75

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Інтегровані інформаційні системи»
зі спеціальності 126 «Інформаційні системи та технології»
на тему: «Система автоматизованого керування тестовими серверами»

Виконав:
студент 2 курсу, групи ІА–з21мп
Шило Олександр Сергійович _____

Керівник:
доцент каф. ІСТ, к.т.н., доцент
Коган Алла Вікторівна
_____

Рецензент:
доцент каф. ОТ, к.т.н., доцент
Роковий Олександр Петрович
_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2024р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Шило Олександру Сергійовичу

1. Тема дисертації «Система автоматизованого керування тестовими серверами», науковий керівник дисертації Коган Алла Вікторівна, к.т.н., доцент, затверджені наказом по університету від «03» 11 2023 р. № 5135-с

2. Термін подання студентом дисертації «08» 01 2024 р.

3. Об'єкт дослідження – процес автоматизованої керування тестовими серверами за допомогою програмного забезпечення

4. Предмет дослідження – розробка і реалізація системи керування тестовими серверами, включаючи інтерфейс користувача, механізми комунікації з серверами та функціонал для запуску, зупинення та моніторингу статусу Java процесів на серверах

5. Перелік завдань, які потрібно розробити – аналіз проблеми та існуючих рішень; розробка моделі/методу/алгоритму/програмного забезпечення; дослідження ефективності розробленого методу/алгоритму/програмного забезпечення

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 8 плакатів

7. Орієнтовний перелік публікацій – відсутні.

8. Дата видачі завдання «01» 06 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Підготовчий етап	02.06.2023	
2	Проектування системи керування	25.06.2023	
3	Реалізація програмного забезпечення	22.07.2023	
4	Тестування та оптимізація	14.08.2023	
5	Інтеграція та впровадження	29.08.2023	
6	Аналіз результатів та публікація	07.09.2023	
7	Виконання експериментальних досліджень	22.09.2023	
8	Оформлення пояснювальної записки	11.10.2023	
9	Подання дисертації на попередній захист	19.11.2023	
10	Подання дисертації на захист	08.01.2024	

Студент Олександр ШИЛО

Науковий керівник Алла КОГАН

РЕФЕРАТ

Система автоматизованого керування тестовими серверами, 153 с., 28 табл., 22 рис., 11 дод., 16 джерел.

АВТОМАТИЗАЦІЯ КЕРУВАННЯ, ТЕСТОВІ СЕРВЕРИ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МОНІТОРИНГ, JAVA ПРОЦЕСИ.

Актуальність. Сучасний інформаційний ландшафт вимагає надійної та ефективної автоматизації керування тестовими серверами, оскільки велика кількість серверів та стендів ускладнює процес тестування програмного забезпечення. З ростом об'єму розроблених програмних продуктів зростає і потреба в їхньому якісному тестуванні. Автоматизація керування тестовими серверами дозволить оптимізувати процес тестування та підвищити продуктивність розробки.

Мета магістерської дисертації. Підвищення ефективності тестування програмного забезпечення за рахунок удосконалення автоматизованого керування тестовими серверами.

Об'єкт дослідження. Об'єктом дослідження є процес автоматизованої керування тестовими серверами за допомогою програмного забезпечення.

Предмет дослідження. Предметом дослідження є розробка і реалізація системи керування тестовими серверами, включаючи інтерфейс користувача, механізми комунікації з серверами та функціонал для запуску, зупинення та моніторингу статусу Java процесів на серверах.

Наукова новизна. Дана робота впроваджує новий підхід до автоматизації керування тестовими серверами, використовуючи сучасні технології та програмне забезпечення для оптимізації процесу тестування. Результати дослідження сприятимуть підвищенню продуктивності та ефективності управління тестовими стендами.

ABSTRACT

System of automated management of test servers, 153 p., 28 tables, 22 figures, 11 appendices, 16 sources.

AUTOMATION MANAGEMENT, TEST SERVERS, SOFTWARE, MONITORING, JAVA PROCESSES.

Relevance. The modern information landscape demands reliable and efficient automation of test server management, as the increasing number of servers and test environments complicates the software testing process. With the growth of developed software products, there is a corresponding need for high-quality testing. Automation of test server management will optimize the testing process and enhance development productivity.

Objective of the master's thesis. Increasing the efficiency of software testing by improving the automated management of test servers.

Research Object. The object of the research is the process of automated test server management using software.

Research Subject. The subject of the research is the development and implementation of a system for managing test servers, including the user interface, communication mechanisms with servers, and functionality for launching, stopping, and monitoring the status of Java processes on servers.

Scientific novelty. This work introduces a new approach to automating the management of test servers, using modern technologies and software to optimize the testing process. The results of the research will contribute to increasing the productivity and efficiency of test bench management.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність;

iFOBS – Interactive Front Office Banking System (Інтерактивна Система Фронт-Офісного Обслуговування Клієнтів Банку);

PuTTY – (від англ. TTY — телетайп, букв. англ. putty — мастика) — вільно розповсюджуваний клієнт для протоколів SSH, Telnet, rlogin і чистого TCP. Спочатку розроблявся для Windows, проте пізніше був портований на різні операційні системи. Наявні офіційні порти для UNIXоподібних платформ [4];

CSS – це спеціальна мова стилю сторінок[en], що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних. CSS є основною технологією всесвітньої павутини, поряд із HTML та JavaScript [5];

Java – Java (вимовляється Джава) — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи [6];

CI/CD – CI/CD є поширеною DevOps-практикою. CI (Continuous Integration) — це неперервна інтеграція, а CD (Continuous Delivery) — неперервна доставка. Цей набір методик дозволяє розробникам частіше і надійніше розгортати зміни в програмному забезпеченні[7];

SSH – Secure Shell, SSH (англ. Secure Shell — «безпечна оболонка») — мережевий протокол прикладного рівня, що дозволяє проводити віддалене управління комп'ютером і тунелювання TCP-з'єднань (наприклад, для передачі

файлів). Схожий за функціональністю з протоколом Telnet і rlogin, проте шифрує весь трафік, в тому числі і паролі, що передаються.[8]

Red Hat Enterprise Linux Server – це серверна операційна система, орієнтована на підтримку найбільш критичних корпоративних додатків. Red Hat Enterprise Linux Server надає компаніям чотири головні переваги: високу продуктивність, необмежену масштабованість, високий рівень безпеки і низькі ціни в порівнянні з аналогами [9];

SQLite – це компактна вбудована реляційна база даних з відкритим кодом. Вона одна з найпопулярніших у світі, має нагороду Google-O'Reilly Open Source Awards і широко використовується в додатках та системах, де потрібно організувати зберігання даних [10];

Oracle – об'єктно-реляційна система керування базами даних від Oracle Corporation [11];

Git – розподілена система керування версіями файлів та спільної роботи [12].

ВСТУП

В сучасному інформаційному середовищі, де розробка програмного забезпечення набуває обривистий ріст, проблема ефективного керування тестовими серверами стає критичною для забезпечення високої якості та швидкості тестування. Магістерська дисертація "Система автоматизованого керування тестовими серверами" присвячена вирішенню цієї актуальної проблеми в рамках освітньо-професійної програми "Інтегровані інформаційні системи".

Актуальність предмета дослідження обумовлена необхідністю підтримки та розвитку сучасних технологій та програмних продуктів, що вимагають великої кількості тестування для забезпечення їхньої стабільності та надійності. Зростання обсягів розробки веде до потреби в ефективних засобах управління тестовими серверами.

Мета даного дослідження полягає в підвищенні ефективності тестування програмного забезпечення шляхом оптимізації процесу керування тестовими серверами.

Об'єкт дослідження – це процес автоматизованого керування тестовими серверами, а предмет дослідження –розробка та реалізація системи керування, включаючи інтерфейс користувача, механізми комунікації та функціонал для моніторингу статусу Java процесів на серверах.

Великою науковою новизною даної роботи є впровадження нового підходу до автоматизації керування тестовими серверами, використовуючи сучасні технології та програмне забезпечення для оптимізації тестування.

Особливість дослідження полягає в можливості зупинки та запуску процесів одночасно на значній кількості серверів, використовуючи доступне безкоштовне програмне забезпечення. У порівнянні з ручним способом зупинки процесу на 50 серверах через PuTTY, автоматизоване керування може скоротити час виконання завдань з 50 годин до всього 2 годин.

Ця магістерська дисертація обіцяє внести суттєвий внесок у сферу автоматизації керування тестовими серверами, забезпечуючи ефективність та

продуктивність управління тестовими стендами. Ключові слова, такі як автоматизація керування, тестові сервери, програмне забезпечення, моніторинг та Java процеси, відзначають основні аспекти роботи, що впливають на розвиток цієї області технологій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення проблеми

Сучасний інформаційний ландшафт характеризується постійним ростом кількості тестових серверів та стендів, використовуваних для тестування програмного забезпечення. Ця динаміка призводить до зростання складності та обсягів процесу тестування. У таких умовах виникає необхідність в надійний та ефективний системі автоматизованого керування тестовими серверами.

В 2005 р. відбулася прем'єра системи iFOBS від IT-компанії CS. В свою чергу “CS Ltd” заснована в 1997 р., та припадає на кінця 90-х, це період який тільки почав набирати обертів в сфері інформаційних технологій.

Продуктова та сервісна IT-компанія CS є найбільшим постачальником IT-рішень, високотехнологічного обладнання та послуг для фінансових компаній. Багатий досвід локальних і міжнародних проектів дозволяє CS створювати сучасні системи комплексної автоматизації бізнес-процесів та операцій фінансового ринку, управління персоналом і бізнесом, обліку та формування звітності.

На рисунку 1.1 зображена інформація про досвід та команду “CS Ltd”.

Рисунок 1.1 – Інформація про досвід та команду, з офіційного сайту “CS Ltd”.

Стислий поверхневий опис продуктів та послуг котрі надає “CS Ltd” зображено на рисунку 1.2.

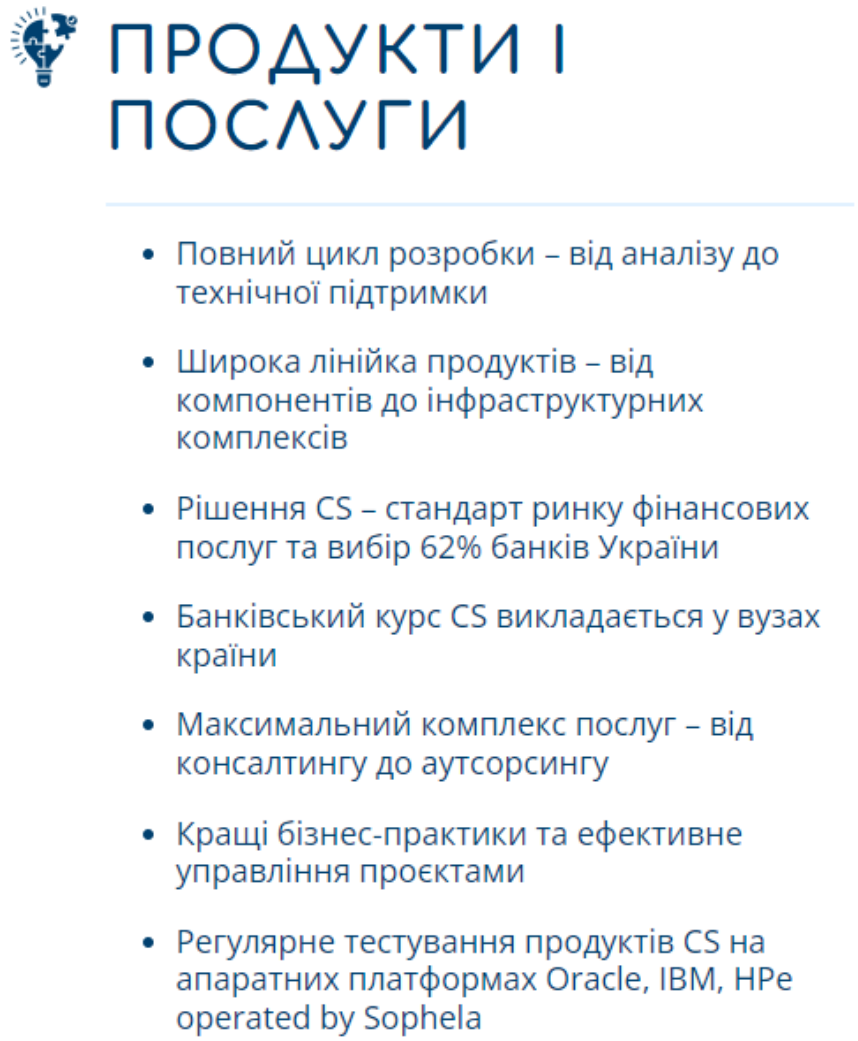


Рисунок 1.2 – Опис продуктів та послуг “CS Ltd”

CS IT COMMUNITY –це велика та впливова група учасників, що об'єднує фінансові організації та компанії в своєму партнерському клубі. З понад 80 клієнтами і включенням учасників з фінансового сектору, ця спільнота створює сильну мережу експертів.

Партнерський клуб відзначається різноманітністю компаній, які входять до складу спільноти, що робить його динамічним та дієвим об'єднанням.

Банківська конференція CS Network виходить за межі традиційної області фінансів і IT. Вона не лише надає платформу для обміну знаннями між фахівцями в цих галузях, але і створює дружбу та сприяє формуванню кола друзів.

Спрямована на обговорення сучасних технологій, конференція підкреслює важливість взаємодії фінансового та IT-середовища. Це місце, де інновації і технології об'єднуються, створюючи важливий форум для розвитку та обміну ідеями[1].

Рисунок 1.3 –CS IT COMMUNITY

Система «клієнт – Інтернет – банк» (iFOBS) – система електронного платіжного документообігу, яка дозволяє підтримувати зв’язок з банком безпосередньо з персонального комп’ютера клієнта, за наявності мережі Internet.

Програмне забезпечення систем дистанційного обслуговування має відповідати вимогам законодавства, в тому числі нормативно-правових актів Національного банку, які пред’являються до технології та захисту електронних банківських розрахунків.

Юридичною підставою для роботи клієнта за допомогою систем дистанційного обслуговування і оброблення банком дистанційних розпоряджень клієнта є договір банківського рахунка. У договорі обов’язково мають обумовлюватися права, обов’язки та відповідальність сторін, порядок вирішення спорів у разі їх виникнення тощо.

Системи типу «клієнт – банк», «клієнт – Інтернет – банк», «телефонний банкінг», «миттєва безконтактна оплата» та інші системи дистанційного обслуговування на підставі дистанційних розпоряджень клієнта можуть виконувати

функції надання інформаційних послуг згідно з переліком, що зазначений в договорі між банком та клієнтом, здійснення операцій за рахунком клієнта.

Дистанційне обслуговування рахунку за допомогою системи «миттєва безконтактна оплата» регулюється нормативно-правовим актом Національного банку України з питань здійснення операцій, ініційованих із використанням електронних платіжних засобів.

Під час здійснення розрахунків за допомогою систем «клієнт – банк», «клієнт – Інтернет – банк» тощо застосовуються електронні розрахункові документи. Якщо це передбачено договором між банком та клієнтом, то використання клієнтом системи не виключає можливе оброблення банком документів клієнта на паперових носіях.

Реквізити електронного розрахункового документа, що використовуються в системах «клієнт – банк», «клієнт – Інтернет – банк», визначаються договором між банком та клієнтом, але обов'язково цей документ має містити такі з них:

–дату і номер;

–назву, код платника та номер його рахунку;

–код банку платника;

–назву, код одержувача та номер його рахунку;

–код банку одержувача;

–суму цифрами;

–призначення платежу;

–електронний(і) підпис(и) / електронний(і) цифровий(і) підпис(и) відповідно до вимог, установлених нормативно-правовим актом Національного банку з питань застосування електронного підпису в банківській системі України;

–інші реквізити, які під час формування електронного розрахункового документа системою електронних платежів розміщуються в полі «Допоміжні реквізити».

Не дозволяється формування клієнтами електронних розрахункових документів на підставі розрахункових документів, які мають додатки (реєстр розрахункових чеків, реєстр документів за акредитивом тощо), а також формування стягувачем електронних розрахункових документів на підставі платіжних вимог на

примусове списання, стягнення коштів, отримувачем – у разі договірного списання коштів, якщо отримувач коштів – клієнт іншого банку. Ці платіжні вимоги стягував (отримувач) надсилає до банку на паперових носіях.

Під час використання систем «клієнт – банк», «клієнт – Інтернет – банк» банк щоденно архівує електронні розрахункові документи, які відправленні клієнтом, та зберігає їх протягом установленого строку.

Під час використання систем «клієнт – банк», «клієнт – Інтернет – банк» клієнт має дотримуватися всіх вимог, що встановлює банк, з питань безпеки оброблення електронних розрахункових документів. Якщо це передбачено в договорі, то банк має право виконувати періодичні перевірки виконання клієнтом вимог щодо захисту інформації та зберігання засобів захисту і припиняти обслуговування клієнта за допомогою системи в разі невиконання ним вимог безпеки.

Для здійснення операцій за рахунком клієнта (оплата комунальних послуг, телефонних переговорів тощо) за допомогою системи «телефонний банкінг» (дистанційне обслуговування клієнтів за допомогою телефонних каналів зв’язку) клієнт у договорі банківського рахунку або іншому договорі про надання банківських послуг зазначає інформацію, яка потрібна банку для списання ним коштів з рахунку клієнта. Якщо це передбачено договором між банком та клієнтом, то використання клієнтом системи не виключає можливе оброблення банком документів клієнта на паперових носіях.

Ідентифікація клієнта для доступу до системи «телефонний банкінг» здійснюється за допомогою засобів ідентифікації, що передбачені в договорі між банком та клієнтом.

Засоби ідентифікації (номер клієнта, особистий ПІН-код, сукупність цифрових та літерних компонентів тощо) банк надає клієнту після укладення договору.

Передавання дистанційного розпорядження за допомогою системи «телефонний банкінг» та реєстрація його банком здійснюються за погодженим каналом доступу в автоматичному режимі. Дистанційне розпорядження вважається таким, що передане клієнтом та прийняття банком до виконання, якщо клієнт:

–для доступу до системи ввів правильне значення засобу ідентифікації;

–увів код операції та всі параметри, які записуються системою;

–підтвердив це розпорядження.

Якщо клієнт не підтвердив розпорядження на здійснення операції, то банк операцію не виконує, про що інформує клієнта.

Загальна характеристика кожного виду дистанційних каналів збуту банківських продуктів та послуг наведена в таблиці 1.1.

Таблиці 1.1 – Характеристика зв'язку клієнта із дистанційними каналами збуту банківських продуктів

Канал збуту	Характеристика зв'язку
Клієнт-банк	Клієнт отримує банківські послуги завдяки прямому з'єднанню його персонального комп'ютера з банківським сервером через модем
Інтернет-банкінг	Здійснення банківських транзакцій та інформаційний обмін через Інтернет
Банкомат	Проведення розрахунків і готівкових операцій з використанням платіжних карток здійснюється через автоматичні пристрої
Центри самообслуговування	Автономні пристрої, встановлені у філіях банку, супермаркетах, вокзалах та ін.
Інтерактивне телебачення	Системи зворотного зв'язку з можливістю візуального і аудіо спілкування клієнта з банкіром (у відділенні банку або через телебачення з використанням телефону)
Контакт-центр	Системи пристроїв для автоматичного або напівавтоматичного з'єднання клієнта з фахівцями банку
Мобільний зв'язок	Здійснення транзакцій та отримання інформації власником рахунку з використанням супутникової, стільникового зв'язку

Канал збуту	Характеристика зв’язку
	та мобільного телефону
Аутсорсинг	Залучення додаткових агентів з компаній-партнерів для продажу банківських продуктів та послуг

Наявність Інтернет-сайтів дозволяє банку надавати клієнтам широкий спектр мультимедійних, персоніфікованих, інтерактивних послуг, а також згодом скоротити кількість філій та відділень. Можливості Інтернет-банкінгу зазначені на рис. 1.4[2, с. 90-93].

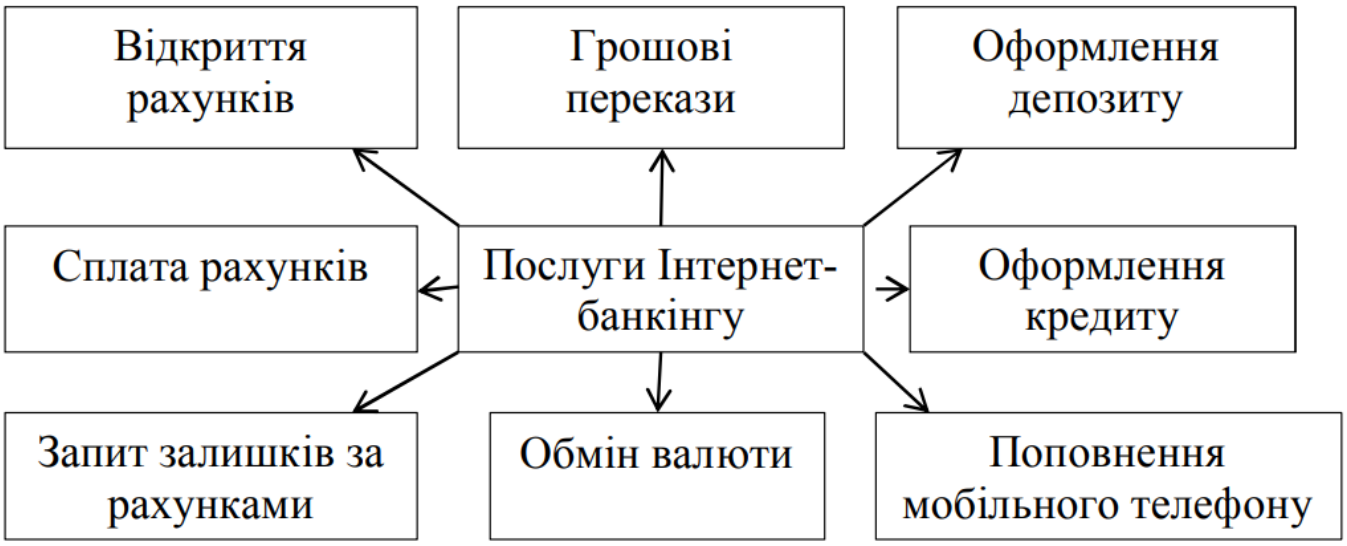


Рисунок. 1.4 – Основні послуги Інтернет-банкінгу вітчизняних банків

Особисто моє перше знайомство з системою iFOBS відбулося в 2007 р., після того, як мене призначили на посаду “адміністратор iFOBS” в АТ “Дельта Банк”. На сьогоднішній день, мая професійна діяльність в сфері IT, банківського сектору. За період з моменту першого знайомства з iFOBS по сьогодні, мені доводилося змінювати роботодавця(Банк), але задачі або залученість до проектів завжди стосувалися інколи меньше, інколи повністю до систему «клієнт – банк».

Розвиток системи iFOBS, можна сказати відбувався на моїх очах, але з різними архітектурними підходами, кількістю сервісів, глибини залежності, спосіб інтеграції та підхід до тестування системи.

1.2 Мета та завдання дослідження

Підвищення ефективності тестування програмного забезпечення за рахунок удосконалення автоматизованого керування тестовими серверами.

Для досягнення цієї мети передбачено вирішення таких завдань.

Аналіз сучасного стану інфраструктури тестових серверів та існуючих підходів до їх керування.

Визначення вимог до розробки автоматизованого керування тестовими серверами.

Розробка архітектури системи та інтерфейсу користувача.

Реалізація механізмів комунікації з серверами та функціоналу для запуску, зупинення та моніторингу статусу Java процесів на серверах.

Тестування системи та оцінка її продуктивності та ефективності.

1.3 Об'єкт та предмет дослідження

Об'єктом дослідження є процес автоматизованого керування тестовими серверами за допомогою програмного забезпечення. Предметом дослідження є розробка і реалізація системи керування тестовими серверами, включаючи розробку інтерфейсу користувача, механізми комунікації з серверами та функціонал для запуску, зупинення та моніторингу статусу Java процесів на серверах.

1.4 Актуальність теми

Актуальність даної теми визначається зростанням обсягів програмного забезпечення та збільшенням кількості тестових серверів у сучасному інформаційному середовищі. Створення програмного забезпечення для автоматизованого керування тестовими серверами дозволить оптимізувати процес тестування та підвищити продуктивність розробки програмного забезпечення. Це важливе завдання для підтримки якості та стабільності програмних продуктів.

1.5 Наукова новизна

Ця дисертація внесе науковий внесок, оскільки впроваджує новий підхід до автоматизації керування тестовими серверами, використовуючи сучасні технології та програмне забезпечення. Результати дослідження сприятимуть підвищенню ефективності тестування програмного забезпечення за рахунок удосконалення автоматизованого керування тестовими серверами, що є значущим внеском в область інформаційних управляючих систем та технологій.

Припустимо, що до впровадження Sense iFOBS Tools час, необхідний для виконання тестів, становив 10 годин, а кількість тестів за один цикл складала 100. Після впровадження системи час виконання тестів може зменшитися на 30%, що означає:

Початковий час виконання тестів: 10 годин * 100 тестів = 1000 годин.

Час виконання тестів після впровадження: 0.7 * 10 годин * 100 тестів = 700 годин.

Економія часу: 1000 годин –700 годин = 300 годин.

Вигода для клієнта: Збільшення продуктивності тестового середовища на 30%, що дозволяє зекономити 300 годин робочого часу.

2. Зниження ризику помилок та виявлення дефектів на ранніх етапах розробки:

Припустимо, що до впровадження Sense iFOBS Tools кількість виявлених дефектів на ранніх етапах розробки становила 20%, а після впровадження ця цифра зросте до 35%.

Початкова кількість виявлених дефектів: 100 дефектів * 20% = 20 дефектів.

Кількість виявлених дефектів після впровадження: 100 дефектів * 35% = 35 дефектів.

Збільшення кількості виявлених дефектів: 35 дефектів –20 дефектів = 15 дефектів.

Вигода для клієнта: Збільшення кількості виявлених дефектів на 15, що дозволяє уникнути їх поширення в подальших етапах розробки та підвищує якість продукту.

3. Зменшення витрат на тестування та підвищення ефективності робочих процесів:

Припустимо, що до впровадження Sense iFOBS Tools витрати на тестування складали $50,000 на місяць. Після впровадження витрати можуть зменшитися на 20%.

Початкові витрати на тестування: $50,000.

Витрати на тестування після впровадження: 0.8 * $50,000 = $40,000.

Зменшення витрат: $50,000 –$40,000 = $10,000.

Вигода для клієнта: Зменшення витрат на тестування на $10,000, що призводить до економії ресурсів компанії.

Ці розрахунки надають загальний уявлення про можливі вигоди, які може отримати клієнт від використання Sense iFOBS Tools.

1.6 Практична цінність

Результати даної магістерської дисертації матимуть велику практичну цінність для організацій, які займаються розробкою програмного забезпечення та тестуванням. Розроблена система автоматизованого керування тестовими серверами допоможе знизити витрати часу та ресурсів на процес тестування, забезпечивши при цьому високу якість та надійність програмного продукту.

Розрахунок практичної цінності в часовому еквіваленті за наступними параметрами: час для встановлення та налаштування ПО (PuTTY порівняно з автоматизованим керуванням), зупинка процесу на 50 серверах вручну за допомогою PuTTY та автоматично з використанням автоматизованим керуванням, може бути представлений у вигляді таблиці, яка показує різницю в часових витратах між двома методами:

Таблиця 1.2 – Різниця в часових витратах між двома методами

Дії	Встановлення та налаштування ПО (PuTTY)	Встановлення та налаштування ПО (автоматизованим керуванням)	Ручний режим (з PuTTY)	Автоматичний режим (з автоматизованим керуванням)	Різниця в часі
Встановлення та налаштування ПО (PuTTY)	4 години	0 годин	0 годин (вже встановлено)	0 годин	4 години
Встановлення та налаштування ПО (автоматизованим керуванням)	0 годин	8 годин	0 годин	0 годин	8 години
Зупинка процесу на 50 серверах (вручну)	0 годин (автоматизовано)	0 годин (автоматизовано)	50 годин	0 годин	50 годин
Зупинка процесу на 50 серверах (автоматично)	0 годин (автоматизовано)	0 годин (автоматизовано)	0 годин	2 години	2 години
Загальний час	4 години	8 годин	50 годин	2 години	48 години

У цьому прикладі, встановлення та налаштування ПО PuTTY вимагає 4 години робочого часу, в той час як встановлення та налаштування ПО автоматизоване керування займає 8 годин.

Зупинка процесу на 50 серверах вручну за допомогою PuTTY вимагає 50 годин робочого часу, в той час як автоматичний режим з використанням автоматизоване керування може виконати ті ж завдання за 2 години.

Різниця в часі між встановленням та налаштуванням ПО PuTTY та автоматизоване керування становить 4 години. Різниця в часі між ручним та автоматичним режимами зупинки процесу на 50 серверах становить 48 годин. Ця інформація показує, як використання автоматизованого керування може значно зекономити час та зробити процес більш ефективним.

Висновки до розділу 1

У результаті проведеного аналізу предметної області була виявлена актуальна проблема, пов'язана із зростанням кількості тестових серверів та стендів у сучасному інформаційному ландшафті. Динаміка цього процесу викликає високу складність та обсяги у тестуванні програмного забезпечення. Магістерська дисертація визначає мету створення програмного забезпечення для автоматизованого керування тестовими серверами з функціоналом моніторингу, запуску та зупинення. Для досягнення цієї мети було поставлено конкретні завдання, такі як аналіз існуючих підходів, визначення вимог, розробка архітектури та інтерфейсу, реалізація механізмів комунікації та функціоналу системи.

Об'єкт дослідження – процес автоматизованого керування тестовими серверами, предмет – розробка системи керування, включаючи інтерфейс, комунікаційні механізми та функціонал для моніторингу Java процесів. Актуальність теми обумовлена збільшенням обсягів програмного забезпечення та кількості тестових серверів, що ставить завдання оптимізації тестування та підвищення продуктивності розробки. Наукова новизна полягає в унікальному підході до автоматизації керування тестовими серверами, використовуючи сучасні технології.

Результати даного дослідження матимуть значущий практичний вплив на організації, що займаються розробкою програмного забезпечення та тестуванням. Розроблена система дозволить ефективно оптимізувати час та ресурси, забезпечуючи високу якість продукту. Часовий розрахунок вказує на значний економічний виграш в порівнянні із традиційними методами, підкреслюючи велику практичну цінність результатів магістерської дисертації для індустрії програмного забезпечення.

2 ТЕОРЕТИЧНІ ОСНОВИ

У цьому розділі проводиться аналіз та систематизація теоретичних аспектів, необхідних для розуміння та підготовки до подальшого дослідження.

2.1 Огляд літератури

Огляд літератури є ключовим етапом у магістерській дисертації, оскільки він допомагає визначити сучасний стан досліджуваної області, виявити основні тенденції та висвітлити розвинуті підходи.

Актуальність теми в сучасному інформаційному ландшафті.

Актуальність теми "Автоматизація керування тестовими серверами" у сучасному інформаційному ландшафті визначається рядом проблем та викликів, які зустрічають організації та розробники програмного забезпечення. Декілька ключових аспектів включають:

Збільшення об'єму та складності програмного забезпечення:

З ростом розміру та складності програмних продуктів збільшується необхідність в автоматизації тестування. Тестові сервери стають основним елементом управління та виконання тестових завдань для ефективного виявлення помилок та забезпечення якості.

Потреба в постійних інтеграціях та постачанні (CI/CD):

Організації все частіше використовують моделі постійної інтеграції та постачання для швидкого та автоматизованого впровадження змін. Керування тестовими серверами важливо для ефективного виконання тестів в рамках цих процесів.

Розподілені та хмарні обчислення:

Застосування розподілених систем та хмарних технологій ускладнює керування тестовими серверами. Необхідно розробляти рішення, які дозволяють ефективно взаємодіяти з серверами, які можуть бути розташовані в різних локаціях чи обчислювати тестові завдання в хмарних середовищах.

Системи з великою кількістю серверів і стендів:

З великою кількістю тестових серверів і стендів виникає потреба в їхньому ефективному керуванні та координації. Проблеми зі скалабельністю, стабільністю та швидкістю стають основними викликами.

Необхідність автоматизованого моніторингу:

Забезпечення стабільності і продуктивності тестових серверів вимагає автоматизованого моніторингу, який дозволяє вчасно виявляти та вирішувати проблеми.

У зв'язку з цим, розробка системи автоматизованого керування тестовими серверами вирішує ці проблеми та відповідає викликам сучасного інформаційного середовища, сприяючи покращенню якості та швидкості розробки програмного забезпечення.

- Існуючі підходи та рішення:

Існуючі підходи та рішення в автоматизації керування тестовими серверами:
Системи управління конфігурацією (Configuration Management Systems):

Приклади: Ansible, Puppet, Chef.

Переваги: Автоматизація розгортання та конфігурації серверів, декларативний підхід до опису стану системи.

Недоліки: Може вимагати додаткового часу для вивчення, можливість збоїв при несправності конфігурації.

Контейнеризація:

Приклади: Docker, Kubernetes.

Переваги: Легка мобільність, ізольованість та стандартизація середовищ, швидкість розгортання.

Недоліки: Деякі виклики з оркестрацією для масштабування, вимагає глибокого розуміння контейнеризації.

Інструменти для автоматизації тестування:

Приклади: Selenium, JUnit, TestNG.

Переваги: Автоматизоване виконання тестів, можливість інтеграції з CI/CD пайплайнами.

Недоліки: Вимагає додаткового часу для розробки та підтримки тестових скриптів.

CI/CD системи:

Приклади: Jenkins, Travis CI, GitLab CI.

Переваги: Постійна інтеграція та постачання, автоматизовані робочі процеси, швидке виявлення та виправлення помилок.

Недоліки: Може бути складним для конфігурування, вимагає налагодження CI/CD пайплайнів.

Системи моніторингу:

Приклади: Prometheus, Nagios, ELK Stack.

Переваги: Автоматизоване виявлення та відстеження аномалій, інформативні звіти.

Недоліки: Може вимагати значних ресурсів для великих систем, можливі ложнопозитиви.

Віртуалізація:

Приклади: VMware, VirtualBox, Hyper-V.

Переваги: Ізольованість середовищ, можливість тестування на різних конфігураціях.

Недоліки: Великі витрати ресурсів, повільне розгортання.

Менеджери контейнерів:

Приклади: Docker Compose, Podman.

Переваги: Зручна локальна розробка та тестування, легка оркестрація.

Недоліки: Обмежені можливості порівняно з Kubernetes для масштабування.

Підбір оптимального рішення залежить від конкретних потреб та характеристик проекту, інфраструктури та обсягу тестових завдань.

- Технологічні та програмні рішення:

Технологічні та програмні рішення в автоматизації керування тестовими серверами:

Kubernetes:

Особливості: Оркестрація контейнерів, автоматизоване масштабування, декларативна конфігурація, управління службами.

Можливості: Розгортання, оновлення та керування додатками у контейнерах на різних серверах.

Docker Compose:

Особливості: Зручна локальна розробка, опис сервісів у файлі docker-compose.yml, управління контейнерами.

Можливості: Локальне тестування, створення та керування множиною контейнерів.

Ansible:

Особливості: Інфраструктурний код, декларативна конфігурація, автоматизація завдань.

Можливості: Автоматизоване розгортання, конфігурація та керування серверами.

Jenkins:

Особливості: Система CI/CD, розширюваність через плагіни, автоматизація робочих процесів.

Можливості: Постійна інтеграція, постачання та автоматичне тестування.

Prometheus:

Особливості: Відкрита система моніторингу, скалабельність, виявлення аномалій.

Можливості: Моніторинг роботи тестових серверів, аналіз метрик та виявлення проблем.

Selenium:

Особливості: Автоматизоване тестування веб-додатків, крос-браузерні та крос-платформенні можливості.

Можливості: Запуск та виконання автоматизованих тестів на різних серверах.

GitLab CI:

Особливості: Вбудована система CI/CD, організація робочих процесів у GitLab.

Можливості: Автоматизована збірка, тестування та постачання програмного забезпечення.

VirtualBox:

Особливості: Віртуалізація, підтримка різних операційних систем.

Можливості: Створення віртуальних тестових середовищ для тестування різних конфігурацій.

Terraform:

Особливості: Інфраструктурний код, декларативна мова конфігурації.

Можливості: Автоматизоване створення та управління інфраструктурою для тестових серверів.

Кожне з цих рішень має свої унікальні особливості та можливості, і вибір залежить від конкретних потреб проекту та вподобань розробників.

- Дослідження:

Таблиця 2.1 – Найпопулярніші засоби автоматизації в IT компаніях

Розробник	Функціональне	Навантажувальне	Якість коду	Керування тестами
IBM	+	+	+	+
Borland	+	+	-	+
AutomatedQA	+	+	-	+
HP	+	+	+	+
Open-Source	Abbot, Selenium, Watir	Grinder, Jmeter, OpenSTA	GCT, NCover, Cobertura	FitNesse, TestLink

[3,18]

Прогалини та напрямки подальших досліджень в автоматизації керування тестовими серверами:

Ефективність інструментів автоматизації:

Розвиток більш ефективних та швидких інструментів для автоматизації керування тестовими серверами, спрямованих на скорочення часу розгортання та оптимізацію робочих процесів.

Інтеграція зі штучним інтелектом (ШІ):

Вивчення можливостей використання ШІ для оптимізації прийняття рішень щодо конфігурації та масштабування тестових середовищ.

Моделювання реального середовища:

Розробка методів моделювання реального виробничого середовища для тестування, що враховувати різні конфігурації та умови.

Моніторинг забезпечення якості:

Розробка інструментів для постійного моніторингу та аналізу якості роботи тестових серверів з урахуванням метрик продуктивності та відповідності стандартам.

Системи управління та оркестрації для крауд-тестування:

Дослідження систем, які дозволяють ефективно керувати великою кількістю тестових серверів під час проведення крауд-тестування.

Стійкість та безпека:

Розширення досліджень у сфері стійкості та безпеки тестових серверів, зокрема виявлення та усунення вразливостей.

Гнучкість та сумісність з різними інфраструктурами:

Розробка рішень, які забезпечують гнучкість управління тестовими серверами та їх сумісність з різними хмарними і локальними інфраструктурами.

Стандартизація процесів:

Розробка стандартизованих процесів та протоколів для взаємодії між різними інструментами автоматизації та системами управління тестовими серверами.

Оптимізація витрат ресурсів:

Дослідження нових стратегій для оптимізації витрат ресурсів та забезпечення їх ефективного використання.

Вивчення впливу на навколишнє середовище:

Оцінка екологічного впливу великої кількості тестових серверів та розробка екологічно ефективних рішень.

2.2 Основні поняття та технології

Основні поняття та технології у магістерській дисертації є ключовим для розуміння фундаментальних аспектів обраної теми, а саме "Автоматизація керування тестовими серверами". У цьому розділі представлені та розкрити основні терміни, поняття та технології, які стануть основою для подальшого розуміння дослідження.

Цей перелік стає основоположним елементом, на якому будується подальший аналіз та дослідження в рамках розробки системи автоматизованого керування тестовими серверами.

2.3 Аналіз існуючих систем керування тестовими серверами

Ansible (https://www.ansible.com/) – це інструмент для автоматизації конфігурації та керування серверами. Він може бути використаний для автоматизації задач управління процесами на багатьох серверах і віртуальних машинах одночасно.

Після завершення налаштування ми змоделюємо такий сценарій:

На веб-сервері NGINX стався незапланований простій.

Монітор процесів, який підтримує Dynatrace OneAgent, миттєво виявляє збій процесу NGINX і генерує сповіщення про проблему на платформі Dynatrace.

Вихідний модуль Dynatrace, як визначено в книзі правил, яку використовує Event-Driven Ansible, активно опитує події збою.

Керований подією Ansible у відповідь на подію виконує шаблон завдання, який виконує такі дії:

- ініціює створення квитка інциденту ServiceNow;
- спроби перезапустити процес NGINX;
- оновлює статус заяви про інцидент на «Виконується»;
- закриває заявку, лише якщо процес NGINX успішно відновлено.

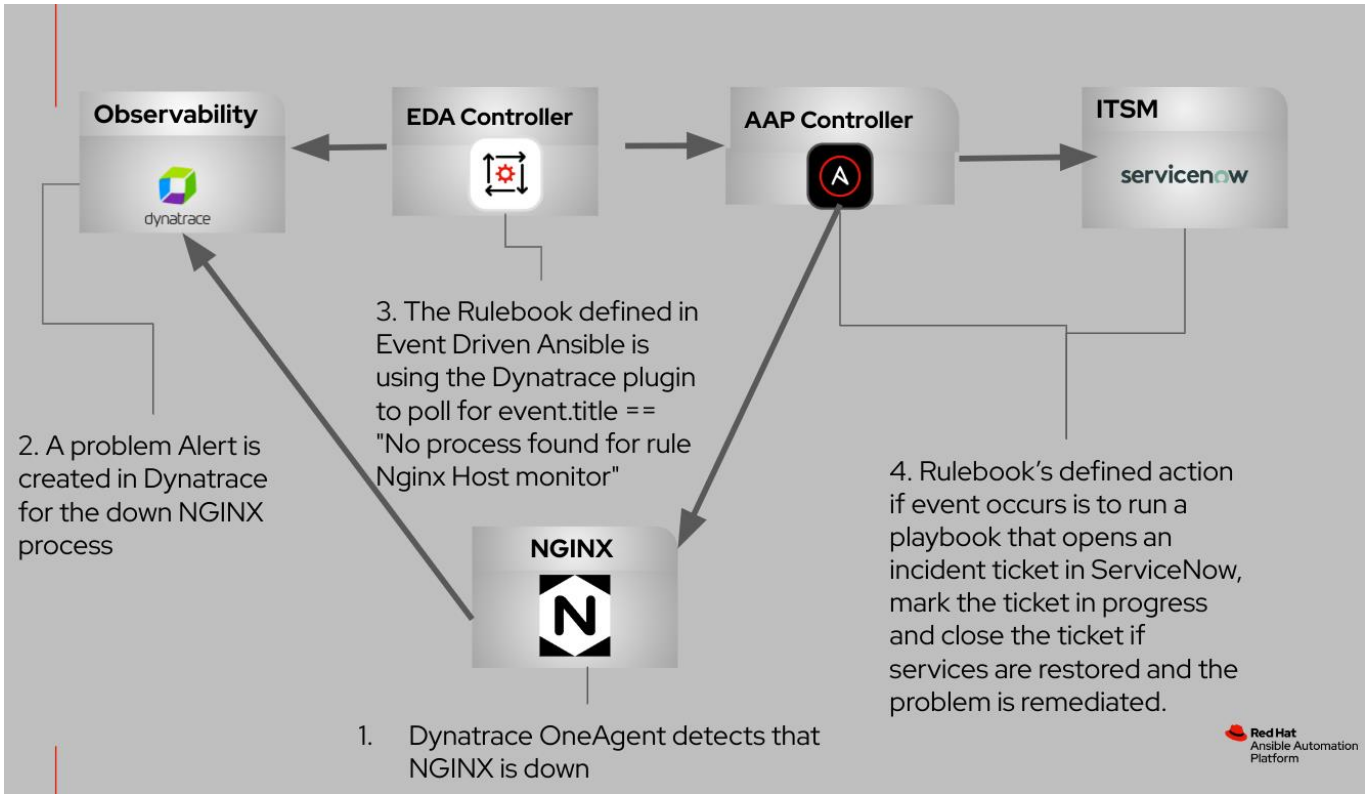


Рисунок 2.1 – Блок-схема між інтегрованими компонентами[13]

Chef (https://www.chef.io/, рис. 2.2) – це інструмент для автоматизації конфігурації та розгортання інфраструктури. Він дозволяє описати стан системи і задачі у вигляді коду та застосувати їх до багатьох серверів.

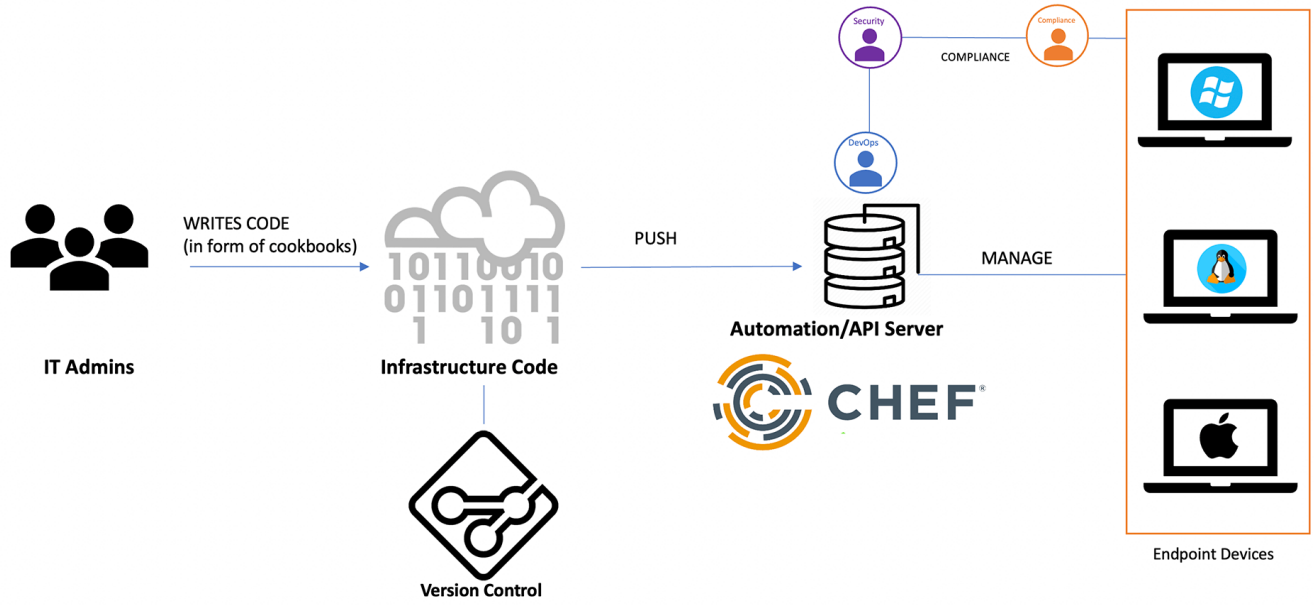


Рисунок 2.2 – Схема автоматизації інфраструктури[14]

Це усуває помилки та витрачає час на ручні процеси, тим самим збільшуючи швидкість організації. Chef може допомогти автоматизувати конфігурацію інфраструктури, керування оновленнями та безперервну відповідність вашого ноутбука/настільного комп'ютера, допомагаючи вашій організації підтримувати узгоджену конфігурацію та стандарти відповідності на підприємстві. Крім того, це дозволяє отримати безперервну видимість конфігурації та стану відповідності IT-ресурсів на підприємстві за допомогою єдиного інструменту[14].

Puppet (https://www.puppet.com/) – це інструмент для автоматизації управління конфігурацією серверів. Він надає можливість описати бажаний стан системи та застосувати його до великої кількості серверів.

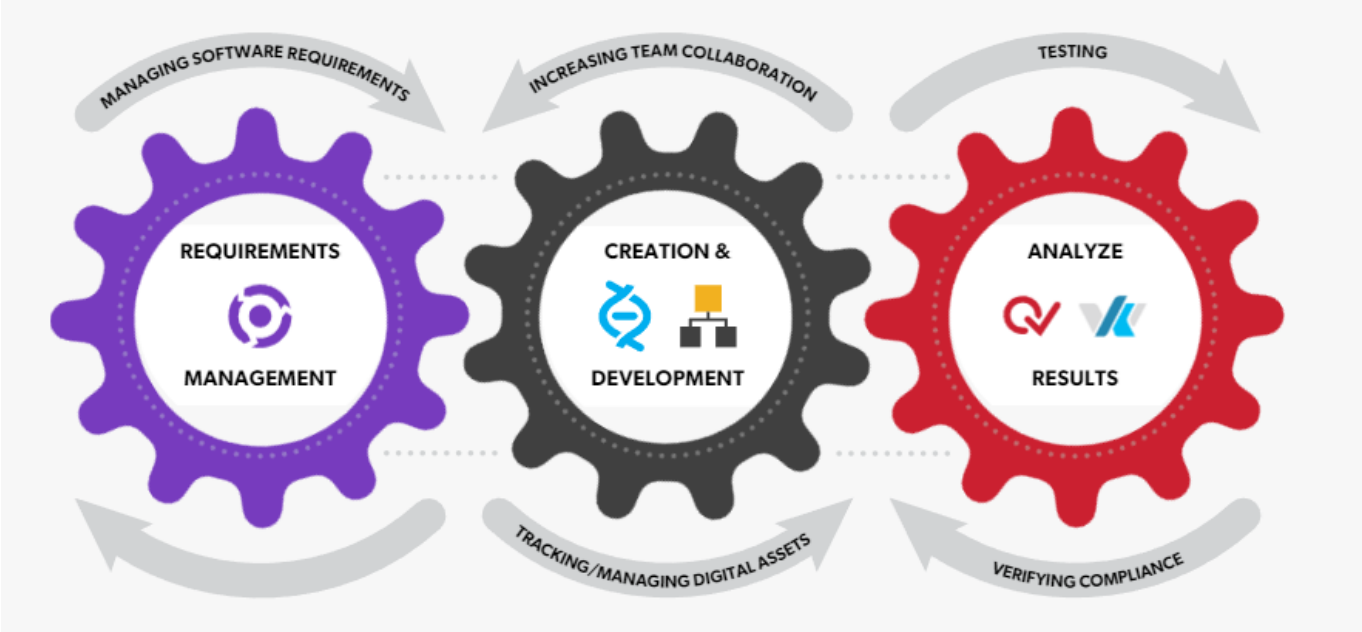


Рисунок 2.3 – Інструменти розробки автомобільного програмного забезпечення для кожного етапу розробки[16]

Програмні рішення Perforce дають можливість команді виробляти конкурентоспроможну, високоякісну автомобільну продукцію, яка є безпечною, захищеною та сумісною — незалежно від розміру чи обсягу проекту.

SaltStack (https://saltproject.io/) – це інструмент для автоматизації та управління інфраструктурою. Він дозволяє керувати процесами на багатьох серверах і віртуальних машинах одночасно (рис. 2.4).

SaltStack Config — це інструмент, який дозволяє вам керувати конфігураціями машини, встановлюючи та відстежуючи дрейф необхідного програмного забезпечення для певної машини. Незалежно від того, використовуєте його для встановлення агентів моніторингу на всіх машинах або розгортання програм баз даних чи повних стеків програм, SaltStack Config може допомогти виконати ці завдання.

Конфігурація SaltStack розташована поверх відкритого вихідного коду Salt. Salt Open Source — це інструмент командного рядка, де Salt Master підтримує елементи конфігурації та застосовує їх до Salt Minions. Солт-міньйони — це всі машини, якими керує Солт. SaltStack Config застосовує рівень графічного інтерфейсу користувача, додаючи до Salt Open Source планування завдань, звітність, масштабованість тощо.

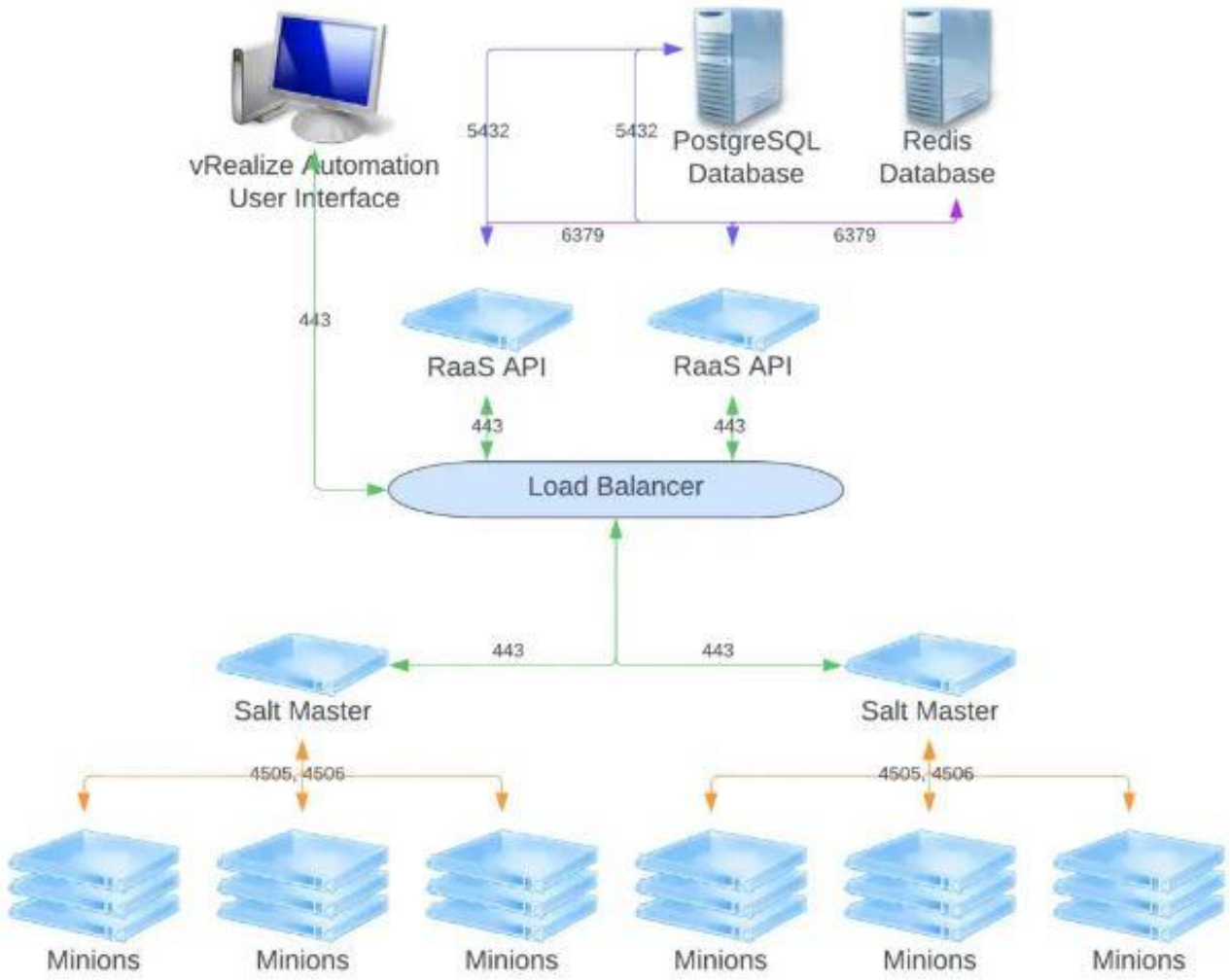


Рисунок 2.4 – Архітектура платформи SaltStack

Порівняння інструментів Ansible, Chef, Puppet і SaltStack за країною компанії-розробника наведено в табл. 2.3.

Таблиця 2.3 – Порівняльна таблиця за різними критеріями для інструментів Ansible, Chef, Puppet і SaltStack

Критерій	Ansible	Chef	Puppet	SaltStack
Мова програмування	Ansible використовує YAML для опису конфігурації.	Chef використовує Ruby для опису рецептів конфігурації.	Puppet використовує спеціальну мову Puppet DSL.	SaltStack використовує YAML для опису стейтів та Python для скриптів і модулів.
Агенти	Не потребує агентів на вузлах, працює за принципом SSH або WinRM.	Вимагає агенти (Chef-клієнт) на вузлах.	Вимагає агенти (Puppet-агент) на вузлах.	Вимагає агенти (Salt Minion) на вузлах.
Архітектура	Архітектура "інфраструктура як код".	Архітектура "інфраструктура як код".	Архітектура "інфраструктура як код".	Архітектура "інфраструктура як код".
Спільнота та екосистема	Має велику активну спільноту та багато доступних модулів.	Має активну спільноту та готові рецепти в репозиторіях.	Має активну спільноту та багато доступних модулів.	Має активну спільноту та модулів.

Критерій	Ansible	Chef	Puppet	SaltStack
Швидкість впровадження	Швидке впровадження завдяки простому YAML синтаксису.	Потребує часу для вивчення Ruby та оголошення рецептів.	Вимагає часу для вивчення Puppet DSL та налаштування модулів.	Вимагає часу для вивчення YAML та Python.
Комунікація	Використовує SSH, WinRM, або інші протоколи для зв'язку з вузлами.	Комунікація через Chef-сервер або Chef Solo.	Комунікація через Puppet Master або Puppet Standalone.	Комунікація через Salt Master.
Масштабованість	Здатний керувати великою кількістю вузлів.	Може бути важким для масштабування великих інфраструктур.	Здатний керувати великою кількістю вузлів.	Здатний керувати великою кількістю вузлів.
Підтримка операційних систем	Підтримує багато різних Linux-дистрибутивів та Windows.	Підтримує багато різних Linux-дистрибутивів та Windows.	Підтримує багато різних Linux-дистрибутивів та Windows.	Підтримує багато різних Linux-дистрибутивів та Windows.
Компанія та країна розробник	Red Hat, США	Chef Software, Inc., США	Puppet, Inc., США	SaltStack, Inc., США

Критерій	Ansible	Chef	Puppet	SaltStack
Переваги	Активно підтримується великою спільнотою. Має безкоштовний варіант Ansible Community та комерційну версію Ansible Automation Platform	Має комерційні та відкриті версії. Має активну спільноту і багато готових рецептів (cookbooks)	Підтримує відкритий джерело (Puppet Open Source) і комерційну версію (Puppet Enterprise). Має активну спільноту	Має відкрите джерело (Salt Open) і комерційну версію (SaltStack Enterprise). Відомий своєю швидкістю та простотою використання

Висновки до розділу 2

У висновку до розділу 2 "Теоретичні основи" магістерської дисертації можна зазначити, що в процесі аналізу та систематизації теоретичних аспектів, пов'язаних з автоматизацією керування тестовими серверами, було виявлено глибоку актуальність обраної теми в сучасному інформаційному ландшафті.

Актуальність дослідження базується на розширенні об'єму та складності програмного забезпечення, що призводить до зростання потреби в автоматизованому тестуванні. Враховуючи тенденції в розвитку постійної інтеграції та постачання, а також використання розподілених та хмарних обчислень, керування тестовими серверами стає ключовим елементом для забезпечення ефективного виконання тестових завдань.

Розділ також детально розглянув існуючі підходи та рішення у цій області, включаючи системи управління конфігурацією, контейнеризацію, інструменти для автоматизації тестування, CI/CD системи, системи моніторингу, віртуалізацію та інші. Кожне з цих рішень володіє своїми унікальними перевагами та можливостями,

що відображається у різних варіантах їхнього використання залежно від конкретних потреб та характеристик проекту.

Основна мета розділу полягає в створенні теоретичної основи для подальшого дослідження та розробки системи автоматизованого керування тестовими серверами, що враховує сучасні тенденції та вимоги індустрії розробки програмного забезпечення. Аналіз літератури та розгляд існуючих підходів надають твердий теоретичний фундамент для подальшої практичної реалізації та вдосконалення процесів управління тестовими серверами з метою підвищення якості та швидкості розробки програмного забезпечення в умовах сучасного інформаційного середовища.

У даному розділі магістерської дисертації ми вивчили та систематизували ключові поняття, терміни та технології, які є необхідними для розуміння основ теми "Автоматизація керування тестовими серверами". Зазначені інструменти управління конфігурацією серверів, такі як Ansible, Chef, Puppet і SaltStack, представляють сучасний рівень автоматизації у сфері інформаційних технологій.

Ретельний аналіз цих інструментів, який ми провели у розділі, вказує на їхню різноманітність, переваги та особливості. Виокремлено країни розробників, основні переваги кожного інструменту та їхню застосовність в управлінні тестовими серверами.

Ця інформація стане фундаментом для подальшого розгортання системи автоматизованого керування тестовими серверами в рамках магістерської роботи. Враховуючи відмінності між інструментами, матиму можливість обрати оптимальний засіб для вирішення конкретних завдань та досягнення поставлених цілей.

Отже, зазначений у розділі аналіз не лише розкриває сутність інструментів автоматизації, але й створює підґрунтя для подальших досліджень та розробки системи, яка відповідатиме сучасним вимогам у сфері тестування програмного забезпечення.

3 АНАЛІЗ ТА ПРОЕКТУВАННЯ СИСТЕМИ

У третьому розділі магістерської дисертації розглядається аналіз та проектування системи автоматизованого керування тестовими серверами. Цей розділ важливий, оскільки він визначає основи розробки системи та встановлює вимоги до її функціоналу та архітектури. Нижче наведені пункти, які необхідно вирішити у цьому розділі.

Інфраструктура АТ Сенс Банк побудована наступним чином:

- Блок ІТ;
- Блок Бізнес;
- Департамент безпеки;
- Департамент персоналу;
- Департамент підтримки клієнтів;
- Інші структурні підрозділи.

Так як, тема магістерської дисертації є автоматизоване керування тестовими серверами, то розглянемо більш детально інфраструктурну схему блоку ІТ та взаємодію між іншими підрозділами (рис. 3.1).

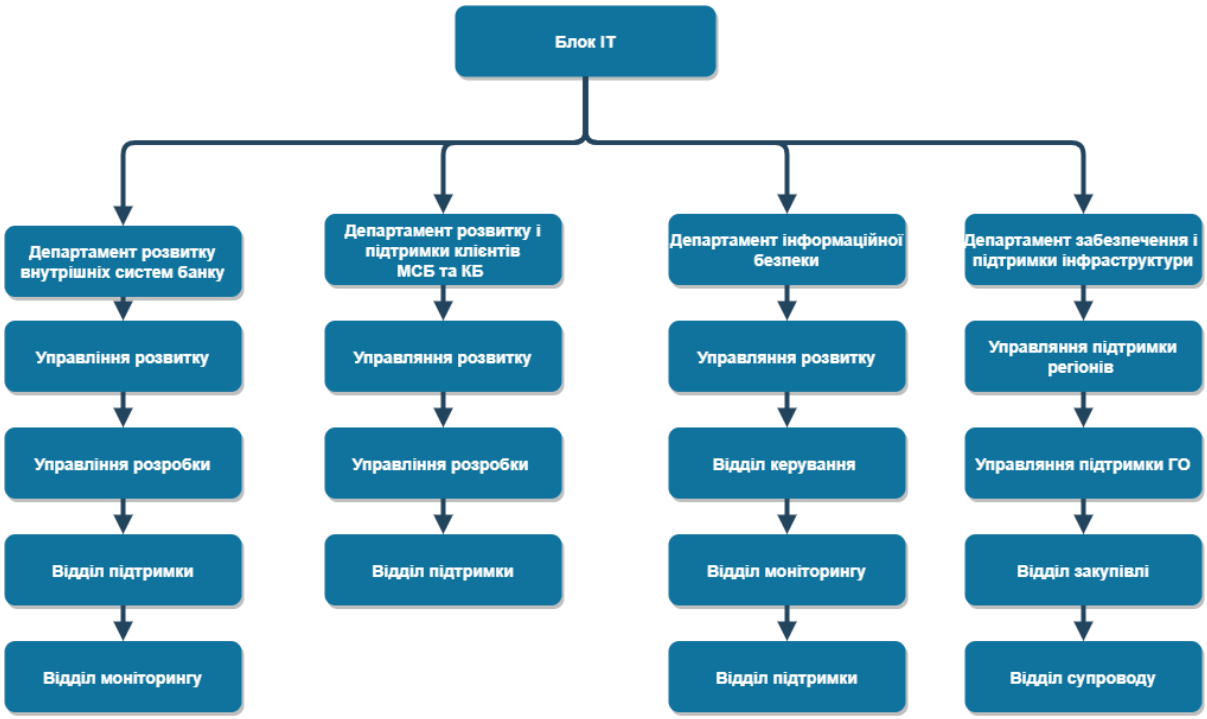


Рисунок 3.1 – Організаційна схема Блоку ІТ

Згідно з приведеної схеми блоку IT можна побачити, що в кожному департаменті присутні управління розвитку, які в свою чергу складаються з проектних команд в середньому з 10 співробітників (табл. 3.1).

Таблиця 3.1 – Склад проектної команди

Назва посади	Кількість	Обов'язки
Керівник проекту	1	Керування проектами та ресурсами команди
Системний аналітик	2	Аналіз архітектури систем, технічний опис вимог до розробки. Розгортання та керування тестовими серверами.
Бізнес аналітик	3	Аналіз вимог від бізнес підрозділів та формування ТЗ для розробників
Тестувальник	4	Тестування систем та новими розробками, створення заявок з помилками.

Аналіз системи повинен враховувати потреби кожного департаменту, визначати оптимальні методи взаємодії між відділами та забезпечувати потрібний рівень комунікації між управлінням розвитку та іншими структурними підрозділами.

3.1 Вимоги до системи

Вимоги до автоматизованого керування тестовими середовищами. Перше та найголовніше – це зменшити час недоступності тестового середовища та максимально автоматизувати керування тестовими серверами.

Елементи розробки наведені в табл. 3.2.

Таблиця 3.2 – Основні елементи розробки

Назва елементу	Опис елементу	Пріоритет
Дизайн системи	Простий та приємний дизайн системи	4
Функціональність системи	Зупинка, запуск, оновлення, моніторинг	1
Налаштування системи	Звучність у керуванні налаштуванні системою	2
Масштабування	Можливість додавати видаляти тестові стенди, сервера тощо	3
Простота в використані	Мінімальна залученість системного аналітика	5

Технічні вимоги:

- безпека та надійність – Забезпечення безпеки та надійності системи для уникнення втрати даних чи некоректного функціонування;
- інтеграція з іншими інструментами – Система повинна інтегруватися з іншими інструментами розробки та тестування для максимальної ефективності;
- підтримка різних операційних систем – Система повинна бути сумісною та підтримувати різні операційні системи (Windows, Linux, MacOS).

Ці вимоги повинні забезпечити створення ефективної, легко використовуваної та автоматизованої системи для керування тестовими середовищами, що відповідає вимогам до часу недоступності та максимально спрощує розробку та тестування програмного продукту.

3.2 Архітектура системи

Розвиток інформаційних технологій, а особливо в банківській сфері не стоїть на місці та постійно розвивається, вдосконалюється та автоматизується. Згідно даних опублікованих в офіційних джерелах то Сенс банк завжди знаходиться в ТОП-5 IT

розвинутих фінансових установ. А для того щоб займати високий рівень IT розвитку, потрібно вкладати дуже великі ресурси як людськи так і фінансові.

До фінансових ресурсів відноситься вартість тестових серверів, які потрібні для розгортання тестових середовищ ідентичних продуктовій, щоб мінімізувати появу помилок в процесу переносу на продуктів оновлень, нових доробок та інше.

Ресурси які витрачаються на розгортання, керування та оновлення тестовими середовищами (серверами), для можливості виконувати свої роботи іншими підрозділами проектними командами витрачається дуже багато часу. Для того, щоб оцінити та отримати розуміння загальної картини, розглянемо загальну архітектурну схему залежних підрозділів від тестових середовищ (рис. 3.2).

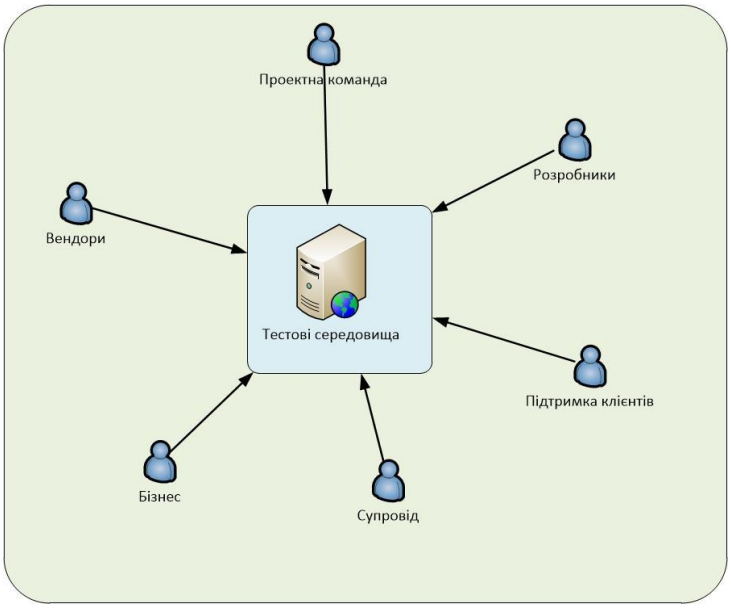


Рисунок 3.2 – Загальна архітектурна схема залежних підрозділів від тестових середовищ

Комунікація з серверами в захищеній банківській мережі та зовнішніми підключеннями клієнта вимагає високого рівня безпеки та ефективності. Нижче наведено загальний опис механізмів комунікації:

Шифрування даних:

Усі дані, що передаються між клієнтом та серверами банку, повинні бути шифрованими для захисту від несанкціонованого доступу. Зазвичай

використовується протокол HTTPS (HTTP Secure) для зашифрування трафіку між клієнтом та серверами.

Використання VPN (Віртуальна Приватна Мережа):

Для забезпечення конфіденційності та цілісності даних може бути використана VPN для безпечного тунелювання через неприватні мережі, такі як Інтернет.

Аутентифікація та авторизація:

Всі клієнти повинні пройти процедуру аутентифікації для підтвердження їхньої ідентичності. Після цього система визначає рівень доступу (авторизація) до різних ресурсів на серверах.

Фаєрвол та IPS (Системи Захисту Вторгнень):

Використання фаєрволів та систем захисту вторгнень допомагає фільтрувати та виявляти підозрілу активність в мережі, що може вказувати на можливий кібератаки.

Моніторинг та аудит:

Система моніторингу слідкує за активністю в мережі, виявляє аномалії та спроби несанкціонованого доступу. Журнали аудиту зберігають інформацію про всі події для подальшого аналізу та слідкування за діяльністю користувачів.

Токени та Двоетапна аутентифікація:

Використання токенів та двоетапної аутентифікації додає додатковий рівень безпеки. Крім пароля, користувач повинен підтвердити свою особу за допомогою іншого фактору, такого як одноразовий код або біометричні дані.

SSL/TLS Протоколи:

Використання найновіших версій протоколів SSL (Secure Sockets Layer) або TLS (Transport Layer Security) для забезпечення безпеки комунікації та обміну криптографічних ключів.

Правильна конфігурація серверів:

Важливо налаштувати сервери згідно з найкращими практиками безпеки, включаючи вимкнення зайвих служб, встановлення необхідних патчів та обмеження прав доступу.

Ці механізми допомагають створити захищену банківську мережу та забезпечити безпеку комунікації з серверами та зовнішніми підключеннями клієнта.

Топологія балансувальників (load balancer topology) визначає, як самі балансувальники розташовані та як взаємодіють у мережі для розподілу навантаження. Важливо обрати ефективну топологію в залежності від конкретних потреб та характеристик системи. Ось кілька типових топологій балансувальників:
Однорівнева (Flat Topology, рис. 3.3):

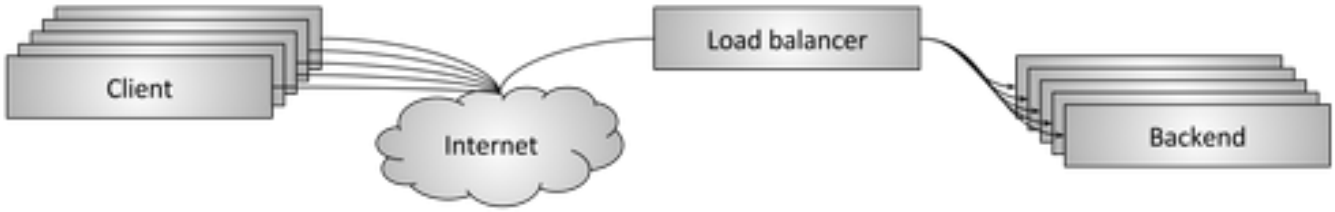


Рисунок 3.3 – Схема однорівневого балансування[4,1]

У цій топології всі балансувальники розташовані на одному рівні, без ієрархії. Кожен балансувальник має прямий доступ до серверів. Це простий підхід, але може бути обмежений в масштабуванні.
Топологія з ланцюгами (Chained Topology, рис. 3.4):

Рисунок 3.4 – Схема ланцюгового балансування[4,2]

У цій топології балансувальники утворюють ланцюг, де кожен наступний балансувальник обробляє трафік, який не був розподілений попереднім балансувальником. Це дозволяє розділити функції балансування навантаження та забезпечити більшу гнучкість.
Кластерна топологія (Cluster Topology, рис. 3.5):

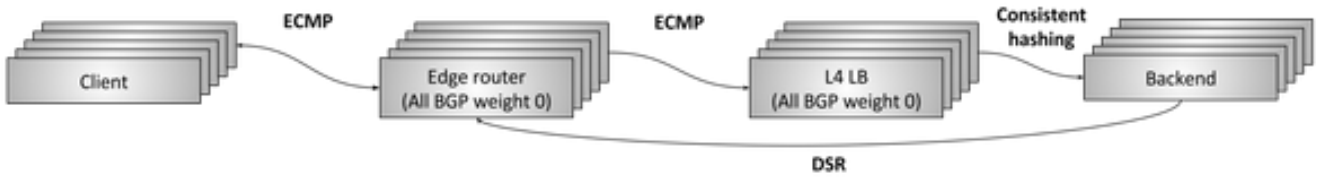


Рисунок 3.5 – Схема кластерного балансування[4,3]

У кластерній топології кілька балансувальників спільно працюють в якості одного логічного об'єкту, який розподіляє трафік. Це забезпечує високу доступність та масштабованість, адже нові балансувальники можуть легко додаватися до кластера.

Топологія з різною доступністю (Geographically Dispersed Topology, рис. 3.6):

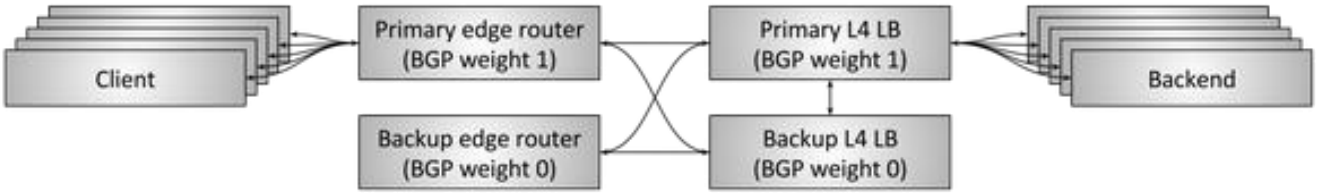


Рисунок 3.6 – Схема балансування з різно доступністю[4,4]

У цій топології балансувальники розташовані на різних фізичних місцях або в різних дата-центрах. Це може бути корисно для забезпечення високої доступності та витривалості при виникненні збоїв в одному регіоні.

Кожна топологія має свої переваги та недоліки, і вибір конкретної залежить від вимог до масштабування, доступності, надійності та інших факторів системи. Комбінації різних топологій також можуть використовуватися для досягнення оптимального розподілу навантаження в конкретному випадку.

Таблиця 3.3 – Типи тестових середовищ

Назва тестового середовища	Опис
DEV	Тестові середовища на якій виконуються всі нові розробки
TEST	Тестові середовища на яких проводяться тестування нових версій та розробок
Навантажувальний тест	Тестове середовище на якій виконується тестування навантаження на роботу систем

Пред-прод	Тестове середовище на якій виконується тестування встановлення та перевірки системи супроводом.

Розглянемо більш детально мінімальну архітектуру одного тестового середовища (рис. 3.7).

Середній набір серверів тестового середовища.

Максимальна архітектура тестового середовища може складатися від 20 та більше серверів на яких запущені відповідні процеси.

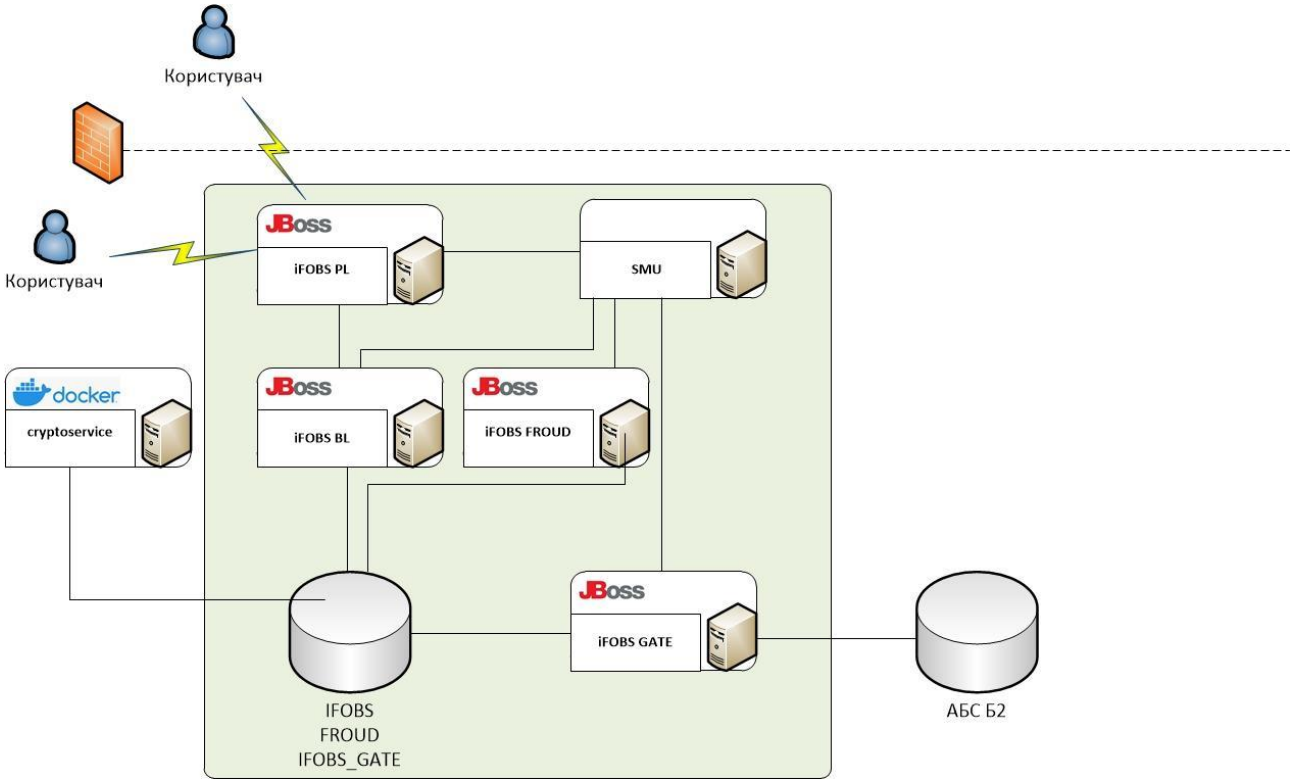


Рисунок 3.7 – Мінімальна архітектура одного тестового середовища.

3.3 Проектування інтерфейсу користувача

Проектування інтерфейсу користувача (UI) є важливою частиною створення програмного продукту. Ефективний та зручний інтерфейс дозволяє користувачам легко взаємодіяти з програмою. Кроки з проектування інтерфейсу користувача:

1. Збір вимог – є першим та одним з найважливіших етапів в проектуванні інтерфейсу користувача (UI). На цьому етапі важливо зрозуміти, для кого призначений продукт і які конкретні потреби користувачів повинен вирішити:
   - аналіз цільової аудиторії;
   - вивчення конкурентів;
   - участь користувачів;
   - визначення функціональності;
   - створення вимог до інтерфейсу;
   - аналіз технічних вимог;
2. Створення концепції:
   - створення Wireframes;
   - визначення основних функціональних блоків;
   - розробка структури меню;
   - взаємодія та переходи;
   - підготовка прототипу;
3. Скетчі та макети:
   - головний екран;
   - форми додавання/редагування;
   - пошук;
   - налаштування;
   - контроль помилок та сигналізація;
   - адаптивний дизайн;
4. Вибір кольорової палітри:
   - текст та фон;
   - важливі дії;

- помилки та попередження;
- акцентні кольори;
- спокійні кольори;

5. Типографіка:
   - основний шрифт;
   - розміри шрифтів;
   - виділення тексту;
   - міжрядковий інтервал;
   - колір тексту;
   - шрифтові сімейства;
   - респонсивність;
6. Інтерактивність:
   - навігація;
   - кнопки та елементи керування;
   - вікна та панелі;
   - форми та поля введення;
   - анімації;
   - відгуки та підказки;
7. Адаптивність:
   - рідкісний дизайн;
   - контентно-орієнтований дизайн;
   - тестування на різних пристроях;
8. Тестування прототипу:
   - визначення мети тестування;
   - взаємодія з реальними користувачами;
   - збір фідбеку;
   - аналіз фідбеку та внесення змін;
   - тестування різних сценаріїв;
   - перевірка навігації;
   - тестування на різних пристроях;

- підготовка фінальної версії;

9. Реалізація:
   - вибір технології;
   - створення інтерфейсу;
   - стилі та графіка;
   - інтерактивність;
   - адаптивність;
   - тестування;
   - документація;
   - запуск та моніторинг;
10. Тестування та оптимізація:
   - функціональне тестування;
   - тестування адаптивності;
   - тестування швидкодії;
   - тестування безпеки;
   - тестування умов екстремального використання;
   - збір фідбеку від користувачів;
   - оптимізація;
   - тестування бекенду;
   - повторне тестування;
   - моніторинг вирішення питань;
11. Документація:
   - створити документацію з інтерфейсу для користувачів та розробників.

3.4 Механізми комунікації з серверами

Після проведення аналізу інфраструктури та архітектури тестових середовищ, головною проблемою являється керування процесами на серверах та час який витрачається на зупинку та запуск процесу, а також супутніх дій, які потрібно

виконувати в процесі, а саме видалення логів, додавання нових, оновлення існуючих робочих файлів системи від яких залежить робота всієї системи.

При використанні безкоштовного та дозволеного ПО в банку Putty, на обслуговування одного сервера системному аналітику потрібно 10-25 хвилин. Детальний перелік дій, які потрібно виконати при умові, що були виконана попередня підготовка по налаштуванню, а саме, внесені всі сервери тестових середовищ в налаштуваннях ПО Putty та прописані всі залежності за призначенням (рис. 3.8):

- підключитися до сервера, ввести логін та пароль;
- ввести команду яка зупинить відповідний процес на сервері;
- ввести команду для видалення логів, тимчасових файлів та інше;
- за необхідності скопіювати нові або оновити існуючі файли;
- ввести команду яка запустить процес на сервері.

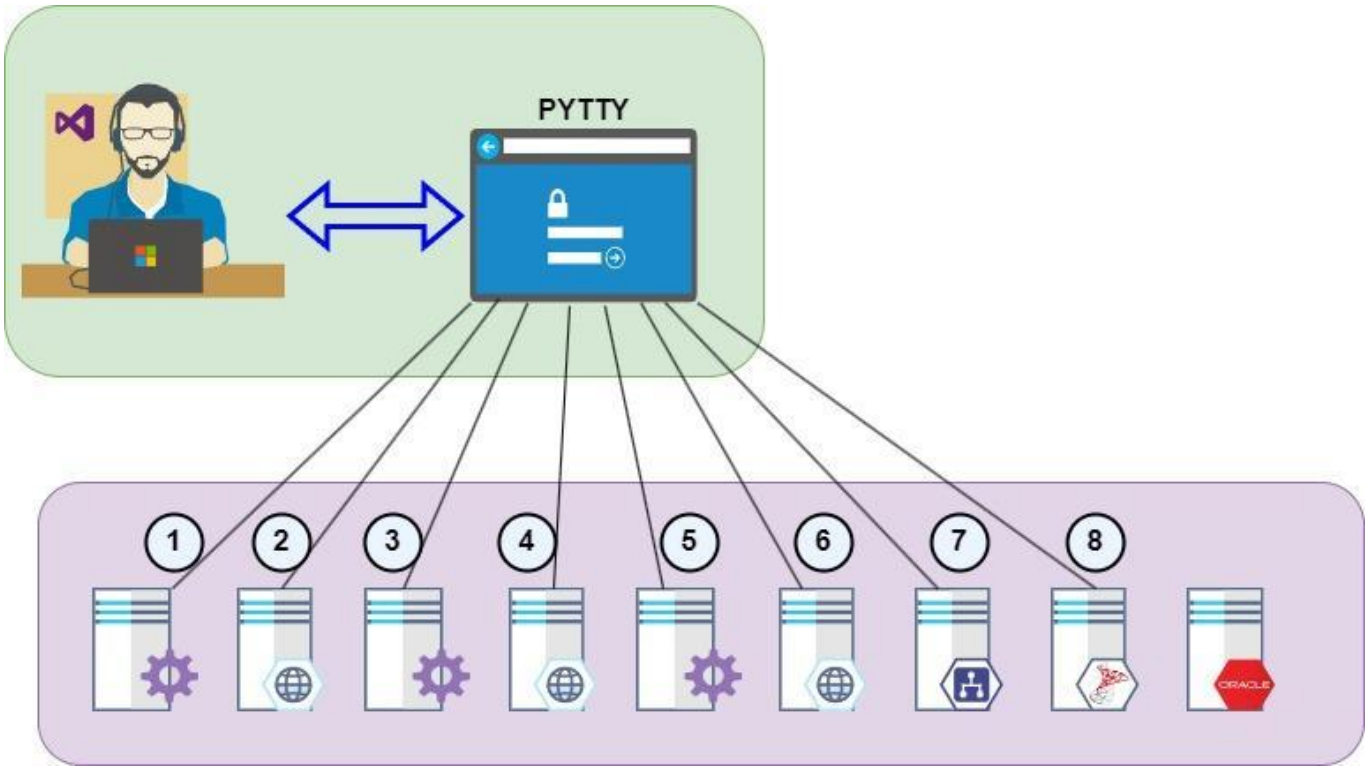


Рисунок 3.8 – Керування серверами в послідовно

Крім перелічених вище дій які потрібно виконати на сервері, існує же велика залежність в послідовності зупинці та запуску процесів на серверах.

За відомими даними про кількість серверів та час якій потрібно для виконання робіт можна підрахувати час недоступності тестового стенду в залежності від кількості компонентів (табл. 3.4).

Таблиця 3.4 – Розрахунок недоступності тесту залежно від конфігурації

Назва набору серверів	Кількість серверів	Час недоступності(хв.)
Мінімальний	5	50-125
Середній	13	130-325
Максимальний	20+	200-500+

Під час виконання розробок, оновлень, перевірки виправлень то що, дуже часто необхідно перезавантажувати повністю один або інший тестовий стенд, в залежності від потреб.

3.5 Функціонал для запуску, зупинення та моніторингу статусу java процесів на серверах

Функціонал для запуску, зупинення та моніторингу статусу Java-процесів на серверах в контексті даної магістерської дисертації буде реалізований через розробку спеціалізованого програмного забезпечення. Нижче подано функціонал, який буде реалізований у додатку:

- Інтерфейс користувача:

Розроблення зручного інтерфейсу для користувача для взаємодії з системою. Інтерфейс повинен мати можливість вибору серверів, налаштування параметрів запуску та зупинення процесів.

- Запуск Java-процесів:

Реалізація механізму для запуску Java-процесів на обраних серверах. Передбачення можливості налаштування параметрів запуску, таких як параметри JVM (Java Virtual Machine) та шлях до JAR-файлу.

- Зупинення Java-процесів:

Розроблення функціоналу для зупинення запущених Java-процесів. Врахування можливості зупинення окремих процесів або груп процесів на вибраних серверах.

- Моніторинг статусу:

Реалізація системи моніторингу для відстеження статусу запущених Java-процесів. Це може включати в себе виведення інформації про використання ресурсів, стан пам'яті, час роботи та інші ключові метрики.

- Логування:

Введення системи логування для реєстрації подій, таких як запуск, зупинення та зміни статусу Java-процесів. Це допоможе при аналізі проблем та відлагодженні.

- Безпека:

Врахування питань безпеки при реалізації, такі як автентифікація та авторизація користувачів, захист від несанкціонованого доступу до системи.

- Шкалювання:

Розробка механізмів для швидкого розгортання та масштабування системи на різних серверах.

- Інтеграція з іншими інструментами:

Врахування можливості інтеграції з іншими інструментами для моніторингу, логування чи автоматизації конфігурації.

Цей функціонал визначається в контексті розробки системи автоматизованого керування тестовими серверами і може бути доповнений чи змінений відповідно до конкретних вимог та умов дослідження.

Висновки до розділу 3

У висновку розділу "Аналіз та проектування системи автоматизованого керування тестовими серверами" виявляється, що цей етап дослідження визначає стратегічні аспекти розробки та функціональні вимоги до системи. Організаційна схема Блоку ІТ та розгляд його взаємодії з іншими підрозділами уточнюють необхідність інтеграції та комунікації всередині компанії.Детальний аналіз проектної команди, представленої у таблиці 3.1, підкреслює важливість розподілу обов'язків та

спільної роботи для успішного розвитку системи. Поставлені вимоги до системи включають аспекти як безпеки та надійності, так і інтеграції з іншими інструментами розробки та підтримки різних операційних систем.

Архітектурна схема системи, викладена у розділі, відображає необхідність ефективного використання ресурсів та мінімізації витрат при розгортанні тестових середовищ. Оцінка величезних фінансових та людських ресурсів, витрачених на ці процеси, підкреслює вагомість завдань, які стоять перед системою автоматизованого керування тестовими серверами.

Надано повний огляд механізмів комунікації в захищеній банківській мережі та зовнішніх підключень клієнта, де високий рівень безпеки є пріоритетом. Використання шифрування даних, VPN, аутентифікації та авторизації, фаєрволів та IPS, моніторингу та аудиту, токенів та двоетапної аутентифікації, а також протоколів SSL/TLS та правильної конфігурації серверів свідчать про комплексний підхід до забезпечення безпеки мережі.

Детально розглянуті топології балансувальників, такі як однорівнева, з ланцюгами, кластерна та з різною доступністю, надають обширний вигляд на оптимальний вибір залежно від потреб та характеристик системи.

Розділ завершується вивченням архітектур тестових середовищ, розглядом мінімальної та максимальної конфігурації, що дозволяє краще розуміти вимоги та можливості для тестування продукту.

Окремий аспект розділу становить проектування інтерфейсу користувача (UI), де відбувається аналіз цільової аудиторії, визначення вимог та розробка концепції. Процес включає створення wireframes, розробку структури меню, взаємодію та переходи, а також підготовку прототипу з урахуванням адаптивного дизайну.

Загалом, цей розділ не лише розкриває принципи безпеки та механізми комунікації в банківській мережі, але й надає повний обзор інфраструктурних рішень та інтерфейсних аспектів, що формують основу для подальшого успішного розвитку банківської системи.

4 РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Вибір технологій та інструментів

Перед вибором конкретних технологій та інструментів проведено детальний аналіз функціональних та нефункціональних вимог до розроблюваного програмного продукту. Визначено основні функції, які повинна виконувати система, а також враховано параметри продуктивності, масштабованості та безпеки.

При виборі мови програмування для розробки програмного продукту з функціоналом керування тестовими стендами було вирішено використовувати мову програмування Python. Нижче подано обґрунтування цього вибору.

Обґрунтування вибору Python:

- **зручність розробки** – Python є мовою високого рівня, що визначається читабельністю та простотою синтаксису. Це дозволяє розробникам швидше писати, тестувати і супроводжувати код, сприяючи зручності розробки та підтримці;

- широка підтримка бібліотек – Python має велику кількість стандартних бібліотек і сторонніх модулів, які спрощують розробку та дозволяють легко інтегрувати різноманітні функції без необхідності писати код з нуля;

- загальна популярність – Python є однією з найпопулярніших мов програмування, що сприяє наявності великої спільноти розробників. Це означає доступність обширної бази знань, форумів, та ресурсів для отримання допомоги та підтримки;

- розширюваність та інтеграція – Python легко інтегрується з іншими мовами програмування і технологіями, що може бути важливим фактором для розвитку майбутніх можливостей системи та її інтеграції з існуючими рішеннями.

Вибір мови для оформлення дизайну – CSS.

Окрім вибору мови програмування для реалізації функціоналу, важливим елементом є вибір мови для оформлення дизайну. В даному випадку, обрано Cascading Style Sheets (CSS) з наступним обґрунтуванням.

Обґрунтування вибору CSS для оформлення дизайну:

- розширена підтримка браузерів – CSS є стандартом для оформлення веб-сторінок та має велику підтримку у всіх сучасних браузерах, що гарантує однаковий вигляд та поведінку на різних платформах;

- легкість використання та навчання – синтаксис CSS простий та легко засвоюється розробниками, що робить його ефективним інструментом для оформлення інтерфейсів;

- можливості адаптивного дизайну – CSS дозволяє легко створювати адаптивний дизайн, що адаптується до різних розмірів екранів, що є важливим для забезпечення коректного відображення на різних пристроях;

– інтеграція з HTML – CSS безперервно використовується для стилізації HTML-документів, що робить його природним вибором для оформлення користувацького інтерфейсу.

Вибір системи управління базою даних.

Для управління базою даних вибрано системи Oracle та SQLite з наступним обґрунтуванням:

- Oracle – Oracle обрано за його потужність та надійність у роботі з великим обсягом даних. Ця система баз даних має широкі можливості управління та оптимізації запитів, що є важливим для ефективної роботи великих систем;

- SQLite – SQLite вибрано для легкості впровадження та використання в невеликих проектах. Вона не вимагає окремого серверу баз даних та може бути легко вбудована в програмний продукт, що робить її оптимальним вибором для невеликих тестових стендів та експериментальних проектів;

- комбінація Oracle та SQLite – обрано комбінацію Oracle та SQLite для реалізації багаторівневої архітектури баз даних, де Oracle використовується для обробки та зберігання обсяжних даних, а SQLite — для локальних потреб та швидкого доступу до окремих компонентів системи. Це дозволяє оптимально використовувати ресурси в залежності від потреб проекту.

Середовище розробки: Visual Studio

1. Інтегроване Середовище Розробки (IDE):

Широкий Функціонал: Visual Studio надає повноцінне інтегроване середовище розробки з рядом корисних інструментів.

Підтримка Python: Visual Studio має вбудовану підтримку для розробки на Python, що полегшує роботу з цією мовою в середовищі VS.

2. Відлагодження та Профілювання:

Інтегрована Відлагоджувальна Панель: В Visual Studio вбудована потужна відлагоджувальна панель, яка спрощує виявлення та виправлення помилок.

Інструменти Профілювання: VS надає інструменти для аналізу продуктивності коду (профілювання), що корисно для оптимізації.

3. Підтримка Командного Рядка та Версійний Контроль:

Інтеграція з Командним Рядком: Visual Studio підтримує роботу з Python через командний рядок, що дозволяє використовувати різні інструменти та скрипти.

Вбудована Підтримка Git: Visual Studio включає в себе інструменти для роботи з Git, що спрощує ведення версій контролю коду.

Обрана комбінація Python та Visual Studio забезпечить зручну та продуктивну розробку з використанням потужного інтегрованого середовища та широкого спектру функцій мови Python.

Для вирішення проблематики з часом який витрачається на керування тестовими стендами, розробимо програму з інтерфейсом та базою даних.

Масштабування:

Вертикальне та Горизонтальне Масштабування:

Вертикальне масштабування включає в себе збільшення потужності одного сервера, тоді як горизонтальне масштабування полягає в додаванні нових серверів для розподілу навантаження.

4.2 Реалізація інтерфейсу користувача

Дизайн системи програми написаний на мові програмування Python, яка використовує різноманітні бібліотеки та модулі для створення графічного інтерфейсу

користувача з взаємодією з базою даних. Короткий опис використаних бібліотек та їх призначення у цьому коді:

sys:

Призначення: Модуль sys дає доступ до змінних та функцій, пов'язаних із інтерпретатором Python. У коді використовується для доступу до аргументів командного рядка та інших системних опцій.

urllib.error.HTTPError:

Призначення: Використовується для обробки помилок, що пов'язані з HTTP-запитами. У цьому коді, використовується для обробки помилок, які можуть виникнути при роботі з HTTP-запитами.

PyQt5:

Призначення: Бібліотека PyQt5 використовується для створення графічного інтерфейсу користувача. Включає класи та методи для роботи з віджетами, подіями, анімацією тощо.

requests:

Призначення: Модуль requests використовується для виконання HTTP-запитів. У даному випадку, використовується для отримання даних з сервера.

datetime:

Призначення: Модуль datetime дозволяє працювати з датою та часом. У коді використовується для роботи з датами.

BaseInteraction:

Призначення: Власний модуль, який включає в себе клас для взаємодії з базою даних. З використанням цього класу можна взаємодіяти з базою даних та виконувати операції читання та запису даних.

Код використовується для розробки GUI-додатку для керування серверами та взаємодії з базою даних, і він залежить від різних бібліотек для реалізації різноманітних функцій.

Структура графічного інтерфейсу реалізована за допомогою мови розмітки Qt (QML). Розмітка визначає різні елементи форми та їх стилізацію. Опис верхнерівневих елементів (кнопок):

1. frame_topbar (QFrame)`:

    Тип: QFrame

    Стилізація:

        frameShape: QFrame::StyledPanel

        frameShadow: QFrame::Raised

    Вміст:

        Горизонтальний контейнер (QHBoxLayout) з кнопками:

        pushButton_server_top: Кнопка "Server"

        pushButton_add_top: Кнопка "Add server"

        pushButton_change_top: Кнопка "Change"

        pushButton_delete_top: Кнопка "Delete"

        pushButton_setting_top: Кнопка "Setting"

        pushButton_status_top: Кнопка "Status server"

2. frame_control (QFrame)`:

    Тип: QFrame

    Стилізація:

        styleSheet: Задано стилізацію за допомогою CSS для фону і елементів QPushButton, QLabel, QTableWidget та QHeaderView.

    Вміст:

        Містить різні кнопки та таблиці (QPushButton, QLabel, QTableWidget, QHeaderView), які зазначені у властивостях frame_control. Стилізація використовується для кращого оформлення та відображення на екрані.

3. Стилізація:

Стилізація для QPushButton, QLabel, QTableWidget та QHeaderView реалізована з використанням CSS. Стилізація включає задання кольорів, тіні, фону і інших характеристик для покращення зовнішнього вигляду форми.

Код з додатку А визначає структуру та вигляд графічного інтерфейсу для програми.

Вигляд і структура елементів форм для сторінок з інформацією наведена на рис. 4.1–4.6.

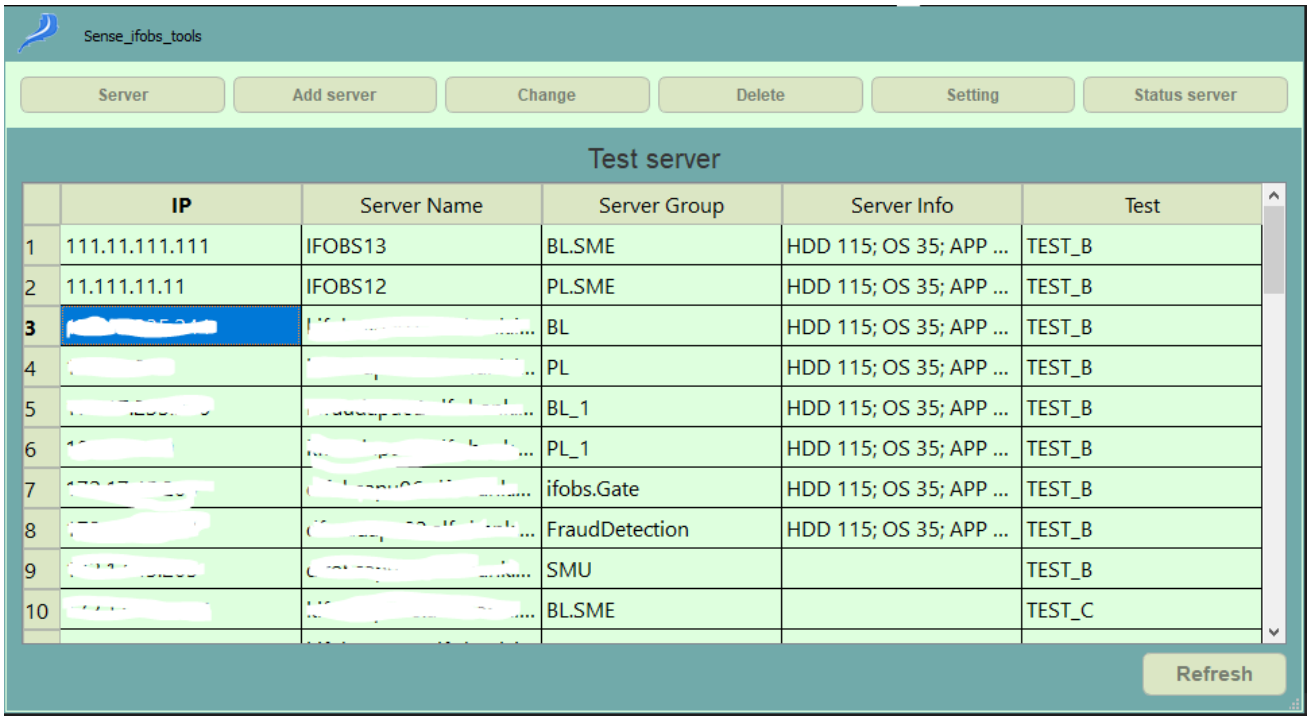


Рисунок 4.1 – Форма Server

Структура елементів форми для сторінки з інформацією про тестові сервери.

page_server (QWidget)`:

Тип: QWidget

Дочірні елементи:

label_server (QLabel)`:

Тип: QLabel

Стилізація:

Шрифт: Arial, розмір 15, жирний

Вирівнювання тексту: Qt::AlignCenter

Текст: "Test server"

table_server (QTableWidget)`:

Тип: QTableWidget

Стилізація:

Шрифт: розмір 12

Мінімальний розмір секції горизонтального заголовка: 150

Властивості відображення полос прокрутки

Колонки таблиці:

"IP"

"Server Name"

"Server Group"

"Server Info"

"Test"

Розміщення: за встановленими властивостями QTableWidget horizontalLayout_5 (QHBoxLayout)`:

Тип: QHBoxLayout

Містить кнопку "Refresh" (pushButton_refresh) і горизонтальний роздільник (horizontalSpacer)

horizontalSpacer (QSpacerItem)`:

Тип: QSpacerItem

Орієнтація: Qt::Horizontal

Розміри: ширина 40, висота 20

pushButton_refresh (QPushButton)`:

Тип: QPushButton

Текст: "Refresh"

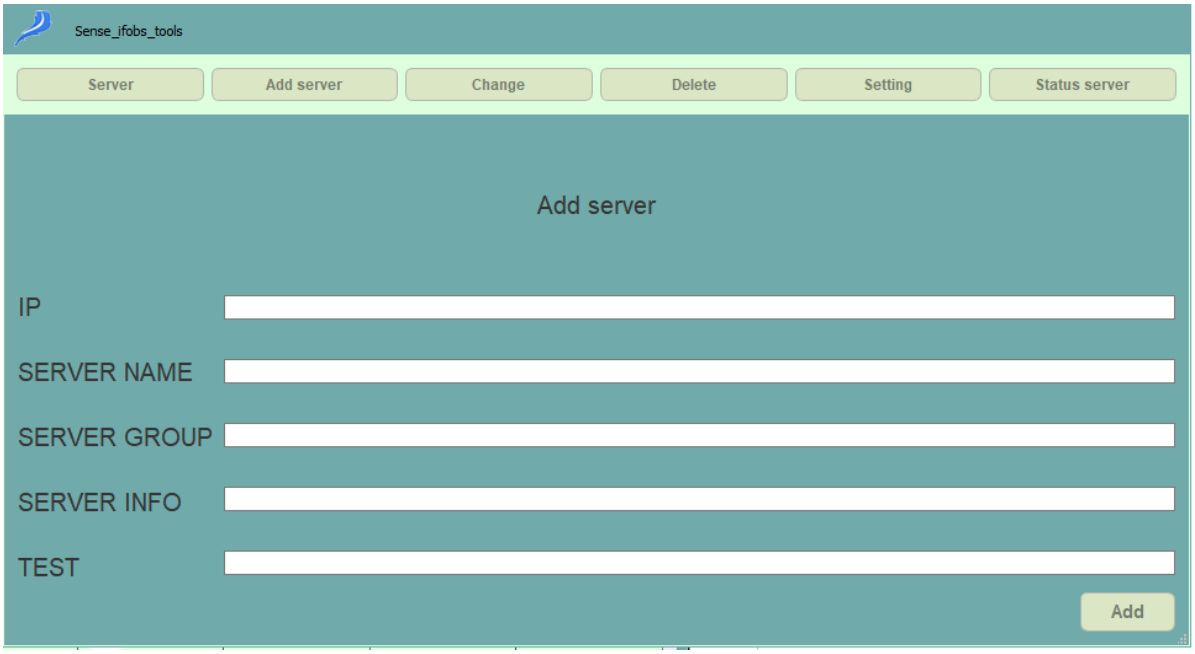


Рисунок 4.2 –Форма add server

Структура елементів форми для сторінки з додаванням нового сервера.

page_add (QWidget)`:

Тип: QWidget

Дочірні елементи:

label (QLabel)`:

Тип: QLabel

Текст: "Add server"

Вирівнювання тексту: Qt::AlignCenter

horizontalLayout_6 (QHBoxLayout)`:

Тип: QHBoxLayout

Містить дві вертикальні компоненти (verticalLayout_6, verticalLayout_5), які розташовані горизонтально.

verticalLayout_6 (QVBoxLayout)`:

Тип: QVBoxLayout

Містить п'ять міток QLabel:

"IP"

"SERVER NAME"

"SERVER GROUP"

"SERVER INFO"

"TEST"

Розділювач між елементами: 30

verticalLayout_5 (QVBoxLayout)`:

Тип: QVBoxLayout

Містить п'ять текстових полів QLineEdit для введення даних:

lineEdit_ip_add

lineEdit_name_add

lineEdit_group_add

lineEdit_info_add

lineEdit_test_add

Розділювач між елементами: 30

horizontalLayout_7 (QHBoxLayout)`:

Тип: QHBoxLayout

Містить:

signal_add (QLabel)`:

Тип: QLabel

Мінімальний розмір: ширина 120

Текст: порожній рядок

horizontalSpacer_add (QSpacerItem)`:

Тип: QSpacerItem

Орієнтація: Qt::Horizontal

Розміри: ширина 40, висота 20

pushButton_add (QPushButton)`:

Тип: QPushButton

Текст: "Add”

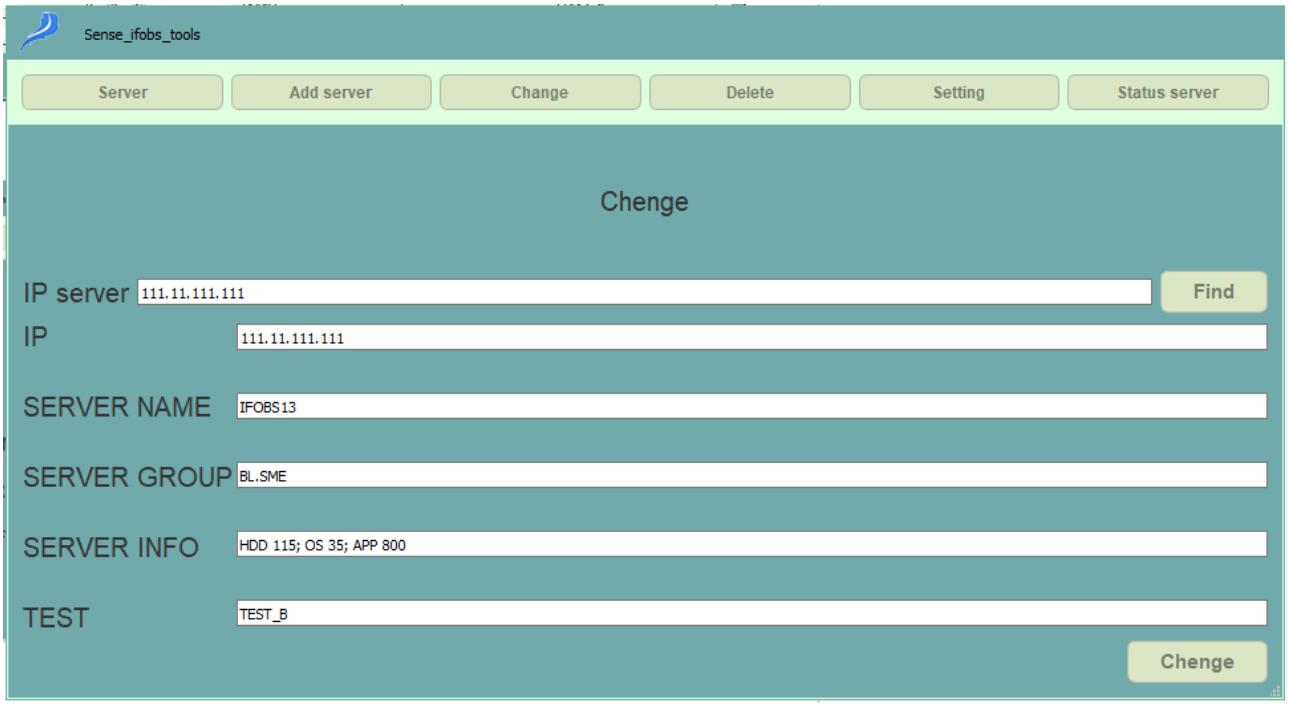


Рисунок 4.3 – Форма Change

Структура елементів форми для сторінки зміни даних сервера.

page_chenge (QWidget)`:

Тип: QWidget

Дочірні елементи:

label_title_chenge (QLabel)`:

Тип: QLabel

Текст: "Change"

Вирівнювання тексту: Qt::AlignCenter

horizontalLayout_10 (QHBoxLayout)`:

Тип: QHBoxLayout

Містить три елементи:

label_find_chenge (QLabel)`:

Тип: QLabel

Текст: "IP server"

lineEdit_find_cheng (QLineEdit)`:

Тип: QLineEdit

pushButton_find_cheng (QPushButton)`:

Тип: QPushButton

Текст: "Find"

horizontalLayout_9 (QHBoxLayout)`:

Тип: QHBoxLayout

Включає дві вертикальні компоненти (verticalLayout_11, verticalLayout_10), які розташовані горизонтально.

verticalLayout_11 (QVBoxLayout)`:

Тип: QVBoxLayout

Містить п'ять міток QLabel:

"IP"

"SERVER NAME"

"SERVER GROUP"

"SERVER INFO"

"TEST"

Розділювач між елементами: 30

verticalLayout_10 (QVBoxLayout)`:

Тип: QVBoxLayout

Містить п'ять текстових полів QLineEdit для введення даних:

lineEdit_ip_chenge

lineEdit_name_chenge

lineEdit_group_chenge

lineEdit_info_chenge

lineEdit_test_chenge

Розділювач між елементами: 30

horizontalLayout_8 (QHBoxLayout)`:

Тип: QHBoxLayout

Включає три елементи:

signal_chenge (QLabel)`:

Тип: QLabel

Мінімальний розмір: ширина 120

Текст: порожній рядок

horizontalSpacer_chenge (QSpacerItem)`:

Тип: QSpacerItem

Орієнтація: Qt::Horizontal

Розміри: ширина 40, висота 20

pushButton_chenge (QPushButton)`:

Тип: QPushButton

Текст: "Change”

Описані елементи форми утворюють інтерфейс для зміни даних сервера. Включають в себе назву операції, поле для введення IP сервера для пошуку, блок для введення нових даних сервера та кнопку "Change".

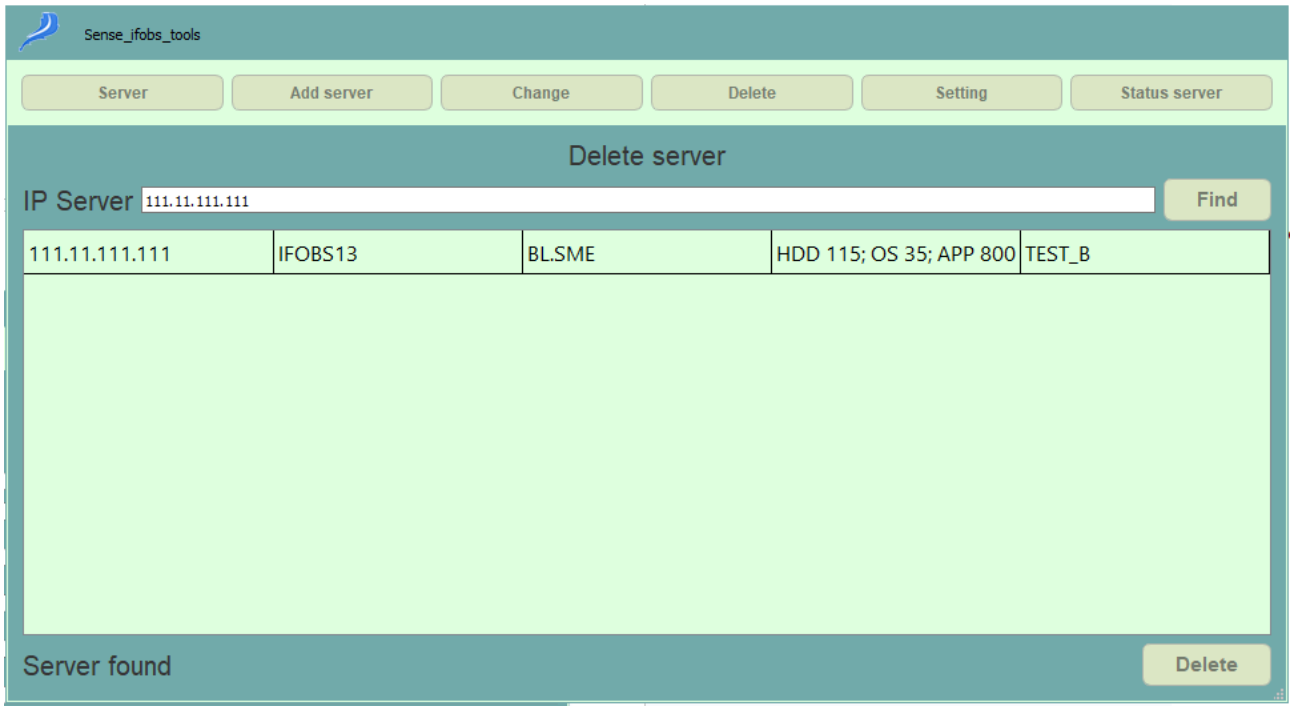


Рисунок 4.4 –Форма Delete

Структуру елементів форми для сторінки видалення сервера.

page_delete (QWidget)`:

Тип: QWidget

Дочірні елементи:

label_delete (QLabel)`:

Тип: QLabel

Текст: "Delete server"

Вирівнювання тексту: Qt::AlignCenter

horizontalLayout_11 (QHBoxLayout)`:

Тип: QHBoxLayout

Містить три елементи:

label_find_del (QLabel)`:

Тип: QLabel

Текст: "IP Server"

lineEdit_find_del (QLineEdit)`:

Тип: QLineEdit

pushButton_find_del (QPushButton)`:

Тип: QPushButton

Текст: "Find"

tableWidget_2 (QTableWidget)`:

Тип: QTableWidget

Має 5 стовпців та 0 рядків.

Налаштовані різні атрибути вигляду, такі як розміри, політика розміщення, вирівнювання тексту та інші.

Чотири стовпці (IP, SERVER NAME, SERVER GROUP, TEST) мають вирівнювання тексту "AlignLeading|AlignVCenter".

horizontalLayout_12 (QHBoxLayout)`:

Тип: QHBoxLayout

Включає три елементи:

signal_del (QLabel)`:

Тип: QLabel

Мінімальний розмір: ширина 120

Текст: порожній рядок

horizontalSpacer_del (QSpacerItem)`:

Тип: QSpacerItem

Орієнтація: Qt::Horizontal

Розміри: ширина 40, висота 20

pushButton_del (QPushButton)`:

Тип: QPushButton

Текст: "Delete"

Описані елементи форми утворюють інтерфейс для видалення сервера. Включають в себе назву операції, поле для введення IP сервера для пошуку, таблицю для відображення даних сервера та кнопку "Delete".

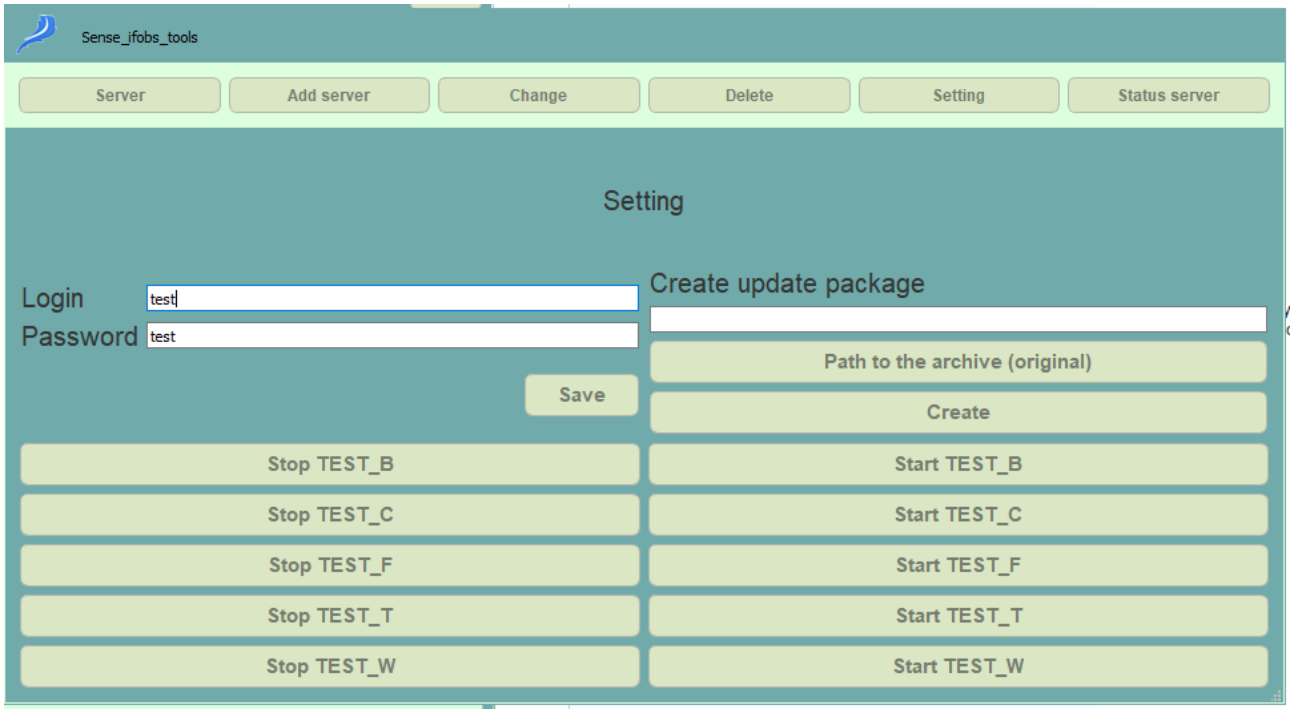


Рисунок 4.5 – Форма Setting

Структура елементів форми для сторінки налаштувань.

page_setting (QWidget)`:

Тип: QWidget

Дочірні елементи:

label_setting (QLabel)`:

Тип: QLabel

Текст: “Setting"

Вирівнювання тексту: Qt::AlignCenter

horizontalLayout_18 (QHBoxLayout)`:

Тип: QHBoxLayout

Містить два дочірніх лейаути (verticalLayout_19 та verticalLayout_14).

verticalLayout_19 (QVBoxLayout)`:

Тип: QVBoxLayout

Містить елементи для введення логіну та паролю.

horizontalLayout_16:

Лейаут горизонтальний.

Містить два вкладені лейаути (verticalLayout_15 та verticalLayout_16).

label_login (QLabel)`:

Тип: QLabel

Текст: "Login"

label_pas (QLabel)`:

Тип: QLabel

Текст: "Password"

lineEdit_login (QLineEdit)`:

Тип: QLineEdit

lineEdit_pas (QLineEdit)`:

Тип: QLineEdit

horizontalLayout_15:

Лейаут горизонтальний.

Містить signal_setting (QLabel), горизонтальний пробіл (horizontalSpacer_2), та pushButton_save (QPushButton).

signal_setting (QLabel)`:

Тип: QLabel

Мінімальний розмір: ширина 150, висота 0

Текст: порожній рядок

horizontalSpacer_2 (QSpacerItem)`:

Тип: QSpacerItem

Орієнтація: Qt::Horizontal

Розміри: ширина 40, висота 20

pushButton_save (QPushButton)`:

Тип: QPushButton

Текст: "Save"

verticalLayout_14 (QVBoxLayout)`:

Тип: QVBoxLayout

Містить елементи для створення оновлення пакету.

label_3 (QLabel)`:

Тип: QLabel

Текст: "Create update package"

lineEdit (QLineEdit)`:

Тип: QLineEdit

pushButton (QPushButton)`:

Тип: QPushButton

Текст: "Path to the archive (original)"

pushButton_2 (QPushButton)`:

Тип: QPushButton

Текст: "Create"

horizontalLayout_17 (QHBoxLayout)`:

Тип: QHBoxLayout

Містить два вкладені вертикальні лейаути (verticalLayout_17 та verticalLayout_18).

verticalLayout_17 (QVBoxLayout)`:

Тип: QVBoxLayout

Містить кнопки для зупинки тестувань.

pushButton_stop_b,

pushButton_stop_c,

pushButton_stop_f,

pushButton_stop_t,

pushButton_stop_w.

verticalLayout_18 (QVBoxLayout)`:

Тип: QVBoxLayout

Містить кнопки для запуску тестувань.

pushButton_start_b,

pushButton_start_c,

pushButton_start_f,

pushButton_start_t,

pushButton_start_w.

Описані елементи визначає розмітку та організацію елементів для сторінки налаштувань, де користувач може вказати логін та пароль, шлях для створення оновлення пакету, а також керувати тестуванням.

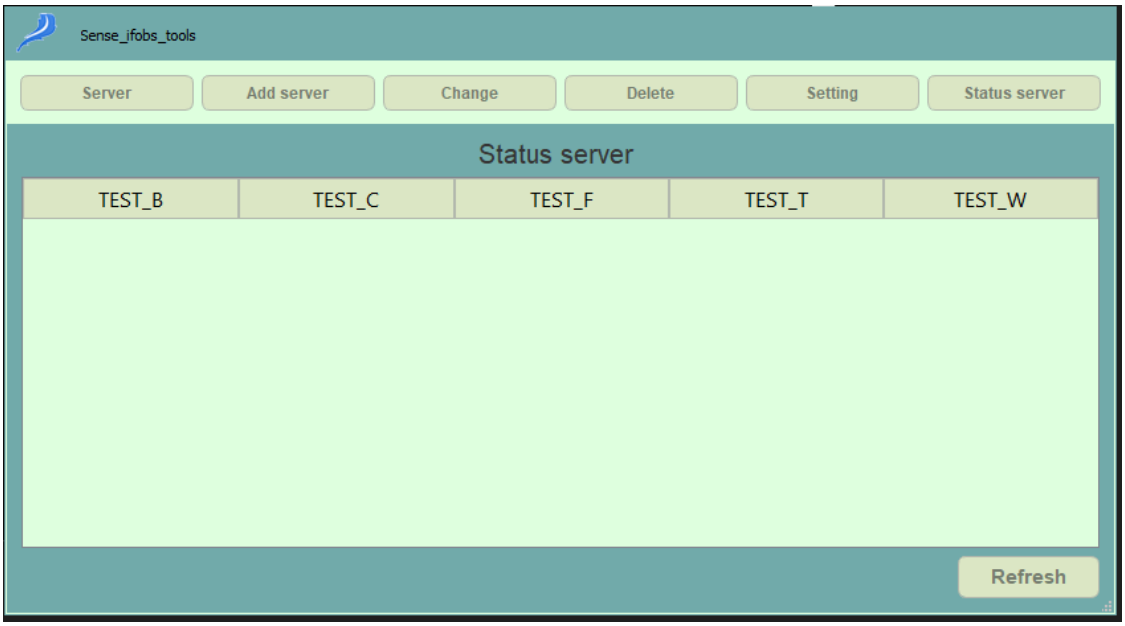


Рисунок 4.6 – Форма Status server

Структура елементів форми для сторінки з інформацією про статус серверів. page_status (QWidget)`:

Тип: QWidget

Дочірні елементи:

label_status_title (QLabel)`:

Тип: QLabel

Текст: "Status server"

Вирівнювання тексту: Qt::AlignCenter

table_status (QTableWidget)`:

Тип: QTableWidget

Стилізація:

Шрифт: розмір 12

Мінімальний розмір секції горизонтального заголовка: 150

Значення розмірів секцій горизонтального і вертикального заголовків

Властивості відображення сортування та розтягування останньої секції горизонтального заголовка

Колонки таблиці:

“TEST_B"

"TEST_C"

"TEST_F"

"TEST_T"

"TEST_W"

horizontalLayout_2 (QHBoxLayout)`:

Тип: QHBoxLayout

Містить мітку (label_status), горизонтальний роздільник (horizontalSpacer_status), і кнопку "Refresh" (pushButton_status)

label_status (QLabel)`:

Тип: QLabel

Мінімальний розмір: ширина 120

Текст: порожній рядок

horizontalSpacer_status (QSpacerItem)`:

Тип: QSpacerItem

Орієнтація: Qt::Horizontal

Розміри: ширина 40, висота 20

pushButton_status (QPushButton)`:

Тип: QPushButton

Текст: "Refresh"

4.3 Реалізація механізмів комунікації з серверами

Архітектура взаємодії між серверами та системою керування грає критичну роль у забезпеченні ефективності, масштабованості та надійності системи. Вона включає в себе розташування та організацію серверів, механізми комунікації та системи керування цими ресурсами. Нижче наведено загальний опис такої архітектури:

Комунікація між серверами:

Внутрішня мережа:

Сервери можуть спілкуватися між собою через внутрішню мережу для обміну даними та виконання різних завдань.

Механізми комунікації:

Використання стандартних протоколів комунікації, таких як HTTP/HTTPS, MQTT, або власні протоколи для взаємодії між серверами.

Перелік створених функцій на Python, який використовує бібліотеку Paramiko для здійснення з'єднання з віддаленим сервером через SSH та виконання команд на цьому сервері.

Фрагмент коду, а саме функцій якими виконується зупинка та запуск процесів на серверах:

```
def remote_connect(host, user, pwd):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    logMain.info('connect ' + user+' to –'+host)
    try:
```

```
            ssh_connection = ssh.connect(host, username=user, password=pwd,
timeout=100)
        except paramiko.AuthenticationException:
            print "Auth failed"
            logMain.error(u'Auth failed')
            ssh.close()
            sys.exit(0)
        return ssh
```

–remote_connect(host, user, pwd):

- Призначення: Функція для встановлення SSH-з'єднання з віддаленим сервером.
- Параметри:
  - host (рядок): IP-адреса або ім'я хоста віддаленого сервера.
  - user (рядок): Ім'я користувача для входу на віддалений сервер.
  - pwd (рядок): Пароль для входу на віддалений сервер.
- Повертає: Об'єкт SSH-з'єднання, який можна використовувати для відправки команд.

```
def remote_cmd(tnl, command):
    logMain.info('send command: ' + command)
    (stdin, stdout, stderr) = tnl.exec_command(command)
    out = stdout.read().decode("utf-8")
    error = stderr.read().decode("utf-8")
    if error:
        logMain.error(error)
        print re.sub(u"(\u2018|\u2019)", "",error)
    else:
        logMain.info(out)
    print (out.encode('utf-8'))
```

- remote_cmd(tnl, command):
    - Призначення: Функція для відправлення команди на віддалений сервер і отримання результату виконання.
    - Параметри:
        - tnl (об'єкт SSH): Об'єкт SSH-з'єднання, отриманий з remote_connect.
        - command (рядок): Команда, яку потрібно виконати на віддаленому сервері.

    Вивід: Виводить результат виконання команди на консоль.

    Логування: Записує інформацію про виконання команди у журнал (лог).

```
def remote_end(tnl):
    #logging.info('disconnect ' + user+' from –'+host)
    tnl.close()
```

- remote_end(tnl):
    - Призначення: Функція для завершення SSH-з'єднання з віддаленим сервером.
    - Параметри:
        - tnl (об'єкт SSH): Об'єкт SSH-з'єднання, отриманий з remote_connect.

Логування: Закриває SSH-з'єднання та реєструє інформацію про завершення з'єднання у журнал (лог).

Створений клас з описаними функціями дозволяє вам взаємодіяти з віддаленим сервером через SSH, надсилаючи команди та отримуючи результати їх виконання.

На рисунку 4.7 видно, як автоматизація працює. Система керування процесами на сервері відправляє необхідні команди одночасно. Це призводить до значної економії часу.

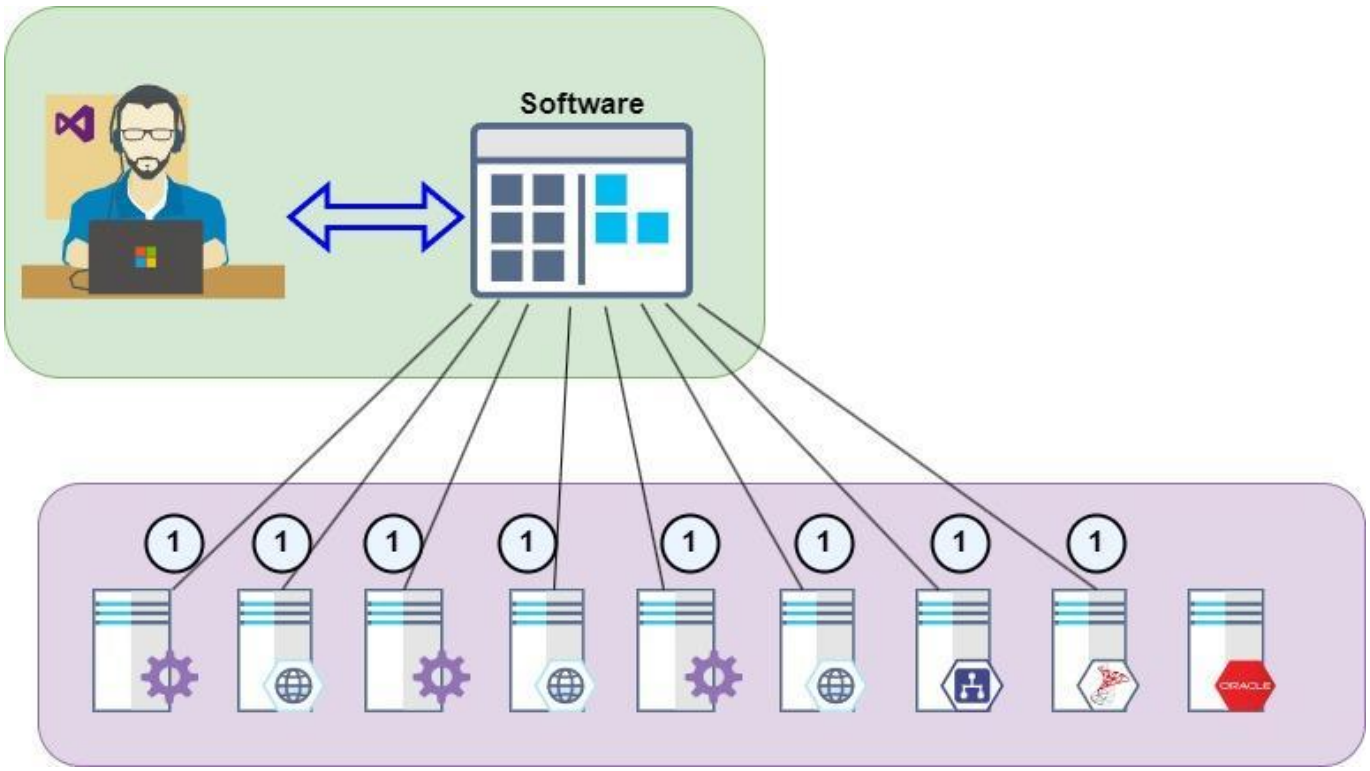


Рисунок 4.7 – Керування серверами в паралелі

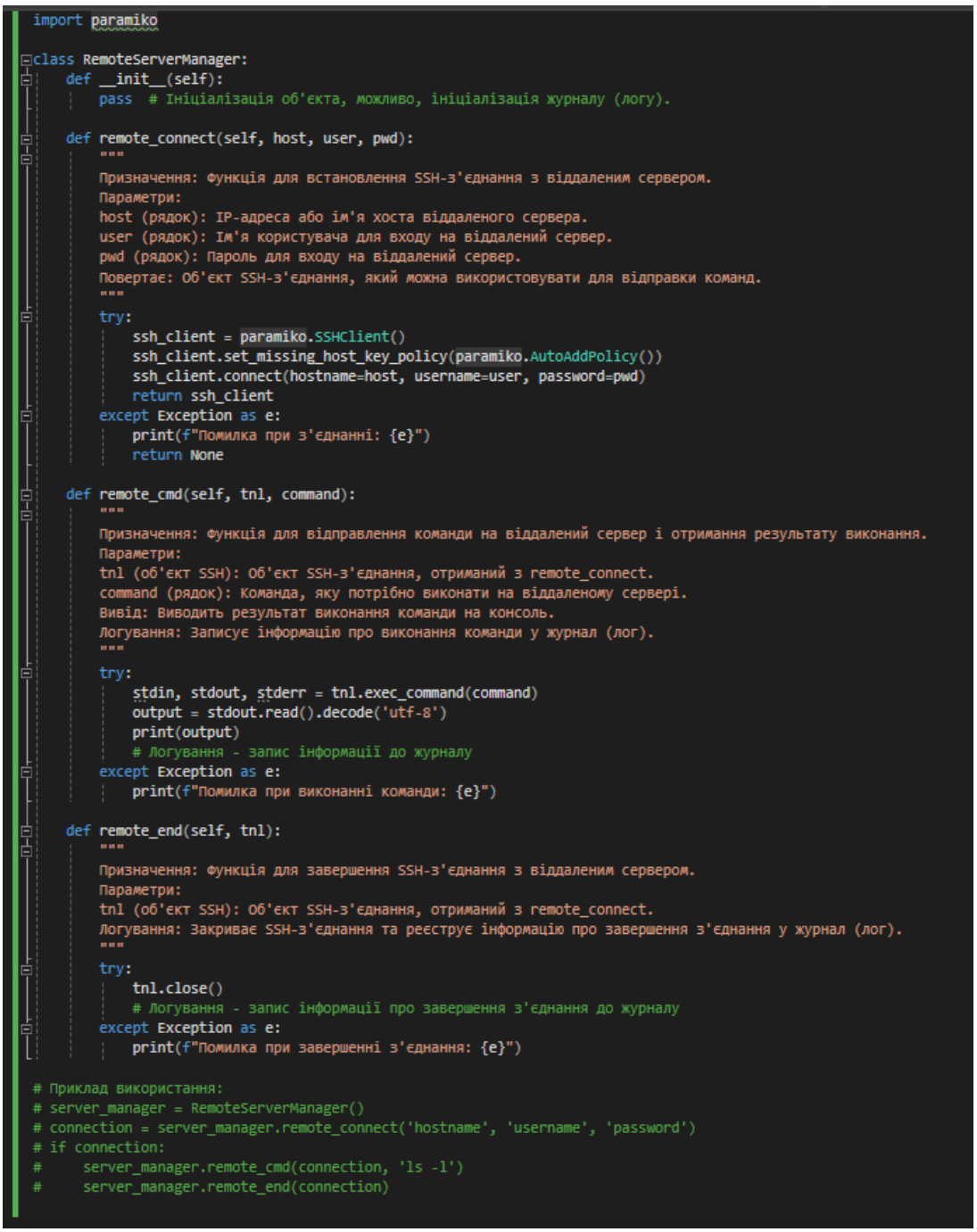
4.4 Реалізація функціоналу для запуску, зупинення та моніторингу статусу java процесів

Ця структура системи (рис. 4.8) дозволяє нам здійснювати SSH-з'єднання з віддаленим сервером, відправляти команди та отримувати їх результати, а також закривати з'єднання. Логіка реалізована через бібліотеку paramiko для взаємодії з SSH-протоколом.

Paramiko – це бібліотека для мови програмування Python, яка надає інструменти для роботи з протоколом SSH (Secure Shell). Зокрема, Paramiko дозволяє вам встановлювати з'єднання SSH, взаємодіяти з віддаленими серверами, передавати файли та виконувати команди на віддалених машинах.

Основні можливості та компоненти Paramiko:

- SSHClient;
- Transport;
- SFTPClient;
- Key Handling.



```python
import paramiko

class RemoteServerManager:
    def __init__(self):
        pass  # Ініціалізація об'єкта, можливо, ініціалізація журналу (логу).

    def remote_connect(self, host, user, pwd):
        """
        Призначення: Функція для встановлення SSH-з'єднання з віддаленим сервером.
        Параметри:
        host (рядок): IP-адреса або ім'я хоста віддаленого сервера.
        user (рядок): Ім'я користувача для входу на віддалений сервер.
        pwd (рядок): Пароль для входу на віддалений сервер.
        Повертає: Об'єкт SSH-з'єднання, який можна використовувати для відправки команд.
        """
        try:
            ssh_client = paramiko.SSHClient()
            ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
            ssh_client.connect(hostname=host, username=user, password=pwd)
            return ssh_client
        except Exception as e:
            print(f"Помилка при з'єднанні: {e}")
            return None

    def remote_cmd(self, tnl, command):
        """
        Призначення: Функція для відправлення команди на віддалений сервер і отримання результату виконання.
        Параметри:
        tnl (об'єкт SSH): Об'єкт SSH-з'єднання, отриманий з remote_connect.
        command (рядок): Команда, яку потрібно виконати на віддаленому сервері.
        Вивід: Виводить результат виконання команди на консоль.
        Логування: Записує інформацію про виконання команди у журнал (лог).
        """
        try:
            stdin, stdout, stderr = tnl.exec_command(command)
            output = stdout.read().decode('utf-8')
            print(output)
            # Логування - запис інформації до журналу
        except Exception as e:
            print(f"Помилка при виконанні команди: {e}")

    def remote_end(self, tnl):
        """
        Призначення: Функція для завершення SSH-з'єднання з віддаленим сервером.
        Параметри:
        tnl (об'єкт SSH): Об'єкт SSH-з'єднання, отриманий з remote_connect.
        Логування: Закриває SSH-з'єднання та реєструє інформацію про завершення з'єднання у журнал (лог).
        """
        try:
            tnl.close()
            # Логування - запис інформації про завершення з'єднання до журналу
        except Exception as e:
            print(f"Помилка при завершенні з'єднання: {e}")

# Приклад використання:
# server_manager = RemoteServerManager()
# connection = server_manager.remote_connect('hostname', 'username', 'password')
# if connection:
#     server_manager.remote_cmd(connection, 'ls -l')
#     server_manager.remote_end(connection)
```

Рисунок 4.8 – Структура системи

Загалом, Paramiko є потужною бібліотекою для взаємодії з SSH-протоколом в Python і є важливим інструментом для автоматизації взаємодії з віддаленими серверами та передачі файлів через безпечне з'єднання.

Ця структура (рис. 4.9) описує діаграму, яка представляє систему керування тестовими серверами, де є основні компоненти цієї системи включають в себе: Workplace.

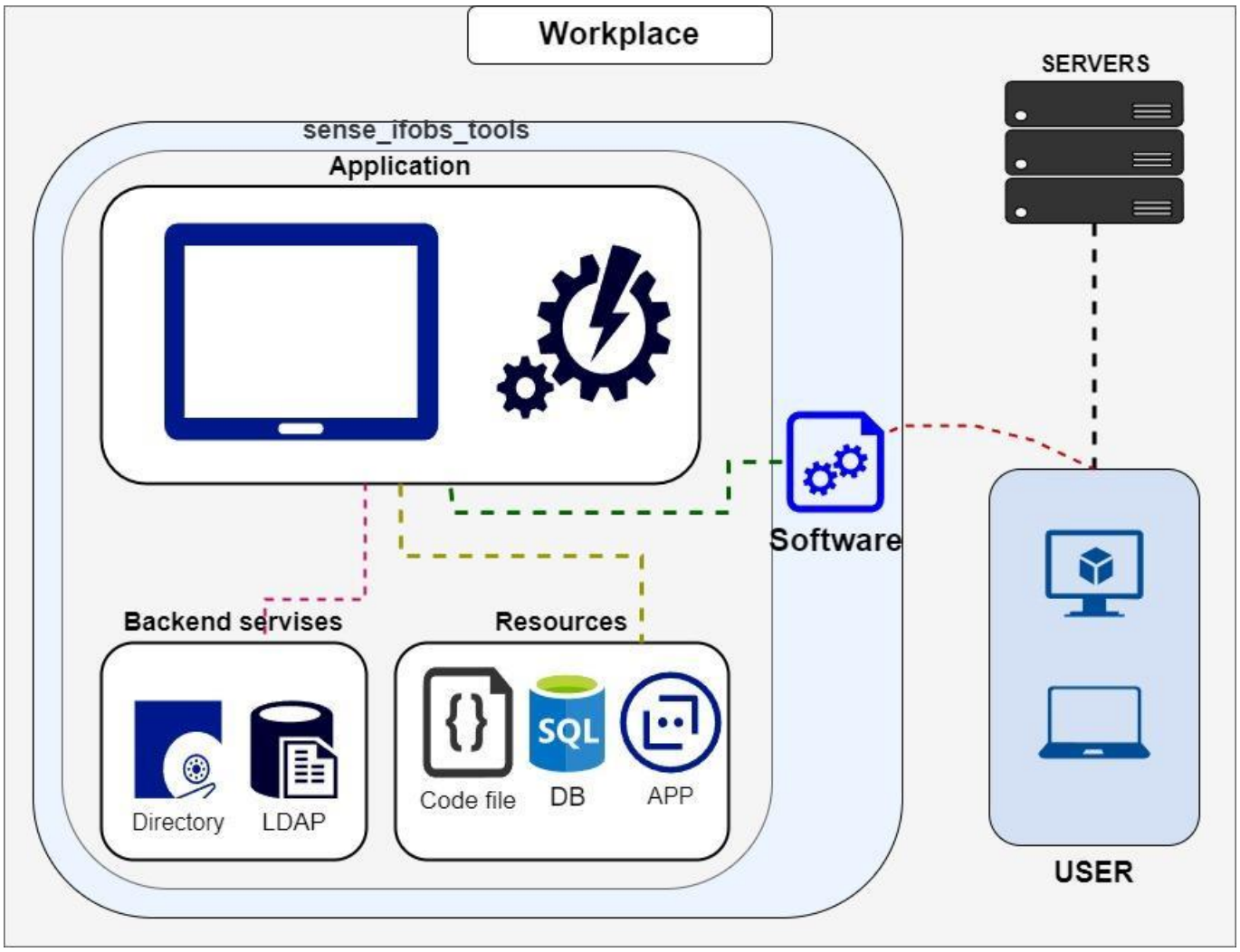


Рисунок 4.9 – Структура системи керування тестовими серверами

Робоче місце, яка, представляє загальний простір для взаємодії тестових серверів.

USER:

Представлення користувача, тестувальника чи адміністратора системи.

sense_ifobs_tools:

Назва програмного забезпечення.

Software:

Системне програмне забезпечення з набором іншими програмними компонентами.

Application:

Додаток, який керує та моніторить процеси на тестових серверах.

SERVERS:

Представлення тестових серверів.

Структура бази даних системи (рис. 4.10)

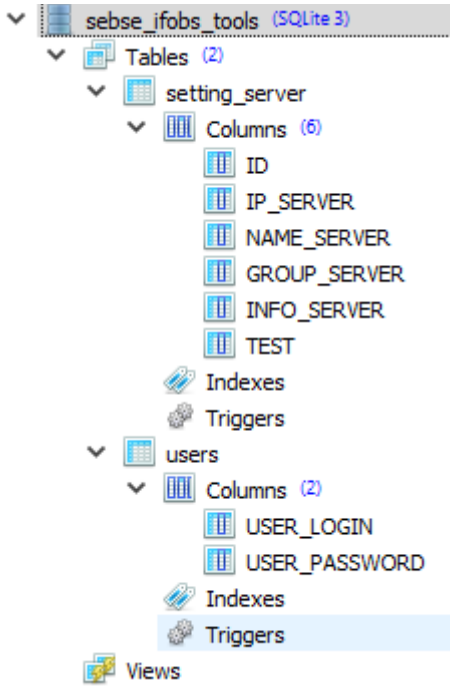


Рисунок 4.10 – Структура бази даних системи

Налаштування системи можливо виконати двума способами.

1. Через інтерфейс системи
2. Напряму в БД

Масштабування

Реалізована можливість додавати нові тестові стенди або видаляти які не використовуються.

Модуль з класом та набором функцій для підключення та керування даними в базі даних.

```
import sqlite3
class BaseInteraction ():
    def __init__(self):
        self.connection = sqlite3.connect('sebse_ifobs_tools.db')
    def
insert_server(self,IP_SERVER,NAME_SERVER,GROUP_SERVER,INFO_SERVER,TEST):
```

```
    def m_server(self):
    def find_server(self,ip_server):
    def del_server(self,ip_server):
   def
update_server(self,Id,IP_SERVER,NAME_SERVER,GROUP_SERVER,INFO_SERVER
,TEST):
    def find_server_status(self,test_name,sId):
    def find_test_name2(self,test_name):
    def find_test_name(self,test_name):
    def find_login_pass(self,login):
    def insert_login_pass(self,u_login,u_pass):
```

Висновки до розділу 4

У цьому розділі подано вичерпний погляд на процес реалізації системи, починаючи від важливого вибору технологій та інструментів. Детальний аналіз функціональних та нефункціональних вимог дав змогу обґрунтувати вибір мови програмування, приділяючи увагу зручності розробки, широкій підтримці бібліотек, загальній популярності та розширюваності.

Мова програмування Python обрана для функціоналу керування тестовими стендами з урахуванням її читабельності, простоти синтаксису та широкої спільноти розробників. Вибір Cascading Style Sheets (CSS) для оформлення дизайну визначено через його розширену підтримку браузерів, легкість використання та навчання, можливості адаптивного дизайну та інтеграції з HTML.Для управління базою даних обрано комбінацію систем Oracle та SQLite, що дозволяє оптимально використовувати ресурси залежно від потреб проекту. Середовище розробки обрано Visual Studio, з урахуванням його інтегрованого середовища розробки, інструментів відлагодження та профілювання, підтримки командного рядка та версійного контролю.

Особлива увага приділена архітектурі взаємодії серверів та системи керування, яка включає в себе внутрішню мережу для обміну даними та використання стандартних протоколів комунікації, таких як HTTP/HTTPS, MQTT.

Представлений перелік функцій на Python з використанням бібліотеки Paramiko дозволяє ефективно керувати віддаленими серверами через SSH, що важливо для виконання різноманітних завдань та обміну даними. Клас, описаний в тексті, надає зручний інтерфейс для здійснення з'єднання, відправлення команд та отримання результатів.

Далі, структура системи керування тестовими серверами інформативно представлена діаграмою, де основні компоненти включають робоче місце, користувача, програмне забезпечення, додаток та тестові сервери. Ця структура відображає загальний простір для взаємодії та контролю серверів.

Окремо висвітлено структуру бази даних системи з діаграмою класів, що вказує на ключові компоненти та їх взаємозв'язки. Підкреслено можливість налаштування системи через інтерфейс або безпосередньо в базі даних, що робить її більш гнучкою та налаштованою під потреби користувача.

Зазначено, що система має вбудовану можливість масштабування, дозволяючи додавати нові тестові стенди та видаляти ті, які не використовуються. Це важливий аспект для забезпечення ефективності та оптимального використання ресурсів при збільшенні обсягів роботи системи.

Особлива увага приділена масштабуванню, де розглянуті вертикальне та горизонтальне масштабування, щоб забезпечити ефективну роботу системи з ростом обсягів даних та навантаження.Загалом, вибір технологій та інструментів відбувався з урахуванням вимог проекту, спрямованим на покращення продуктивності, надійності та ефективності системи управління тестовими стендами.У результаті, розділ надає повний обзор реалізації механізмів комунікації, системної структури та бази даних, підкреслюючи гнучкість та ефективність взаємодії і керування тестовими серверами.

5 ТЕСТУВАННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ

5.1 Тестування системи

У даному розділі проведено тестування розробленої системи автоматизованого керування тестовими серверами з метою визначення її функціональності та надійності. Метою тестування є перевірка правильності роботи та взаємодії модулів системи, реалізованої в програмі. Тестування включає два аспекти: правильність роботи окремих модулів та взаємодію між цими модулями.

5.2 Оцінка продуктивності та ефективності системи

Для тестування системи використовувалися наступні методи та підходи.

Модульне тестування.

Кожен модуль системи був протестований окремо для перевірки правильності їх роботи та взаємодії між собою.

Тестування функціональності додавання сервера.

Задання: Введення коректних даних сервера.

Дії:

Запуск програми.

Введення IP, імені, групи, інформації та тестових даних сервера.

Натискання кнопки "Додати".

Очікуваний результат: Сервер додається до системи, та виводиться повідомлення про успішне додавання.

Тестування функціональності оновлення інформації про сервер

Задання: Введення коректних даних для оновлення інформації про сервер.

Дії:

Запуск програми.

Введення ID сервера для оновлення.

Зміна IP, імені, групи, інформації та тестових даних сервера.

Натискання кнопки "Оновити".

Очікуваний результат: Інформація про сервер оновлюється, та виводиться повідомлення про успішне оновлення.

Інтеграційне тестування:

Проводилися тести, що перевіряли правильність взаємодії між різними компонентами системи.

Тестування переходу між сторінками за допомогою верхніх кнопок

Задання: Клік на верхні кнопки навігації.

Дії:

Запуск програми.

Натискання кнопок "Сервер", "Додати", "Змінити", "Видалити", "Налаштування", "Статус".

Очікуваний результат: Перехід на відповідну сторінку.

Тестування відображення інформації на головній сторінці

Завдання: Додавання сервера та перевірка його відображення на головній сторінці.

Дії:

Додавання сервера через форму.

Перехід на головну сторінку.

Очікуваний результат: Інформація про доданий сервер відображається на головній сторінці.

Тестування коректності роботи системи під час запуску

Задання: Запуск програми та перевірка наявності несправностей під час запуску.

Дії:

Запуск програми.

Очікуваний результат: Програма запускається без помилок, інтерфейс коректно відображається.

Тестування функціоналу закриття та максимізації вікна

Задання: Використання кнопок згортання, розгортання та закриття вікна.

Дії:

Запуск програми.

Натискання кнопок мінімізації, максимізації та закриття.

Очікуваний результат: Вікно програми коректно мінімізується, максимізується та закривається.

Системне тестування:

Системне тестування є важливою фазою в життєвому циклі розробки програмного забезпечення. Цей тип тестування фокусується на перевірці та переконанні в правильній роботі всіх компонентів системи як єдиної сутності, щоб гарантувати, що система в цілому відповідає визначеним вимогам та функціональності.

Система тестувалася як єдина сутність для переконання у правильній роботі всіх компонентів в їхньому комплексі.

Валідація функціональності: Перевірка, чи виконує кожен компонент системи свої функціональні обов'язки вірно та ефективно.

Інтеграція компонентів: Впевнений, що всі компоненти взаємодіють один з одним вірно та не порушують взаємодії між собою.

Верифікація відповідності вимогам: Переконання, що система відповідає всім визначеним вимогам, включаючи функціональні та нефункціональні аспекти.

Виявлення дефектів: Забезпечення виявлення та виправлення дефектів чи неполадок, які можуть виникнути при роботі системи в цілому.

Оцінка продуктивності: Перевірка ефективності та продуктивності системи під час виконання реальних завдань та обсягів роботи.

Підготовка тестового середовища: Створення і налаштування середовища, аналогічного реальному виробничому середовищу.

Розробка тестових сценаріїв: Створення набору тестових сценаріїв, які охоплюють всі можливі шляхи взаємодії користувачів з системою.

Виконання тестів: Запуск тестів відповідно до розроблених сценаріїв та реєстрація результатів.

Виправлення дефектів: Якщо тестові результати показують виявлені дефекти, їх фіксація та подальше тестування для перевірки виправлення.

Оцінка продуктивності: Вимірювання та оцінка роботи системи під різними навантаженнями та умовами.

Документація результатів: Формування та збереження звіту з результатами тестування, який включає виявлені дефекти та рекомендації.

Завершення тестування

Тестування успішно завершено. Помічено відсутність критичних помилок та збоїв у роботі програми. Тестування включає перевірку правильності роботи окремих модулів та їх взаємодії, а також загальне тестування функціоналу та інтерфейсу програми.

5.3 Виправлення помилок та покращення системи

Під час тестування виявлені наступні помилки та пропозиції щодо покращення системи:

Помилки:

1. На формах не відображаються іконки кнопок: закрити, згорнути, розгорнути. Але, при наведенні курсором, на місце де вони повинні відображатися –система реагує(змінюється колір). Та при натисканні –свою функцію виконують.
2. На формі “Status server”, повільно відпрацьовує перевірка статусу Java процесу.

Покращення:

1. Необхідно внести зміни в дизайн програми, та виконати оновлення.
2. Внести додаткові перевірки, на коректність заповнення даних про сервер, при додаванні та редагуванні.

Виправлення помилок та впровадження покращень планується у наступних версіях системи для її подальшого удосконалення.

Висновки до розділу 5

У даному розділі виконано комплексне тестування розробленої системи автоматизованого керування тестовими серверами з акцентом на перевірку функціональності та надійності. Результати тестування вказують на високу якість

роботи окремих модулів та їх взаємодії, підтверджуючи правильність реалізації програми.

Методи та підходи, такі як модульне тестування, інтеграційне тестування та системне тестування, дозволили перевірити різні аспекти системи, від окремих функцій до загальної взаємодії її компонентів. Тестування функціональності додавання та оновлення інформації про сервер, інтеграційне тестування переходів між сторінками та відображення інформації на головній сторінці показали правильність реалізації ключових функціональних елементів.

Під час тестування були виявлені окремі помилки, такі як відсутність іконок кнопок та повільна реакція на перевірку статусу Java процесу. Проте, узагальнений висновок тестування вказує на відсутність критичних помилок та збоїв у роботі програми.

У розділі також надано конкретні рекомендації щодо виправлення виявлених помилок та покращення системи. Проектування змін в дизайні програми, додаткові перевірки коректності введених даних та виправлення інших помилок є частиною подальшого плану розвитку системи.

Таким чином, результати тестування підтверджують готовність системи до подальшого використання та вказують на її потенціал для вдосконалення та розвитку в майбутньому.

6 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

6.1 Опис ідеї проекту

Спільно з інтенсивним ростом розробки програмного забезпечення, з'являється необхідність вдосконалення процесу тестування, щоб забезпечити якість та надійність продуктів. У цьому контексті стартап-проєкт "Система автоматизованого керування тестовими серверами" прагне вирішити цю проблему шляхом розробки і впровадження системи, яка забезпечить ефективне управління тестовими серверами. Опис його ідеї в табл. 6.1.

Таблиця 6.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка системи автоматизованого керування тестовими серверами	1. Оптимізація процесу тестування програмного забезпечення. 2. Підвищення продуктивності розробки.	1. Зниження часу та зусиль, потрібних для управління тестовими серверами. 2. Підвищення ефективності та точності тестування.

Опис стартапу:

Sense iFOBS Tools є передовим інноваційним стартапом, що зосереджений на вдосконаленні та розробці системи автоматизованого керування тестовими серверами для підвищення ефективності тестування програмного забезпечення (ПЗ).

Місія:

Місія Sense iFOBS Tools полягає в створенні інтегрованої та інтуїтивно зрозумілої системи, яка значно полегшить та прискорить процес тестування, допомагаючи розробникам та тестувальникам досягти вищої якості ПЗ.

Основні характеристики:

Автоматизація процесу тестування: Sense iFOBS Tools пропонує рішення для автоматизації процесу керування тестовими серверами, що дозволяє виконувати тести швидше та ефективніше.

Гнучкість та адаптивність: Система розробляється з урахуванням потреб різних виробничих середовищ, надаючи гнучкість для легкої інтеграції та адаптації до різних вимог.

Візуалізація результатів: Sense iFOBS Tools забезпечує зручний інтерфейс для візуалізації результатів тестування, що дозволяє швидко виявляти та вирішувати проблеми.

Аналіз та оптимізація продуктивності: Система дозволяє проводити аналіз результатів тестування та використовувати дані для постійної оптимізації процесів керування тестовими серверами.

Переваги Sense iFOBS Tools:

Швидше впровадження продукту: Завдяки автоматизації тестування, розробники можуть швидше впроваджувати високоякісне програмне забезпечення на ринок.

Зменшення витрат: Система дозволяє зменшити час тестування, що в свою чергу призводить до економії ресурсів та коштів.

Підвищення якості продукту: Забезпечення надійного та ефективного тестування сприяє підвищенню якості програмного забезпечення.

В табл. 6.2 визначено характеристики проекту.

Таблиця 6.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
1	Точність прогнозування	+ + + + +	-	
2	Доступність ресурсу	+	- —+	
3	Безпека	+ + + + +	-	

№	Техніко-економічні характеристики ідеї	W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
4	Можливість прогнозу на будь-яку дату	+	- —+	
5	Можливість інтеграції з іншими системами	+ +	- –	+
6	Доступність системи з будь-якого пристрою	- + + +		+

Очікувані вигоди для клієнтів:

Збільшення продуктивності тестового середовища:

Припустимо, що до впровадження Sense iFOBS Tools час, необхідний для виконання тестів, становив 10 годин, а кількість тестів за один цикл складала 100. Після впровадження системи час виконання тестів може зменшитися на 30%, що означає:

Початковий час виконання тестів: 10 годин * 100 тестів = 1000 годин.

Час виконання тестів після впровадження: 0.7 * 10 годин * 100 тестів = 700 годин.

Економія часу: 1000 годин –700 годин = 300 годин.

Вигода для клієнта: Збільшення продуктивності тестового середовища на 30%, що дозволяє зекономити 300 годин робочого часу.

2. Зниження ризику помилок та виявлення дефектів на ранніх етапах розробки:

Припустимо, що до впровадження Sense iFOBS Tools кількість виявлених дефектів на ранніх етапах розробки становила 20%, а після впровадження ця цифра зросте до 35%.

Початкова кількість виявлених дефектів: 100 дефектів * 20% = 20 дефектів.

Кількість виявлених дефектів після впровадження: 100 дефектів * 35% = 35 дефектів.

Збільшення кількості виявлених дефектів: 35 дефектів –20 дефектів = 15 дефектів.

Вигода для клієнта: Збільшення кількості виявлених дефектів на 15, що дозволяє уникнути їх поширення в подальших етапах розробки та підвищує якість продукту.

3. Зменшення витрат на тестування та підвищення ефективності робочих процесів:

Припустимо, що до впровадження Sense iFOBS Tools витрати на тестування складали $50,000 на місяць. Після впровадження витрати можуть зменшитися на 20%.

Початкові витрати на тестування: $50,000.

Витрати на тестування після впровадження: 0.8 * $50,000 = $40,000.

Зменшення витрат: $50,000 –$40,000 = $10,000.

Вигода для клієнта: Зменшення витрат на тестування на $10,000, що призводить до економії ресурсів компанії.

Ці розрахунки надають загальний уявлення про можливі вигоди, які може отримати клієнт від використання Sense iFOBS Tools.

Завершення:

Sense iFOBS Tools розвивається, щоб стати ключовим гравцем в сфері автоматизованого керування тестовими серверами, спрощуючи життя розробників та забезпечуючи високу якість та надійність програмного забезпечення.

6.2 Оцінка можливостей на ринку для запуску стартап-проекту

В табл. 6.3–6.12 проведено оцінку можливості запуску стартап–проекту.

Таблиця 6.3 –Аналіз ринкових можливостей запуску стартап-проекту

№	Покази стану ринку	Характеристика
1	Кількість гравців на ринку,од	~ 3-4

№	Покази стану ринку	Характеристика
2	Загальні обсяги продаж, грн.	~ 80млн
3	Динаміка ринку	зростає
4	Наявність обмежень для входу	немає
5	Специфічні вимоги до стандартизації та сертифікації	немає
6	Середня норма рентабельності в галузі (або по ринку), %	~23-31

Таблиця 6.4 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Керувати процесами на великій кількості серверів	Банківські установи, відділ адміністрування систем	Перенесення керування серверними процесами в одне програмне забезпечення.	Автоматизація в керуванні однотипними процесами серверів, та зменшити час на керування процесами.

Таблиця 6.5 – Аналіз технологій для реалізації проекту

Ідея проекту	Технології реалізації	Наявність технології	Доступність технології
Збереження даних	SQLite, Oracle	Доступні для загального використання	Доступні без оплати
Авторизація	Active Directory (AD)	Доступні для загального використання	Доступні без оплати
Засоби розробки програмного забезпечення	Visual Studio, PLSQL Developer	Доступні для загального використання	Доступні без оплати

Таблиця 6.6 – Фактори загроз

Фактор	Зміст загрози	Можлива реакція компанії
Клієнти віддають перевагу конкурентам	На ринку існують інші компанії-конкуренти, які, хоча не надають повний набір необхідних функцій, але є великою мірою популярнішими.	Ефективне взаємодіє з клієнтами на кожному етапі розробки та впровадження функціоналу, що відповідає їхнім бізнес-процесам.
Відсутність WEB версії	Наявність WEB версії розширює можливості доступності до автоматизованого керування тестовими	Розробка WEB -кліента.

Фактор	Зміст загрози	Можлива реакція компанії
	серверами.	

Таблиця 6.7 – Фактори можливостей

Фактор	Зміст можливості	Можливі реакції компанії
Відсутність безпосередніх конкурентів на ринку, які пропонують повний набір необхідних функцій для даної сфери застосування.	Конкурентам необхідна розробка функціоналу, для того щоб покрити вимоги.	Активно працювати з банківськими установами, та презентувати перспективи продукту.

Таблиця 6.8 – Аналіз конкуренції в галузі за Майклом Портером

	Прямі конкуренти	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Компоненти аналізу	Відсутні	Puppet, SaltStack, Chef, Ansible		Відділ адміністрування та керування серверами	людські витрати часу
Висновки	У зв'язку з відсутністю прямих конкурентів, характер боротьби не є інтенсивним.	Необхідна доробка	Залежно від потреб, можуть відбуватися зміни умов.	Замовник диктує умови.	Постачання спеціалізованого рішення для конкретної сфери застосування.

Таблиця 6.9 – Аргументація факторів, що впливають на конкурентоспроможність

Фактор конкурентоспроможності	Обґрунтування
Легкість доступу та зручність використання	Урахування відгуків клієнтів під час розробки та подальших оновлень системи
Інноваційність	Вирішення, що забезпечують високий рівень конкурентної переваги
Фокус на потребах клієнтів	Урахування відгуків клієнтів під час розробки та подальших оновлень системи
Комплексність функціоналу	Містить повний набір функціоналу для автоматичного керування тестовими сервера, з можливостями розширення згідно потреб клієнта

Таблиця 6.10 – Порівняльний аналіз сильних та слабких сторін проекту

Фактор конкурентоспроможності	Бали 1-20	Оцінка товарів конкурентів в порівнянні.						
		-3	-2	-1	0	+1	+2	+3
Легкість доступу та зручність використання	19	+						
Інноваційність	18			+				
Фокус на потребах клієнтів	16				+			
Комплексність функціоналу	14		+					

Таблиця 6.11 – SWOT–аналіз стартап-проекту

Сильні сторони: Урахування відгуків клієнтів під час розробки та подальших оновлень системи	Слабкі сторони: Урахування відгуків клієнтів під час розробки та подальших оновлень системи
Можливості: Відсутність безпосередніх конкурентів на ринку, які пропонують повний набір необхідних функцій для даної сфери застосування. Конкурентам необхідна розробка функціоналу, для того щоб покрити вимоги	Загрози: На ринку існують інші компанії-конкуренти, які, хоча не надають повний набір необхідних функцій, але є великою мірою популярнішими. Наявність WEB версії розширює можливості доступності до автоматизованого керування тестовими серверами

Таблиця 6.12 – Варіанти впровадження стартап-проекту на ринку

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Створення мінімально життєздатної продукції (MVP)	Висока	3-6 місяців
2	Створення продукту з повним функціоналом	Середня	6-8 місяців
3	Внесення змін доробки тощо	Середня	1-3 місяців

6.3 Створення стратегії введення на ринок для проекту

В табл. 6.13–6.20 визначено стратегію введення проекту на ринок.

Таблиця 6.13 –Визначення цільових груп потенційних клієнтів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Фінансові установи, банки	Високий	Середній	Середній	Середній

Ця таблиця детально аналізує цільову групу потенційних клієнтів, враховуючи готовність споживачів приймати продукт, орієнтований попит у цьому сегменті, інтенсивність конкуренції, а також простоту входу для нових учасників. Високий рівень готовності сприйняття продукту та середні показники попиту, конкуренції та входу вказують на потенційну привабливість цього сегменту для фінансових установ та банків.

Таблиця 6.14 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Простота в	Стратегія	Швидке введення	Автоматизація

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспромо жні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
налаштуванні	сфокусованого маркетингу	на ринок, повний функціонал в межах обраної сфери застосування	рутинних процесів

Таблиця 6.15 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Так	Пошук нових клієнтів	Так	Стратегія сфокусованого маркетингу

Таблиця 6.15 надає визначення базової стратегії конкурентної поведінки компанії. Проект вважається "першопрохідцем" на ринку, і компанія має намір активно шукати нових споживачів. Одночасно вони візьмуть на увагу основні характеристики товару конкурента та реалізують стратегію сфокусованого маркетингу, спрямовану на задоволення потреб цільової аудиторії.

Таблиця 6.16 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Автоматизоване керування процесами на тестових серверах	Стратегія сфокусованого маркетингу	Швидке введення на ринок, повний функціонал в межах обраної сфери застосування	Автоматизація рутинних процесів Швидкість налаштування Економія часу

Таблиця 6.17 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Програмне забезпечення з автоматизованого керування процесів на тестових серверах, для пришвидшення готовності тестового КБ до тестування		
II.Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1.Швидкість налаштування	Нм	Тх
	2.Многопоточность та точність алгоритму керування процесами	М	Тл
	3.Зручний та інтуїтивно зрозумілий інтерфейс.	М	Ор

Рівні товару	Сутність та складові		
	4. Власний підхід до моніторингу статусу процесів	Нм	Тл
III.Товар із підкріпленням	До продажу: Тренування користувачів та технічна підтримка		
	Після продажу: Отримання відгуків від користувачів та удосконалення системи		
За рахунок чого потенційний товар буде захищено від копіювання: Отримання патентів на унікальні алгоритми та постійне оновлення та вдосконалення програми			

М/Нм – монотонні/немонотонні; Вр/Тх/Тл/Е/Ор – вартісні/ технічні/ технологічні/ ергономічні/ органолептичні.

Таблиця 6.18 – Визначення меж встановлення ціни

Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
$500-1200\міс.	$500-1200\міс.	$150-350\міс.

Таблиця 6.18 визначає межі встановлення цін для продукту/послуги, враховуючи рівень цін на аналогічні товари, а також рівень доходів цільової групи споживачів. У даному контексті, верхня межа встановлення ціни становить $150 за місяц, щоб залишатися конкурентоспроможною порівняно з аналогічними продуктами, тоді як нижня межа встановлення ціни дорівнює $350 за місяц, забезпечуючи додатковий розрахунок для консервативного ринку.

Таблиця 6.19 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Установлення прямих контрактів з клієнтами	Створення посібника користувача та проведення навчання користувачів щодо використання системи	Користувач, – розробник продукту	Продажі, здійснювані компанією-розробником

Таблиця 6.20 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Формальне взаємодія з організація ми, укладання угод та інших	Кореспонденція, електронна пошта та телефон	Оригінальність системи, автоматизована система керування процесами на тестових	Демонструвати зручність використання, ефективність роботи, підвищення продуктивност	Виступ про продукт, відображення його можливостей

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
документів щодо використання продукту		серверах, та їх моніторинг.	і та якості обслуговування	

Висновки до розділу 6

В даному розділі проведено ретельний аналіз та розробку стартап-проєкту Sense iFOBS Tools, який спрямований на трансформацію процесу тестування програмного забезпечення. Місія проекту визначає його завдання у створенні інтегрованої та інтуїтивно зрозумілої системи для полегшення та прискорення тестування, сприяючи підвищенню якості програмного забезпечення.

Основні характеристики Sense iFOBS Tools, такі як автоматизація процесу тестування, гнучкість, візуалізація результатів та аналіз продуктивності, формують унікальний продукт, здатний швидко впроваджувати високоякісне програмне забезпечення, зменшуючи витрати та підвищуючи його якість.

Sense iFOBS Tools відзначається інноваційним рішенням, яке може визначити нові стандарти в галузі тестування, надаючи потужний інструмент для розробників і тестувальників. Його спроможність відсутність прямих конкурентів у поточний момент створює вільну нішу на ринку, але важливо підтримувати високий рівень якості та готовність до можливого з'явлення конкуренції у майбутньому.

ВИСНОВКИ

У підсумку проведення дослідницької роботи в області тестування програмного забезпечення можна визначити ключові висновки. Актуальність даного дослідження підтверджена реальною необхідністю у вирішенні проблем, пов'язаних із зростаючою складністю інформаційного середовища та розширенням обсягу розробки програмного забезпечення. В сучасному бізнес-середовищі інформаційні технології відіграють ключову роль, і ефективна автоматизація керування тестовими серверами стає стратегічно важливим завданням для оптимізації та підвищення продуктивності розробки.

Мета магістерської дисертації досягнута та підвищена ефективність тестування програмного забезпечення шляхом системи автоматизованого керування тестовими серверами. Вирішення проблем, пов'язаних із складнощами керування тестовими середовищами, зменшили витрати часу та ресурсів на цей процес.

Аналіз архітектури тестового середовища підкреслює стратегічне значення інформаційних технологій для підприємств, особливо в банківській галузі. Інвестиції в тестові сервери стають необхідним елементом високорівневого IT-розвитку та підтримки стабільної роботи бізнес-процесів.

Дослідження проблеми керування тестовими середовищами наголошує на важливості оптимізації та автоматизації процесів на серверах. Розрахунки часу недоступності тестового стенду свідчать про великий потенціал для зменшення витрат часу та ресурсів через впровадження автоматизованих процесів.

Узагальнюючи отримані результати, важливо підкреслити, що розробка системи автоматизованого керування тестовими серверами є актуальним та значущим завданням, спрямованим на підвищення продуктивності та ефективності управління тестовими стендами на підприємстві.

Наступні рекомендації можуть слугувати основою для подальшого розвитку та вдосконалення системи автоматизованого керування тестовими серверами:

- розширення функціоналу – рекомендується розгляд можливості розширення функціоналу системи автоматизованого керування тестовими серверами.

Додавання нових можливостей, таких як графічне відображення статусу серверів, відправлення сповіщень при збоях чи автоматизоване відновлення роботи серверів, може значно покращити ефективність системи;

- безпека та автентифікація – для покращення безпеки системи слід розглянути можливість впровадження механізмів аутентифікації та авторизації для користувачів, які мають доступ до системи. Це дозволить забезпечити конфіденційність та захист від несанкціонованого доступу;
- інтеграція з іншими IT–Системами – Дослідження можливості інтеграції системи з іншими IT–системами в організації. Наприклад, автоматичний обмін даними з системами моніторингу або іншими інструментами для кращого управління IT–інфраструктурою;
- тестування навантаження та масштабованість – проведення тестів на навантаження для оцінки працездатності системи під високим та інтенсивним завантаженням. Розгляд можливості оптимізації та підвищення масштабованості системи для забезпечення стабільної роботи під час зростання обсягу роботи;
- документація та інструкції – створення детальної документації, включаючи інструкції зі встановлення, налаштування та використання системи. Це допоможе користувачам та адміністраторам швидше ознайомитися з системою та використовувати її ефективно;
- тестування помилок та відновлення – наступним перспективним розвитком буде автоматизувати керування серверами на мікросервісний архітектуру.

Проведення тестування на виявлення помилок та розробка механізмів автоматичного відновлення системи в разі виявлення проблем. Це забезпечить більшу стабільність та надійність системи.

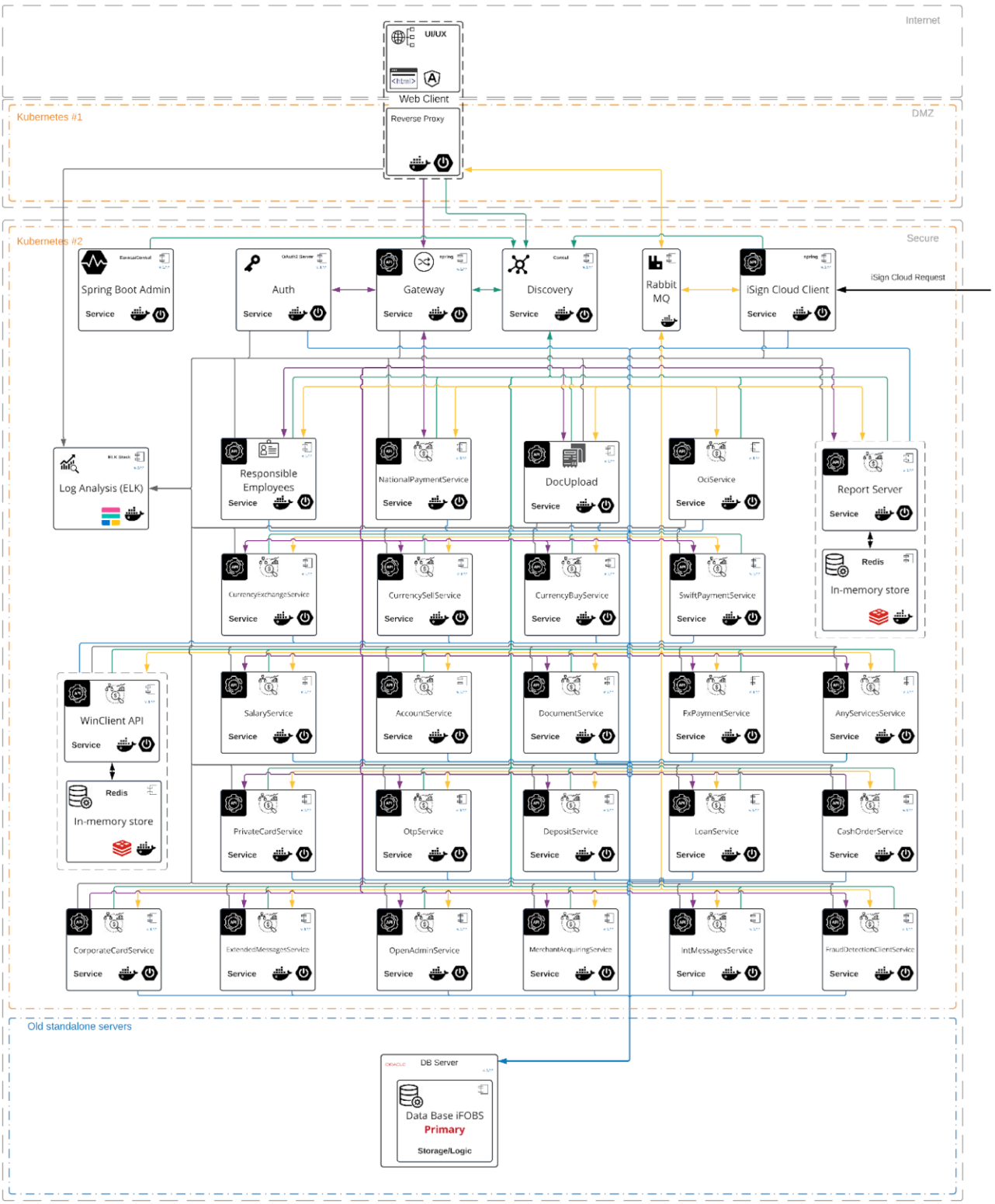


Рисунок 22 – Архітектурна схема мікросервісів

- моніторинг та звітність – впровадження засобів моніторингу для відстеження працездатності серверів, обсягів використання ресурсів та виявлення аномалій. Розробка засобів звітності для отримання детальної інформації про роботу системи.

- інтерфейс та зручність використання – здійснення оновлення та поліпшення інтерфейсу користувача для забезпечення зручності та інтуїтивної зрозумілості. Залучення користувацького фідбеку для подальшого вдосконалення інтерфейсу.

- тестування на різних операційних системах – перевірка сумісності та роботи системи на різних операційних системах. Це може включати операційні системи Windows, Linux та інші, які використовуються у вашій організації;

- оптимізація витрат ресурсів – аналіз та оптимізація витрат ресурсів системи для ефективного використання обчислювальної потужності та зменшення витрат енергії.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. CS Ltd. Офіційний сайт. https://csltd.com.ua/info
2. Чайковський, А. О. (2019). "Організація процесу тестування програмного забезпечення." Дисертація на здобуття наукового ступеня кандидата технічних наук. http://dspace.wunu.edu.ua/bitstream/316497/31568/1/%D0%9E%D0%9F%D0%9E%D0%91%20%D0%A7%D0%B0%D0%B9%D0%BA%D0%BE%D0%B2%D1%81%D1%8C%D0%BA%D0%B8%D0%B9.pdf
3. Мацько, І. В. (2018). "Автоматизована система тестування програмного забезпечення." Доступно на https://openarchive.nure.ua/server/api/core/bitstreams/5584e69b-4e93-4a4d-9cb7-7bf3d556625e/content
4. Шпак, І. (2017). "Впровадження Continuous Integration та Continuous Deployment." Habr. https://habr.com/ru/companies/vk/articles/347026/
5. CSS (Cascading Style Sheets). Wikipedia. https://uk.wikipedia.org/wiki/CSS
6. Java. Wikipedia. https://uk.wikipedia.org/wiki/Java
7. "Що таке CI/CD: як він працює та коли знадобиться на проекті." Highload.today. https://highload.today/uk/blogs/shho-take-ci-cd-yak-vin-pratsyuye-ta-koli-znadobitsya-na-proyekti-lajfhaki-ta-bad-practices/
8. SSH (Secure Shell). Wikipedia. https://uk.wikipedia.org/wiki/SSH
9. "Red Hat Enterprise Linux Server Self-Support (самопідтримка)." Genesis UA. https://genesisua.com/RedHat-EL-Server-Self.html
10. "Що таке SQLite?" FreeHost. https://freehost.com.ua/ukr/faq/articles/chto-takoe-sqlite/
11. Oracle Database. Wikipedia. https://uk.wikipedia.org/wiki/Oracle_Database
12. Git. Wikipedia. https://uk.wikipedia.org/wiki/Git
13. "While You Sleep: Automate Resolving Dynatrace Problem Alerts and Report Them to ServiceNow." Ansible Blog. https://www.ansible.com/blog/while-you-sleep-automate-resolving-dynatrace-problem-alerts-and-report-them-to-servicenow
14. Chef. https://www.chef.io/

15. "Security Announcements." Salt Project. https://saltproject.io/security-announcements/

16. Pupper. Infrastructure Automation https://www.puppet.com/

ДОДАТОК А

Код взаємодії користувача з системою

```
from ast import Lambda
from fileinput import close
#from os import getpriority
import sys
from urllib.error import HTTPError
from PyQt5.QtWidgets import QApplication, QMainWindow, QHeaderView
from PyQt5.QtCore import QPropertyAnimation, QEasingCurve
from PyQt5 import QtCore, QtWidgets
from PyQt5.uic import loadUi
from connection_sql import BaseInteraction
import requests
import datetime


class MyGlobalWindow (QMainWindow):
    def __init__(self):
        super(MyGlobalWindow,self).__init__()
        loadUi('sense_ifobs_tools.ui',self)

        self.base_dates = BaseInteraction()

        self.pushButtonnormal.hide()

        self.pushButton_refresh.clicked.connect(self.m_server)
        self.pushButton_add.clicked.connect(self.add_server)
        self.pushButton_find_cheng.clicked.connect(self.find_server_chenge)
        self.pushButton_chenge.clicked.connect(self.update_server)
        self.pushButton_find_del.clicked.connect(self.find_server_del)
        self.pushButton_del.clicked.connect(self.server_delete)
        self.pushButton_status.clicked.connect(self.find_server_status)
        self.pushButton_save.clicked.connect(self.add_login)

        #title botton
        self.pushButtonmin.clicked.connect(self.bt_min)
        self.pushButtonnormal.clicked.connect(self.bt_normal)
        self.pushButtonmax.clicked.connect(self.bt_max)
        self.pushButtonclos.clicked.connect(lambda: self.close())

        self.setWindowFlag(QtCore.Qt.FramelessWindowHint)
        self.setWindowOpacity(1)


        self.gripSize = 10
        self.grip = QtWidgets.QSizeGrip(self)
        self.grip.resize(self.gripSize,self.gripSize)
        #mouse
        self.frame_top.mouseMoveEvent = self.mov_form

        #Top_botton
        self.pushButton_server_top.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.page_server))
        self.pushButton_add_top.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.page_add))
        self.pushButton_change_top.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.page_chenge))
        self.pushButton_delete_top.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.page_delete))
        self.pushButton_setting_top.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.page_setting))
        self.pushButton_status_top.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.page_status))

        self.table_server.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
```

```
    self.tableWidget_2.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
    self.table_status.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)


def bt_min(self):
    self.showMinimized()

def bt_max(self):
    self.showMaximized()
    self.pushButtonmax.hide()
    self.pushButtonnormal.show()

def bt_normal(self):
    self.showNormal()
    self.pushButtonmax.show()
    self.pushButtonnormal.hide()

def resizeEvent(self,event):
    rect = self.rect()
    self.grip.move(rect.right() - self.gripSize,rect.bottom()-self.gripSize)

def mousePressEvent(self,event):
    self.click_position = event.globalPos()

def m_server(self):
    datas= self.base_dates.m_server()
    i= len(datas)
    self.table_server.setRowCount(i)
    tablerow = 0
    for row in datas:
        self.Id =row[0]
        self.table_server.setItem(tablerow,0,QtWidgets.QTableWidgetItem(row[1]))
        self.table_server.setItem(tablerow,1,QtWidgets.QTableWidgetItem(row[2]))
        self.table_server.setItem(tablerow,2,QtWidgets.QTableWidgetItem(row[3]))
        self.table_server.setItem(tablerow,3,QtWidgets.QTableWidgetItem(row[4]))
        self.table_server.setItem(tablerow,4,QtWidgets.QTableWidgetItem(row[5]))
        tablerow +=1
        self.signal_add.setText("")
        self.signal_chenge.setText("")
        self.signal_del.setText("")

def add_server(self):
    IP_SERVER = self.lineEdit_ip_add.text().upper()
    NAME_SERVER = self.lineEdit_name_add.text().upper()
    GROUP_SERVER = self.lineEdit_group_add.text().upper()
    INFO_SERVER = self.lineEdit_info_add.text().upper()
    TEST = self.lineEdit_test_add.text().upper()
    if IP_SERVER !='' and NAME_SERVER !='' and GROUP_SERVER !='' and INFO_SERVER !='' and TEST !='':
        self.base_dates.insert_server(IP_SERVER,NAME_SERVER,GROUP_SERVER,INFO_SERVER,TEST)
        self.signal_add.setText('Server insert')
        self.lineEdit_ip_add.clear()
        self.lineEdit_name_add.clear()
        self.lineEdit_group_add.clear()
        self.lineEdit_info_add.clear()
        self.lineEdit_test_add.clear()
    else:
        self.signal_add.setText('Error')

def mov_form(self,event):
    if self.isMaximized()== False:
        if event.buttons()==QtCore.Qt.LeftButton:
            self.move(self.pos()+ event.globalPos()-self.click_position)
            self.click_position = event.globalPos()
```

```
            event.accept()
        if event.globalPos().y()<=10:
            self.showMaximized()
            self.pushButtonmax.hide()
            self.pushButtonnormal.show()
        else:
            self.showNormal()
            self.pushButtonnormal.hide()
            self.pushButtonmax.show()


def find_server_chenge(self):
    id_server = self.lineEdit_find_cheng.text().upper()
    id_server = str("'" + id_server + "'")
    self.b_server = self.base_dates.find_server(id_server)
    if len(self.b_server) !=0:
        self.Id = self.b_server[0][0]
        self.lineEdit_ip_chenge.setText(self.b_server[0][1])
        self.lineEdit_name_chenge.setText(self.b_server[0][2])
        self.lineEdit_group_chenge.setText(self.b_server[0][3])
        self.lineEdit_info_chenge.setText(self.b_server[0][4])
        self.lineEdit_test_chenge.setText(self.b_server[0][5])
    else:
        self.signal_chenge.setText('Not found')

def update_server(self):
    if self.b_server !='':
        IP_SERVER = ("'"+self.lineEdit_ip_chenge.text().upper()+"'")
        NAME_SERVER = self.lineEdit_name_chenge.text().upper()
        GROUP_SERVER = self.lineEdit_group_chenge.text().upper()
        INFO_SERVER = self.lineEdit_info_chenge.text().upper()
        TEST = self.lineEdit_test_chenge.text().upper()
        upt =self.base_dates.update_server(self.Id,IP_SERVER,NAME_SERVER,GROUP_SERVER,INFO_SERVER,TEST)
        if upt ==1:
            self.signal_chenge.setText('Server update')
            self.lineEdit_ip_chenge.clear()
            self.lineEdit_name_chenge.clear()
            self.lineEdit_group_chenge.clear()
            self.lineEdit_info_chenge.clear()
            self.lineEdit_test_chenge.clear()
            self.lineEdit_find_cheng.setText('')
        elif upt ==0:
            self.signal_chenge.setText('Error')
        else:
            self.signal_chenge.setText('Incorrect data')

def find_server_del(self):
    id_server = self.lineEdit_find_del.text().upper()
    id_server = str("'" + id_server + "'")
    b_server = self.base_dates.find_server(id_server)
    self.tableWidget_2.setRowCount(len(b_server))
    if len(b_server) ==0:
        self.signal_del.setText('Not found')
    else:
        self.signal_del.setText('Server found ')
    tablerow =0
    for row in b_server:
        self.b_server_a = row[2]
        self.tableWidget_2.setItem(tablerow,0,QtWidgets.QTableWidgetItem(row[1]))
        self.tableWidget_2.setItem(tablerow,1,QtWidgets.QTableWidgetItem(row[2]))
        self.tableWidget_2.setItem(tablerow,2,QtWidgets.QTableWidgetItem(row[3]))
        self.tableWidget_2.setItem(tablerow,3,QtWidgets.QTableWidgetItem(row[4]))
        self.tableWidget_2.setItem(tablerow,4,QtWidgets.QTableWidgetItem(row[5]))
```

```
        tablerow +=1


def server_delete(self):
    self.row_flag =self.tableWidget_2.currentRow()
    if self.row_flag==0:
        self.tableWidget_2.removeRow(0)
        self.base_dates.del_server('''' + self.b_server_a + '''')
        self.signal_del.setText('Server removed')
        self.lineEdit_find_del.setText('')


def add_login(self):
    user_login = self.lineEdit_login.text()
    user_pass = self.lineEdit_pas.text()
    if len(user_login)==0:
        return
    elif len(user_pass)==0:
        return
    if self.base_dates.find_login_pass('''+user_login+''') is None:
        self.base_dates.insert_login_pass(user_login,user_pass)
        self.signal_setting.setText('OK')
    else:
        self.signal_setting.setText('Error')


def find_server_status(self):
    datas= self.base_dates.m_server()
    #print(datas)
    self.table_status.setRowCount(len(datas))
    tablerow_b =0
    tablerow_c =0
    tablerow_f =0
    tablerow_t =0
    tablerow_w =0
    for row in datas:
        self.b_server_a = (self.base_dates.find_server_status('''+row[5]+''',row[0]))
        print('self.b_server_a:',self.b_server_a)
        url_create = (''HTTPS://'' + row[1] )
        print('url_create: ',url_create)
        url_check = requests.get(url_create)
        print('url_check',url_check )
        try:
            url_check = requests.get(url_create,timeout=1)
            print('url_check:',url_check)
            url_check.raise_for_status()
            url_check.status_code()
            print('url_check.raise_for_status():',url_check.raise_for_status())
            print('url_check.status_code:',url_check.status_code())
        except HTTPError as http_err:
            if row[5]=='TEST_C':
                self.table_status.setItem(tablerow_c,1,QtWidgets.QTableWidgetItem(row[3]+" '''Error'))
                tablerow_c +=1
            elif row[5]=='TEST_T':
                self.table_status.setItem(tablerow_t,3,QtWidgets.QTableWidgetItem(row[3]+" '''Error'))
                tablerow_t +=1
            elif row[5]=='TEST_B':
                self.table_status.setItem(tablerow_b,0,QtWidgets.QTableWidgetItem(row[3]+" '''Error'))
                tablerow_b +=1
            elif row[5]=='TEST_F':
                self.table_status.setItem(tablerow_f,2,QtWidgets.QTableWidgetItem(row[3]+" '''Error'))
                tablerow_f +=1
            elif row[5]=='TEST_W':
```

```
                self.table_status.setItem(tablerow_w,4,QtWidgets.QTableWidgetItem(row[3]+" '''Error'))
                tablerow_w+=1
            else:
                print('error')
        except Exception  as err:
            if row[5]=='TEST_C':
                self.table_status.setItem(tablerow_c,1,QtWidgets.QTableWidgetItem(row[3]+" '''Error'))
                tablerow_c +=1
            elif row[5]=='TEST_T':
                self.table_status.setItem(tablerow_t,3,QtWidgets.QTableWidgetItem(row[3]+" '''Error'))
                tablerow_t +=1
            elif row[5]=='TEST_B':
                self.table_status.setItem(tablerow_b,0,QtWidgets.QTableWidgetItem(row[3]+" '''Error'))
                tablerow_b +=1
            elif row[5]=='TEST_F':
                self.table_status.setItem(tablerow_f,2,QtWidgets.QTableWidgetItem(row[3]+" '''Error'))
                tablerow_f +=1
            elif row[5]=='TEST_W':
                self.table_status.setItem(tablerow_w,4,QtWidgets.QTableWidgetItem(row[3]+" '''Error'))
                tablerow_w+=1
            else:
                print('error')
        else:
            if url_check.status_code ==200:
                if row[5]=='TEST_C':
                    self.table_status.setItem(tablerow_c,1,QtWidgets.QTableWidgetItem(row[3]+" '''OK'))
                    tablerow_c +=1
                elif row[5]=='TEST_T':
                    self.table_status.setItem(tablerow_t,3,QtWidgets.QTableWidgetItem(row[3]+" '''OK'))
                    tablerow_t +=1
                elif row[5]=='TEST_B':
                    self.table_status.setItem(tablerow_b,0,QtWidgets.QTableWidgetItem(row[3]+" '''OK'))
                    tablerow_b +=1
                elif row[5]=='TEST_F':
                    self.table_status.setItem(tablerow_f,2,QtWidgets.QTableWidgetItem(row[3]+" '''OK'))
                    tablerow_f +=1
                elif row[5]=='TEST_W':
                    self.table_status.setItem(tablerow_w,4,QtWidgets.QTableWidgetItem(row[3]+" '''OK'))
                    tablerow_w+=1
                else:
                    print('error')

    now=datetime.datetime.now()
    self.label_status.setText('Status updated...'+str(now))



if __name__=="__main__":
    app=QApplication(sys.argv)
    my_app= MyGlobalWindow()
    my_app.show()
    sys.exit(app.exec_())
```

ДОДАТОК Б

Код для підключення та керування даними в БД

```
import sqlite3


class BaseInteraction ():
    def __init__(self):
        self.connection = sqlite3.connect('sebse_ifobs_tools.db')



    def insert_server(self,IP_SERVER,NAME_SERVER,GROUP_SERVER,INFO_SERVER,TEST):
        cursor =self.connection.cursor()
        bd= "'INSERT INTO setting_server (IP_SERVER,NAME_SERVER,GROUP_SERVER,INFO_SERVER,TEST)
        VALUES('{}','{}','{}','{}','{}')'".format(IP_SERVER,NAME_SERVER,GROUP_SERVER,INFO_SERVER,TEST)
        cursor.execute(bd)
        self.connection.commit()
        cursor.close()

    def m_server(self):
        cursor =self.connection.cursor()
        bd= "SELECT * FROM setting_server"
        cursor.execute(bd)
        registro=cursor.fetchall()
        cursor.close()
        return registro

    def find_server(self,ip_server):
        cursor =self.connection.cursor()
        bd= "'SELECT * FROM setting_server WHERE IP_SERVER = {}'".format(ip_server)
        cursor.execute(bd)
        ipserver=cursor.fetchall()
        cursor.close()
        return ipserver

    def del_server(self,ip_server):
        cursor =self.connection.cursor()
        bd="'DELETE FROM setting_server WHERE IP_SERVER = {}'".format(ip_server)
        cursor.execute(bd)
        self.connection.commit()
        cursor.close()
```

```
    def update_server(self,Id,IP_SERVER,NAME_SERVER,GROUP_SERVER,INFO_SERVER,TEST):
        cursor =self.connection.cursor()
        bd="'UPDATE setting_server SET IP_SERVER
=={},NAME_SERVER='{}',GROUP_SERVER='{}',INFO_SERVER='{}',TEST='{}'
        WHERE ID ='{}' '".format(IP_SERVER,NAME_SERVER,GROUP_SERVER,INFO_SERVER,TEST,Id)
        cursor.execute(bd)
        a= cursor.rowcount
        self.connection.commit()
        cursor.close()
        return a

    def find_server_status(self,test_name,sId):
        cursor =self.connection.cursor()
        bd= "'SELECT * FROM setting_server WHERE TEST = {} and ID = {}'".format(test_name,sId)
        cursor.execute(bd)
        ipserver2=cursor.fetchone()
        cursor.close()
        return ipserver2

    def find_test_name2(self,test_name):
        cursor =self.connection.cursor()
        bd= "'SELECT TEST FROM setting_server WHERE IP_SERVER = {}'".format(test_name)
        cursor.execute(bd)
        test_name=cursor.fetchall()
        cursor.close()
        return test_name

    def find_test_name(self,test_name):
        cursor =self.connection.cursor()
        bd= "SELECT TEST FROM setting_server "
        cursor.execute(bd)
        test_name=cursor.fetchall()
        cursor.close()
        return test_name

    def find_login_pass(self,login):
        cursor = self.connection.cursor()
        bd= "'SELECT * FROM users WHERE USER_LOGIN = {}'".format(login)
        cursor.execute(bd)
        ipserver2=cursor.fetchone()
        cursor.close()
```

```
    return ipserver2

def insert_login_pass(self,u_login,u_pass):
    cursor =self.connection.cursor()
    bd= '''INSERT INTO users (USER_LOGIN,USER_PASSWORD)
    VALUES('{}','{}')'''.format(u_login,u_pass)
    cursor.execute(bd)
    self.connection.commit()
    cursor.close()
```

ДОДАТОК В

Код дизайну та розмітки системи

```
<?xml version="1.0" encoding="UTF-8"?>
<<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>840</width>
    <height>461</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <property name="font">
    <font>
     <pointsize>13</pointsize>
    </font>
   </property>
   <property name="styleSheet">
    <string notr="true">QFrame {
background-color:rgb(113, 170, 170);
}

QPushButton{
background-color:#71aaaa;



}

QPushButton:hover{
background-color:rgb(84, 126, 126);
border-radius : 20xp;
}</string>
   </property>
   <layout class="QHBoxLayout" name="horizontalLayout_14" stretch="0">
```

```
<property name="spacing">
 <number>0</number>
</property>
<property name="leftMargin">
 <number>0</number>
</property>
<property name="topMargin">
 <number>0</number>
</property>
<property name="rightMargin">
 <number>0</number>
</property>
<property name="bottomMargin">
 <number>0</number>
</property>
<item>
 <widget class="QFrame" name="frame">
  <property name="styleSheet">
   <string notr="true"/>
  </property>
  <property name="frameShape">
   <enum>QFrame::StyledPanel</enum>
  </property>
  <property name="frameShadow">
   <enum>QFrame::Raised</enum>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout_2" stretch="1,10">
   <property name="spacing">
    <number>0</number>
   </property>
   <property name="leftMargin">
    <number>0</number>
   </property>
   <property name="topMargin">
    <number>0</number>
   </property>
   <property name="rightMargin">
    <number>0</number>
   </property>
   <property name="bottomMargin">
    <number>0</number>
   </property>
```

```
      <item>
       <widget class="QFrame" name="frame_top">
        <property name="enabled">
         <bool>true</bool>
        </property>
        <property name="maximumSize">
         <size>
          <width>16777215</width>
          <height>40</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true"/>
        </property>
        <property name="frameShape">
         <enum>QFrame::StyledPanel</enum>
        </property>
        <property name="frameShadow">
         <enum>QFrame::Raised</enum>
        </property>
        <layout class="QHBoxLayout" name="horizontalLayout" stretch="1,15,4">
         <property name="spacing">
          <number>0</number>
         </property>
         <property name="leftMargin">
          <number>0</number>
         </property>
         <property name="topMargin">
          <number>0</number>
         </property>
         <property name="rightMargin">
          <number>0</number>
         </property>
         <property name="bottomMargin">
          <number>0</number>
         </property>
         <item>
          <widget class="QFrame" name="frame_logo">
           <property name="styleSheet">
            <string notr="true">QFrame {
background-color:rgb(113, 170, 170)
}
```

```
QPushButton{
background-color:#000000ff;
border-radus:20px;
}

QPushButton:hover{
background-color: rgb(61,61,61);
border-radus:20px;
}
</string>
            </property>
            <property name="frameShape">
             <enum>QFrame::StyledPanel</enum>
            </property>
            <property name="frameShadow">
             <enum>QFrame::Raised</enum>
            </property>
            <layout class="QHBoxLayout" name="horizontalLayout_13">
             <property name="spacing">
              <number>0</number>
             </property>
             <property name="leftMargin">
              <number>0</number>
             </property>
             <property name="topMargin">
              <number>0</number>
             </property>
             <property name="rightMargin">
              <number>0</number>
             </property>
             <property name="bottomMargin">
              <number>0</number>
             </property>
             <item>
              <widget class="QPushButton" name="pushButton_top_ico">
               <property name="text">
                <string/>
               </property>
               <property name="icon">
                <iconset>
```

<normaloff>../../Alpha_ifobs_tools/Alpha_ifobs_tools/logo_varik_new.ico</normaloff>../../Alpha_ifobs_tools/Alpha_ifobs_tools/logo_varik_new.ico</iconset>
</property>
<property name="iconSize">
<size>
<width>32</width>
<height>32</height>
</size>
</property>
</widget>
</item>
</layout>
</widget>
</item>
<item>
<widget class="QFrame" name="frame_title">
<property name="styleSheet">
<string notr="true">QFrame {
background-color:rgb(113, 170, 170)
}

QPushButton{
background-color:#000000ff;
border-radus:20px;
}

QPushButton:hover{
background-color: rgb(61,61,61);
border-radus:20px;
}</string>
</property>
<property name="frameShape">
<enum>QFrame::StyledPanel</enum>
</property>
<property name="frameShadow">
<enum>QFrame::Raised</enum>
</property>
<layout class="QVBoxLayout" name="verticalLayout_7">
<item>
<widget class="QLabel" name="label_top_title">
<property name="text">

```
      <string>Sense_ifobs_tools</string>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
</item>
<item>
 <widget class="QFrame" name="frame_bt">
  <property name="styleSheet">
   <string notr="true"/>
  </property>
  <property name="frameShape">
   <enum>QFrame::StyledPanel</enum>
  </property>
  <property name="frameShadow">
   <enum>QFrame::Raised</enum>
  </property>
  <layout class="QHBoxLayout" name="horizontalLayout_3">
   <property name="spacing">
    <number>0</number>
   </property>
   <property name="leftMargin">
    <number>0</number>
   </property>
   <property name="topMargin">
    <number>0</number>
   </property>
   <property name="rightMargin">
    <number>0</number>
   </property>
   <property name="bottomMargin">
    <number>0</number>
   </property>
   <item>
    <widget class="QPushButton" name="pushButtonmin">
     <property name="sizePolicy">
      <sizepolicy hsizetype="Minimum" vsizetype="Minimum">
       <horstretch>0</horstretch>
       <verstretch>0</verstretch>
      </sizepolicy>
     </property>
```

```
            <property name="font">
             <font>
              <stylestrategy>PreferDefault</stylestrategy>
             </font>
            </property>
            <property name="accessibleDescription">
             <string/>
            </property>
            <property name="styleSheet">
             <string notr="true">QFrame {
background-color:rgb(113, 170, 170);
}

QPushButton{
background-color:#71aaaa;
border-radius: 20xp;

}

QPushButton:hover{
background-color:rgb(84, 126, 126);
border-radius : 20xp;
}</string>
            </property>
            <property name="text">
             <string/>
            </property>
            <property name="icon">
             <iconset>

<normaloff>C:/Users/Windows/Downloads/minBt.png</normaloff>C:/Users/Windows/Downloads/minBt.png</iconset>
            </property>
            <property name="iconSize">
             <size>
              <width>20</width>
              <height>20</height>
             </size>
            </property>
           </widget>
          </item>
          <item>
           <widget class="QPushButton" name="pushButtonnormal">
```

```
            <property name="sizePolicy">
             <sizepolicy hsizetype="Minimum" vsizetype="Minimum">
              <horstretch>0</horstretch>
              <verstretch>0</verstretch>
             </sizepolicy>
            </property>
            <property name="styleSheet">
             <string notr="true">QFrame {
background-color:rgb(113, 170, 170);
}

QPushButton{
background-color:#71aaaa;
border-radius: 20xp;

}

QPushButton:hover{
background-color:rgb(84, 126, 126);
border-radius : 20xp;
}</string>
            </property>
            <property name="text">
             <string/>
            </property>
            <property name="icon">
             <iconset>

<normaloff>C:/Users/Windows/Downloads/resize_minimize_maximize_expand_screen_icon_190602.png</normaloff>C:/Users/Windows/Downloads/resize_minimize_maximize_expand_screen_icon_190602.png</iconset>
            </property>
            <property name="iconSize">
             <size>
              <width>20</width>
              <height>20</height>
             </size>
            </property>
           </widget>
          </item>
          <item>
           <widget class="QPushButton" name="pushButtonmax">
            <property name="sizePolicy">
```

```
<sizepolicy hsizetype="Minimum" vsizetype="Minimum">
<horstretch>0</horstretch>
<verstretch>0</verstretch>
</sizepolicy>
</property>
<property name="styleSheet">
<string notr="true">QFrame {
background-color:rgb(113, 170, 170);
}

QPushButton{
background-color:#71aaaa;
border-radius: 20xp;
}

QPushButton:hover{
background-color:#547e7e;
border-radius: 20xp;
}</string>
</property>
<property name="text">
<string/>
</property>
<property name="icon">
<iconset>

<normaloff>C:/Users/Windows/Downloads/resize_minimize_maximize_expand_screen_icon_190600.png</normaloff>C:/Users/Windows/Downloads/resize_minimize_maximize_expand_screen_icon_190600.png</iconset>
</property>
<property name="iconSize">
<size>
<width>20</width>
<height>20</height>
</size>
</property>
</widget>
</item>
<item>
<widget class="QPushButton" name="pushButtonclos">
<property name="sizePolicy">
<sizepolicy hsizetype="Minimum" vsizetype="Minimum">
<horstretch>0</horstretch>
```

```
            <verstretch>0</verstretch>
           </sizepolicy>
          </property>
          <property name="maximumSize">
           <size>
            <width>16777215</width>
            <height>16777215</height>
           </size>
          </property>
          <property name="styleSheet">
           <string notr="true">QFrame {
background-color:rgb(113, 170, 170);
}

QPushButton{
background-color:#71aaaa;
border-radius: 20xp;

}

QPushButton:hover{
background-color:#547e7e;
border-radius: 20xp;
}</string>
          </property>
          <property name="text">
           <string/>
          </property>
          <property name="icon">
           <iconset>

<normaloff>C:/Users/Windows/Downloads/closeBt2.png</normaloff>C:/Users/Windows/Downloads/closeBt2.png</iconset>
          </property>
          <property name="iconSize">
           <size>
            <width>20</width>
            <height>20</height>
           </size>
          </property>
         </widget>
        </item>
       </layout>
```

```
          </widget>
         </item>
        </layout>
       </widget>
      </item>
      <item>
       <widget class="QFrame" name="frame_work">
        <property name="styleSheet">
         <string notr="true">QFrame {
background-color:rgb(222, 255, 222);
}

QPushButton {
            box-shadow:inset 0px 0px 14px -3px #f2fadc;
            background:linear-gradient(to bottom, #dbe6c4 5%, #9ba892 100%);
            background-color:#dbe6c4;
            border-radius:6px;
            border:1px solid #b2b8ad;
            display:inline-block;
            cursor:pointer;
            color:#757d6f;
            font-family:Arial;
            font-size:12px;
            font-weight:bold;
            padding:5px 5px;
            text-decoration:none;
            text-shadow:0px 1px 0px #ced9bf;
}
QPushButton:hover {
            background:linear-gradient(to bottom, #9ba892 5%, #dbe6c4 100%);
            background-color:#9ba892;
}
QPushButton:active {
            position:relative;
            top:1px;
}</string>
        </property>
        <property name="frameShape">
         <enum>QFrame::StyledPanel</enum>
        </property>
        <property name="frameShadow">
         <enum>QFrame::Raised</enum>
```

```
</property>
<layout class="QVBoxLayout" name="verticalLayout" stretch="1,9">
 <property name="spacing">
  <number>0</number>
 </property>
 <property name="leftMargin">
  <number>0</number>
 </property>
 <property name="topMargin">
  <number>0</number>
 </property>
 <property name="rightMargin">
  <number>0</number>
 </property>
 <property name="bottomMargin">
  <number>0</number>
 </property>
 <item>
  <widget class="QFrame" name="frame_topbar">
   <property name="styleSheet">
    <string notr="true"/>
   </property>
   <property name="frameShape">
    <enum>QFrame::StyledPanel</enum>
   </property>
   <property name="frameShadow">
    <enum>QFrame::Raised</enum>
   </property>
   <layout class="QHBoxLayout" name="horizontalLayout_4">
    <item>
     <widget class="QPushButton" name="pushButton_server_top">
      <property name="styleSheet">
       <string notr="true"/>
      </property>
      <property name="text">
       <string>Server</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QPushButton" name="pushButton_add_top">
      <property name="styleSheet">
```

```
  <string notr="true"/>
 </property>
 <property name="text">
  <string>Add server</string>
 </property>
</widget>
</item>
<item>
<widget class="QPushButton" name="pushButton_change_top">
 <property name="styleSheet">
  <string notr="true"/>
 </property>
 <property name="text">
  <string>Change</string>
 </property>
</widget>
</item>
<item>
<widget class="QPushButton" name="pushButton_delete_top">
 <property name="styleSheet">
  <string notr="true"/>
 </property>
 <property name="text">
  <string>Delete</string>
 </property>
</widget>
</item>
<item>
<widget class="QPushButton" name="pushButton_setting_top">
 <property name="styleSheet">
  <string notr="true"/>
 </property>
 <property name="text">
  <string>Setting</string>
 </property>
</widget>
</item>
<item>
<widget class="QPushButton" name="pushButton_status_top">
 <property name="styleSheet">
  <string notr="true"/>
 </property>
```

```
            <property name="text">
             <string>Status server</string>
            </property>
           </widget>
          </item>
         </layout>
        </widget>
       </item>
       <item>
        <widget class="QFrame" name="frame_control">
         <property name="styleSheet">
          <string notr="true">QFrame {
           background-color:#71aaaa;
}

QPushButton {
            box-shadow:inset 0px 0px 14px -3px #f2fadc;
            background:linear-gradient(to bottom, #dbe6c4 5%, #9ba892 100%);
            background-color:#dbe6c4;
            border-radius:6px;
            border:1px solid #b2b8ad;
            display:inline-block;
            cursor:pointer;
            color:#757d6f;
            font-family:Arial;
            font-size:15px;
            font-weight:bold;
            padding:6px 24px;
            text-decoration:none;
            text-shadow:0px 1px 0px #ced9bf;
}
QPushButton:hover {
            background:linear-gradient(to bottom, #9ba892 5%, #dbe6c4 100%);
            background-color:#9ba892;
}
QPushButton:active {
            position:relative;
            top:1px;
}
QLabel{
            background-color:#71aaaa;
            border:0px solid #b2b8ad;
```

```
            color:#353535;
            font:87 15pt &amp;quot;Arial&amp;quot;;
}

QTableWidget{
            background-color: rgb(222, 255, 222);
            color:rgb(84, 126, 126);
            gridline-color: rgb(0, 0, 0);
            font-size:12pt;
            color:#000000;
}

QHeaderView::section{
            background-color: #dbe6c4;
            border:1px solid #b2b8ad;
            font-size:12pt;
}

QTableWidget QTableCornerButton::section{
            background-color: #dbe6c4;
            border:1px solid #b2b8ad;
}
</string>
          </property>
          <property name="frameShape">
           <enum>QFrame::StyledPanel</enum>
          </property>
          <property name="frameShadow">
           <enum>QFrame::Raised</enum>
          </property>
          <layout class="QVBoxLayout" name="verticalLayout_3">
           <property name="spacing">
            <number>0</number>
           </property>
           <property name="leftMargin">
            <number>1</number>
           </property>
           <property name="topMargin">
            <number>1</number>
           </property>
           <property name="rightMargin">
            <number>1</number>
```

```
</property>
<property name="bottomMargin">
 <number>1</number>
</property>
<item>
 <widget class="QStackedWidget" name="stackedWidget">
  <property name="currentIndex">
   <number>5</number>
  </property>
  <widget class="QWidget" name="page_server">
   <layout class="QVBoxLayout" name="verticalLayout_4">
    <item>
     <widget class="QLabel" name="label_server">
      <property name="font">
       <font>
        <family>Arial</family>
        <pointsize>15</pointsize>
        <weight>10</weight>
        <italic>false</italic>
        <bold>false</bold>
       </font>
      </property>
      <property name="styleSheet">
       <string notr="true"/>
      </property>
      <property name="text">
       <string>Test server</string>
      </property>
      <property name="alignment">
       <set>Qt::AlignCenter</set>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QTableWidget" name="table_server">
      <property name="font">
       <font>
        <pointsize>12</pointsize>
       </font>
      </property>
      <property name="layoutDirection">
       <enum>Qt::LeftToRight</enum>
```

```
</property>
<property name="autoFillBackground">
 <bool>false</bool>
</property>
<property name="inputMethodHints">
 <set>Qt::ImhNone</set>
</property>
<property name="lineWidth">
 <number>1</number>
</property>
<property name="midLineWidth">
 <number>0</number>
</property>
<property name="verticalScrollBarPolicy">
 <enum>Qt::ScrollBarAsNeeded</enum>
</property>
<property name="horizontalScrollBarPolicy">
 <enum>Qt::ScrollBarAsNeeded</enum>
</property>
<property name="sizeAdjustPolicy">
 <enum>QAbstractScrollArea::AdjustIgnored</enum>
</property>
<attribute name="horizontalHeaderMinimumSectionSize">
 <number>150</number>
</attribute>
<attribute name="horizontalHeaderShowSortIndicator" stdset="0">
 <bool>false</bool>
</attribute>
<attribute name="horizontalHeaderStretchLastSection">
 <bool>true</bool>
</attribute>
<column>
 <property name="text">
  <string>IP</string>
 </property>
</column>
<column>
 <property name="text">
  <string>Server Name</string>
 </property>
</column>
<column>
```

```
     <property name="text">
      <string>Server Group</string>
     </property>
    </column>
    <column>
     <property name="text">
      <string>Server Info</string>
     </property>
    </column>
    <column>
     <property name="text">
      <string>Test</string>
     </property>
    </column>
   </widget>
  </item>
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_5">
    <item>
     <spacer name="horizontalSpacer">
      <property name="orientation">
       <enum>Qt::Horizontal</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>40</width>
        <height>20</height>
       </size>
      </property>
     </spacer>
    </item>
    <item>
     <widget class="QPushButton" name="pushButton_refresh">
      <property name="text">
       <string>Refresh</string>
      </property>
     </widget>
    </item>
   </layout>
  </item>
 </layout>
</widget>
```

```
<widget class="QWidget" name="page_status">
 <layout class="QVBoxLayout" name="verticalLayout_9">
  <item>
   <widget class="QLabel" name="label_status_title">
    <property name="text">
     <string>Status server </string>
    </property>
    <property name="alignment">
     <set>Qt::AlignCenter</set>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QTableWidget" name="table_status">
    <property name="font">
     <font>
      <pointsize>12</pointsize>
     </font>
    </property>
    <attribute name="horizontalHeaderMinimumSectionSize">
     <number>150</number>
    </attribute>
    <attribute name="horizontalHeaderDefaultSectionSize">
     <number>150</number>
    </attribute>
    <attribute name="horizontalHeaderShowSortIndicator" stdset="0">
     <bool>false</bool>
    </attribute>
    <attribute name="horizontalHeaderStretchLastSection">
     <bool>true</bool>
    </attribute>
    <attribute name="verticalHeaderMinimumSectionSize">
     <number>33</number>
    </attribute>
    <column>
     <property name="text">
      <string>TEST_B</string>
     </property>
    </column>
    <column>
     <property name="text">
      <string>TEST_C</string>
```

```
  </property>
 </column>
 <column>
  <property name="text">
   <string>TEST_F</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>TEST_T</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>TEST_W</string>
  </property>
 </column>
</widget>
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_2">
  <item>
   <widget class="QLabel" name="label_status">
    <property name="minimumSize">
     <size>
      <width>120</width>
      <height>0</height>
     </size>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
  </item>
  <item>
   <spacer name="horizontalSpacer_status">
    <property name="orientation">
     <enum>Qt::Horizontal</enum>
    </property>
    <property name="sizeHint" stdset="0">
     <size>
      <width>40</width>
```

```
      <height>20</height>
     </size>
    </property>
   </spacer>
  </item>
  <item>
   <widget class="QPushButton" name="pushButton_status">
    <property name="text">
     <string>Refresh</string>
    </property>
   </widget>
  </item>
 </layout>
</item>
</layout>
</widget>
<widget class="QWidget" name="page_add">
 <layout class="QVBoxLayout" name="verticalLayout_8" stretch="1,20,1">
  <property name="spacing">
   <number>9</number>
  </property>
  <property name="leftMargin">
   <number>9</number>
  </property>
  <property name="topMargin">
   <number>9</number>
  </property>
  <property name="rightMargin">
   <number>9</number>
  </property>
  <property name="bottomMargin">
   <number>9</number>
  </property>
  <item>
   <widget class="QLabel" name="label">
    <property name="text">
     <string>Add server</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignCenter</set>
    </property>
   </widget>
```

```
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_6">
  <item>
   <layout class="QVBoxLayout" name="verticalLayout_6">
    <property name="spacing">
     <number>30</number>
    </property>
    <item>
     <widget class="QLabel" name="label_ip_add">
      <property name="text">
       <string>IP</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QLabel" name="label_name_add">
      <property name="text">
       <string>SERVER NAME</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QLabel" name="label_group_add">
      <property name="text">
       <string>SERVER GROUP</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QLabel" name="label_2">
      <property name="text">
       <string>SERVER INFO</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QLabel" name="label_test_add">
      <property name="text">
       <string>TEST</string>
      </property>
     </widget>
```

```
   </item>
  </layout>
 </item>
 <item>
  <layout class="QVBoxLayout" name="verticalLayout_5">
   <property name="spacing">
    <number>30</number>
   </property>
   <item>
    <widget class="QLineEdit" name="lineEdit_ip_add"/>
   </item>
   <item>
    <widget class="QLineEdit" name="lineEdit_name_add"/>
   </item>
   <item>
    <widget class="QLineEdit" name="lineEdit_group_add"/>
   </item>
   <item>
    <widget class="QLineEdit" name="lineEdit_info_add"/>
   </item>
   <item>
    <widget class="QLineEdit" name="lineEdit_test_add"/>
   </item>
  </layout>
 </item>
</layout>
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_7">
  <property name="spacing">
   <number>3</number>
  </property>
  <item>
   <widget class="QLabel" name="signal_add">
    <property name="minimumSize">
     <size>
      <width>120</width>
      <height>0</height>
     </size>
    </property>
    <property name="text">
     <string/>
```

```
      </property>
     </widget>
    </item>
    <item>
     <spacer name="horizontalSpacer_add">
      <property name="orientation">
       <enum>Qt::Horizontal</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>40</width>
        <height>20</height>
       </size>
      </property>
     </spacer>
    </item>
    <item>
     <widget class="QPushButton" name="pushButton_add">
      <property name="text">
       <string>Add</string>
      </property>
     </widget>
    </item>
   </layout>
  </item>
 </layout>
</widget>
<widget class="QWidget" name="page_chenge">
 <layout class="QVBoxLayout" name="verticalLayout_12" stretch="1,2,10,1">
  <item>
   <widget class="QLabel" name="label_title_chenge">
    <property name="text">
     <string>Chenge</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignCenter</set>
    </property>
   </widget>
  </item>
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_10">
    <item>
```

```
  <widget class="QLabel" name="label_find_chenge">
   <property name="text">
    <string>IP server</string>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QLineEdit" name="lineEdit_find_cheng"/>
 </item>
 <item>
  <widget class="QPushButton" name="pushButton_find_cheng">
   <property name="text">
    <string>Find</string>
   </property>
  </widget>
 </item>
</layout>
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_9">
  <property name="spacing">
   <number>3</number>
  </property>
  <item>
   <layout class="QVBoxLayout" name="verticalLayout_11">
    <property name="spacing">
     <number>30</number>
    </property>
    <item>
     <widget class="QLabel" name="label_ip_chenge">
      <property name="text">
       <string>IP</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QLabel" name="label_name_chenge">
      <property name="text">
       <string>SERVER NAME</string>
      </property>
     </widget>
    </item>
```

```
  <item>
   <widget class="QLabel" name="label_group_chenge">
    <property name="text">
     <string>SERVER GROUP</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLabel" name="label_info_change">
    <property name="text">
     <string>SERVER INFO</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLabel" name="label_test_chenge">
    <property name="text">
     <string>TEST</string>
    </property>
   </widget>
  </item>
 </layout>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_10">
  <property name="spacing">
   <number>30</number>
  </property>
  <item>
   <widget class="QLineEdit" name="lineEdit_ip_chenge"/>
  </item>
  <item>
   <widget class="QLineEdit" name="lineEdit_name_chenge"/>
  </item>
  <item>
   <widget class="QLineEdit" name="lineEdit_group_chenge"/>
  </item>
  <item>
   <widget class="QLineEdit" name="lineEdit_info_chenge"/>
  </item>
  <item>
   <widget class="QLineEdit" name="lineEdit_test_chenge"/>
```

```
   </item>
  </layout>
 </item>
</layout>
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_8">
  <item>
   <widget class="QLabel" name="signal_chenge">
    <property name="minimumSize">
     <size>
      <width>120</width>
      <height>0</height>
     </size>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
  </item>
  <item>
   <spacer name="horizontalSpacer_chenge">
    <property name="orientation">
     <enum>Qt::Horizontal</enum>
    </property>
    <property name="sizeHint" stdset="0">
     <size>
      <width>40</width>
      <height>20</height>
     </size>
    </property>
   </spacer>
  </item>
  <item>
   <widget class="QPushButton" name="pushButton_chenge">
    <property name="text">
     <string>Chenge</string>
    </property>
   </widget>
  </item>
 </layout>
</item>
```

```
 </layout>
</widget>
<widget class="QWidget" name="page_delete">
 <layout class="QVBoxLayout" name="verticalLayout_13">
  <item>
   <widget class="QLabel" name="label_delete">
    <property name="text">
     <string>Delete server</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignCenter</set>
    </property>
   </widget>
  </item>
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_11">
    <item>
     <widget class="QLabel" name="label_find_del">
      <property name="text">
       <string>IP Server</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QLineEdit" name="lineEdit_find_del"/>
    </item>
    <item>
     <widget class="QPushButton" name="pushButton_find_del">
      <property name="text">
       <string>Find</string>
      </property>
     </widget>
    </item>
   </layout>
  </item>
  <item>
   <widget class="QTableWidget" name="tableWidget_2">
    <property name="sizePolicy">
     <sizepolicy hsizetype="Expanding" vsizetype="Expanding">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
```

```
</property>
<property name="minimumSize">
 <size>
  <width>0</width>
  <height>0</height>
 </size>
</property>
<property name="layoutDirection">
 <enum>Qt::LeftToRight</enum>
</property>
<property name="autoFillBackground">
 <bool>false</bool>
</property>
<property name="sizeAdjustPolicy">
 <enum>QAbstractScrollArea::AdjustIgnored</enum>
</property>
<property name="autoScroll">
 <bool>true</bool>
</property>
<property name="alternatingRowColors">
 <bool>false</bool>
</property>
<property name="selectionMode">
 <enum>QAbstractItemView::ExtendedSelection</enum>
</property>
<property name="selectionBehavior">
 <enum>QAbstractItemView::SelectItems</enum>
</property>
<property name="textElideMode">
 <enum>Qt::ElideRight</enum>
</property>
<property name="verticalScrollMode">
 <enum>QAbstractItemView::ScrollPerItem</enum>
</property>
<property name="horizontalScrollMode">
 <enum>QAbstractItemView::ScrollPerItem</enum>
</property>
<property name="showGrid">
 <bool>true</bool>
</property>
<property name="gridStyle">
 <enum>Qt::SolidLine</enum>
```

```
</property>
<property name="sortingEnabled">
 <bool>false</bool>
</property>
<property name="wordWrap">
 <bool>true</bool>
</property>
<property name="cornerButtonEnabled">
 <bool>true</bool>
</property>
<property name="rowCount">
 <number>0</number>
</property>
<property name="columnCount">
 <number>5</number>
</property>
<attribute name="horizontalHeaderVisible">
 <bool>false</bool>
</attribute>
<attribute name="horizontalHeaderCascadingSectionResizes">
 <bool>false</bool>
</attribute>
<attribute name="horizontalHeaderMinimumSectionSize">
 <number>150</number>
</attribute>
<attribute name="horizontalHeaderDefaultSectionSize">
 <number>150</number>
</attribute>
<attribute name="horizontalHeaderHighlightSections">
 <bool>true</bool>
</attribute>
<attribute name="horizontalHeaderShowSortIndicator" stdset="0">
 <bool>false</bool>
</attribute>
<attribute name="horizontalHeaderStretchLastSection">
 <bool>true</bool>
</attribute>
<attribute name="verticalHeaderVisible">
 <bool>false</bool>
</attribute>
<attribute name="verticalHeaderCascadingSectionResizes">
 <bool>false</bool>
```

```
</attribute>
<attribute name="verticalHeaderMinimumSectionSize">
 <number>33</number>
</attribute>
<attribute name="verticalHeaderDefaultSectionSize">
 <number>33</number>
</attribute>
<attribute name="verticalHeaderHighlightSections">
 <bool>true</bool>
</attribute>
<attribute name="verticalHeaderShowSortIndicator" stdset="0">
 <bool>false</bool>
</attribute>
<attribute name="verticalHeaderStretchLastSection">
 <bool>false</bool>
</attribute>
<column>
 <property name="text">
  <string>IP</string>
 </property>
 <property name="font">
  <font>
   <pointsize>8</pointsize>
  </font>
 </property>
 <property name="textAlignment">
  <set>AlignLeading|AlignVCenter</set>
 </property>
</column>
<column>
 <property name="text">
  <string>SERVER NAME</string>
 </property>
 <property name="textAlignment">
  <set>AlignLeading|AlignVCenter</set>
 </property>
</column>
<column>
 <property name="text">
  <string>SERVER GROUP</string>
 </property>
 <property name="textAlignment">
```

```
   <set>AlignLeading|AlignVCenter</set>
  </property>
 </column>
 <column>
  <property name="text">
   <string>SERVER INFO</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>TEST</string>
  </property>
  <property name="textAlignment">
   <set>AlignLeading|AlignVCenter</set>
  </property>
 </column>
</widget>
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_12">
  <item>
   <widget class="QLabel" name="signal_del">
    <property name="minimumSize">
     <size>
      <width>120</width>
      <height>0</height>
     </size>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
  </item>
  <item>
   <spacer name="horizontalSpacer_del">
    <property name="orientation">
     <enum>Qt::Horizontal</enum>
    </property>
    <property name="sizeHint" stdset="0">
     <size>
      <width>40</width>
      <height>20</height>
```

```
      </size>
     </property>
    </spacer>
   </item>
   <item>
    <widget class="QPushButton" name="pushButton_del">
     <property name="text">
      <string>Delete</string>
     </property>
    </widget>
   </item>
  </layout>
 </item>
</layout>
</widget>
<widget class="QWidget" name="page_setting">
 <layout class="QVBoxLayout" name="verticalLayout_20">
  <item>
   <widget class="QLabel" name="label_setting">
    <property name="text">
     <string>Setting</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignCenter</set>
    </property>
   </widget>
  </item>
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_18">
    <item>
     <layout class="QVBoxLayout" name="verticalLayout_19">
      <property name="leftMargin">
       <number>1</number>
      </property>
      <property name="topMargin">
       <number>1</number>
      </property>
      <property name="rightMargin">
       <number>1</number>
      </property>
      <property name="bottomMargin">
       <number>1</number>
```

```
</property>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_16">
  <item>
   <layout class="QVBoxLayout" name="verticalLayout_15">
    <item>
     <widget class="QLabel" name="label_login">
      <property name="text">
       <string>Login</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QLabel" name="label_pas">
      <property name="text">
       <string>Password</string>
      </property>
     </widget>
    </item>
   </layout>
  </item>
  <item>
   <layout class="QVBoxLayout" name="verticalLayout_16">
    <item>
     <widget class="QLineEdit" name="lineEdit_login"/>
    </item>
    <item>
     <widget class="QLineEdit" name="lineEdit_pas"/>
    </item>
   </layout>
  </item>
 </layout>
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_15">
  <item>
   <widget class="QLabel" name="signal_setting">
    <property name="minimumSize">
     <size>
      <width>150</width>
      <height>0</height>
     </size>
```

```
      </property>
      <property name="text">
       <string/>
      </property>
     </widget>
    </item>
    <item>
     <spacer name="horizontalSpacer_2">
      <property name="orientation">
       <enum>Qt::Horizontal</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>40</width>
        <height>20</height>
       </size>
      </property>
     </spacer>
    </item>
    <item>
     <widget class="QPushButton" name="pushButton_save">
      <property name="text">
       <string>Save</string>
      </property>
     </widget>
    </item>
   </layout>
  </item>
 </layout>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_14">
  <property name="leftMargin">
   <number>1</number>
  </property>
  <property name="topMargin">
   <number>1</number>
  </property>
  <property name="rightMargin">
   <number>1</number>
  </property>
  <property name="bottomMargin">
```

```
    <number>1</number>
   </property>
   <item>
    <widget class="QLabel" name="label_3">
     <property name="text">
      <string>Create update package</string>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QLineEdit" name="lineEdit"/>
   </item>
   <item>
    <widget class="QPushButton" name="pushButton">
     <property name="text">
      <string>Path to the archive (original)</string>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QPushButton" name="pushButton_2">
     <property name="text">
      <string>Create</string>
     </property>
    </widget>
   </item>
  </layout>
 </item>
</layout>
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_17">
  <item>
   <layout class="QVBoxLayout" name="verticalLayout_17">
    <item>
     <widget class="QPushButton" name="pushButton_stop_b">
      <property name="text">
       <string>Stop TEST_B</string>
      </property>
     </widget>
    </item>
    <item>
```

```
  <widget class="QPushButton" name="pushButton_stop_c">
   <property name="text">
    <string>Stop TEST_C</string>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QPushButton" name="pushButton_stop_f">
   <property name="text">
    <string>Stop TEST_F</string>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QPushButton" name="pushButton_stop_t">
   <property name="text">
    <string>Stop TEST_T</string>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QPushButton" name="pushButton_stop_w">
   <property name="text">
    <string>Stop TEST_W</string>
   </property>
  </widget>
 </item>
</layout>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_18">
  <item>
   <widget class="QPushButton" name="pushButton_start_b">
    <property name="text">
     <string>Start TEST_B</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QPushButton" name="pushButton_start_c">
    <property name="text">
     <string>Start TEST_C</string>
```

```
                </property>
               </widget>
              </item>
              <item>
               <widget class="QPushButton" name="pushButton_start_f">
                <property name="text">
                 <string>Start TEST_F</string>
                </property>
               </widget>
              </item>
              <item>
               <widget class="QPushButton" name="pushButton_start_t">
                <property name="text">
                 <string>Start TEST_T</string>
                </property>
               </widget>
              </item>
              <item>
               <widget class="QPushButton" name="pushButton_start_w">
                <property name="text">
                 <string>Start TEST_W</string>
                </property>
               </widget>
              </item>
             </layout>
            </item>
           </layout>
          </item>
         </layout>
        </widget>
       </widget>
      </item>
     </layout>
    </widget>
   </item>
  </layout>
 </widget>
</item>
</layout>
```

```
  </widget>
 </widget>
 <resources>
  <include location="../logo.qrc"/>
 </resources>
 <connections/>
</ui>
```