МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Т.в.о. завідувача кафедри
_____ Сафонова С.О.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Методи автоматизованого тестування web-сервисів

Освітній рівень"Магістр"
Спеціальність 123 "Комп'ютерна інженерія"

Науковий керівник роботи:	_____ (підпис)	В.М.Барбарук (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я.О.Критська (ініціали, прізвище)
Студент:	_____ (підпис)	П.Ю.Кубрак (ініціали, прізвище)
Група:		КІ-18зм

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітній рівень магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 123 "Комп'ютерна інженерія"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Т.в.о. завідувача кафедри _____
_____ С.О. Сафонова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Кубраку Павлу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи автоматизованого тестування web-сервисів

керівник проекту (роботи) Барбарук Віктор Миколайович, к.т.н., доц.
(прізвище, м.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «11» 10 2019 р. № 136/15.15
2. Строк подання студентом роботи 10.01.2020
3. Вихідні дані до роботи Матеріали науково-дослідної практики, вибір веб-сервісу для тестування, теоретичні відомості про веб-сервіси, теоретичні відомості про види тестування веб-сервісів, середовище розробки автоматизованих тестів SoapUI

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Огляд стану предметної області, автоматизоване тестування веб-сервісів, аналіз результатів автоматизованого тестування веб-сервісів, охорона праці та безпека в надзвичайних ситуаціях, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	Критська Я.О. ст. викл. кафедри КНІ		

7. Дата видачі завдання 14.10.2019

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання до магістерської роботи	02.09.2019-15.09.2019	
2	Аналіз завдання, робота з літературою	16.09.2019-22.09.2019	
3	Написання тестових сценаріїв	23.09.2019-25.09.2019	
4	Тестування веб-сервісу	26.09.2019-06.10.2019	
5	Розробка частини проекту "Охорона праці та безпеки в надзвичайних ситуаціях"	07.10.2019-25.11.2019	
6	Оформлення пояснювальної записки, автореферату та презентації	26.11.2019-9.01.2020	
7			

Студент _____ П.Ю.Кубрак
(підпис) (прізвище та ініціали)

Науковий керівник _____ В.М.Барбарук
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кубрак П.Ю. Методи автоматизованого тестування web-сервисів.

Дана робота присвячена дослідженню та аналізу автоматизації тестування веб-сервісу. Для написання авто-тестів необхідно створити тестовий набір,який включає в себе низку тестових сценаріїв з тестовими кроками. Тестовий набір – це сукупність тестових сценаріїв, які об'єднані для перевірки одного модулю чи функціоналу з однією метою. Автоматизовані тести були написані за допомогою платформи SoapUI.

Проведено дослідження результатів автоматизації за допомогою інструментів середовища SoapUI. Результати досліджень дозволяють зробити висновок про доцільність застосування розроблених автоматизованих тестів для покращення якості у ході тестування Web-сервісів, а також для визначення рівню безпеки обміну даними. Зроблений висновок на основі аналізу результатів тестування.

Ключові слова: SOAPUI, тестування, веб-сервіси, види тестування, тестовий набір, тестові кроки, api, wsdl, xml, тестові сценарії, протоколи, автоматизоване тестування.

ABSTRACT

Kubrak P.Y. Methods of automated test web-services.

This paper deals with the research and web-services automation analysis. To write auto-tests is required to create a test suite that includes a set of test cases with test steps. A test suite includes a set of test cases that are combined to test one module or functionality with one purpose. Automated tests were written by means of the SoapUI tool.

Automation testing research results with SoapUI tools has been conducted. Research results allow concluding that it is advisable to use the developed automated tests to improve the quality during the web-services testing, as well as to determine the security level during data exchange. The conclusion is based on the test results analysis.

Keywords: SOAPUI, testing, web-services, types of testing, test suite, test steps, api, wsdl, xml, test cases, protocols, automation testing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення

АТ– автоматизоване тестування

API – Application Programming Interface

WSDL – Web Services Description Language

TCP – Transmission Control Protocol

HTTP – HyperText Transfer Protocol

FTP – File Transfer Protocol

OIM – Oracle Identity Manager

ВСТУП

Пік інтересу до тестування програмного забезпечення відбувся у дев'яностих роках у США. Швидкий розвиток систем автоматизованої розробки програмного забезпечення та мережевих технологій призвів до збільшення виробництва на ринку програмного забезпечення. Посилення конкуренції між виробниками програмного забезпечення вимагало підвищеної уваги до якості продукції. Оскільки асортимент продукції розширився, а ціни стали доступнішими, споживачі почали звертати більшу увагу на якість програмного забезпечення. В даний час практично всі сфери життя схильні до комп'ютеризації. Комп'ютери використовуються в повсякденному житті для звичайних цілей, вони необхідні, коли мова йде про такі значущі сфери, як медицина, транспорт, будівництво, безпека. Таким чином, питання про якість програмного забезпечення є особливо важливим, оскільки це не тільки питання комфорту, а й безпеки [4].

Різні додатки на різних вузлах мережі функціонують на різних апаратно-програмних платформах, і використовують різні технології і мови. Щоб зв'язати все це і надати можливість одним додаткам обмінюватися даними з іншими, були придумані веб-сервіси [3]. Веб-сервіси - це реалізація абсолютно чітких інтерфейсів обміну даними між різними веб-системами, які написані не тільки на різних мовах, але і розподілені на різних вузлах мережі.

Функціональність нового класу веб-систем дуже різноманітна. Розглянемо практичне застосування веб-сервісів: припустимо, авіакомпанія надає веб-сервіс, що дозволяє додаткам отримувати список рейсів між двома містами для заданої дати. У цьому випадку більше не потрібно звертатися до веб-вузла авіакомпанії і вказувати різні критерії пошуку - вся необхідна інформація доступна у вигляді єдиного XML-документа [4]. Тепер припустимо, що авіакомпанія, готель і агентство з прокату автомобілів надають веб-сервіси, що дозволяють програмно купувати квитки, бронювати номери і орендувати автомобілі. В цьому випадку можна об'єднати виклики всіх цих сервісів в єдину програму, яка зможе виконати всю рутинну роботу без участі користувача. Веб-система може, наприклад, періодично звертатися до веб-сервісу авіакомпанії для визначення статусу рейсу і в разі його затримки сповіщати сервіси готелю, служби прокату для продовження бронювання. Наприклад, якщо агентство прокату автомобілів знає, що ваш рейс затримується, воно може більш гнучко розпорядитися своїми автомобілями. У мірі зростання числа веб-сервісів можна побачити більш комплексні приклади їх використання. Однак впровадження концепції веб-сервісів вимагає не тільки перегляду багатьох бізнес-правил і схем взаємодії між галузями і секторами тій чи іншій галузі, а й

підвищення безпеки обміну даними [3].

Щоб визначити рівень безпеки обміну даними необхідно забезпечити якість веб-сервісів за допомогою їх тестування. Щоб значно заощадити час на тестування, необхідно виконати автоматизоване тестування.

Атестаційна робота присвячена розробці авто-тестів для дослідження та аналізу автоматизації тестування веб-сервісів.

1 ОГЛЯД СТАНУ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття «API»

API (Application Programming Interface) – це набір процедур, протоколів та інструментів для створення програмних веб-систем. API визначає, як повинні взаємодіяти програмні компоненти. API складається з двох пов'язаних елементів [8]. Перший елемент – це специфікація, яка описує, як виконується обмін інформацією між програмами та має вигляд запиту на обробку та повернення необхідних даних. Другий елемент – це програмний інтерфейс, написаний до специфікації, який має певний вигляд для використання [9].

Програмне забезпечення, яке було розроблено для конкретної мети, тепер часто має посилання на API, які надають широко корисні функції, зменшують час розробки та вартість,а також зм'якшують ризик помилок. Протягом останнього десятиліття API постійно покращували якість ПЗ. Хороший дизайн API є надзвичайно важливим для його успішного використання, а архітектори програмного забезпечення витрачають значний час, переглядаючи всі можливі програми API та найбільш логічний спосіб його використання. Структури даних та значення параметрів мають особливе значення, оскільки вони повинні співпадати між тим, хто викликає API та його видавцем [8 – 9].

Існують наступні види API:

а) локальні – пропонують послуги операційних систем або проміжних програм для прикладних програм;

б) веб-інтерфейси – інтерфейси, які призначені для широко використовуваних ресурсів, таких як HTML-сторінки, де доступ здійснюється за допомогою простого HTTP-протоколу;

в) програмні – базуються на технології виклику віддаленої процедури, яка робить компоненти віддаленої програми локальними для решти ПЗ.

API має функцію, яка називається «формат відповіді». Вона дозволяє вибирати вид даних, який має вигляд json або xml.

Щоб краще пояснити, як працює API, розглянемо два приклади:

1) Коли ви використовуєте програму на своєму мобільному телефоні, програма підключається до Інтернету та надсилає дані на сервер. Сервер потім витягує ці дані, інтерпретує, виконує необхідні дії та відправляє його назад на ваш телефон. Програма потім інтерпретує ці дані та подає вам потрібну інформацію у зручний для читання спосіб [8 – 9];

2) Наприклад, ви сидите за столом у ресторані з меню вибору на замовлення. Кухня

є частиною системи, яка готує ваше замовлення. Недостатнім компонентом цієї системи є модуль для передачі замовлення на кухню та доставки їжі назад до вашого столу. Для цього потрібен офіціант або API. Офіціант - це API, який приймає ваше замовлення, і каже, що кухні потрібно робити. Потім офіціант передає відповідь вам – їжу [3].

Дані вашого телефону ніколи не піддаються дії сервера та навпаки. Замість цього кожен з них спілкується з невеликими пакетами даних, обмінюючись лише тим, що є необхідним. Ви повідомляєте ресторану, що ви хотіли б поїсти, а вони кажуть вам, що вони потребують гроші. Тобто, це означає що API надає високий рівень безпеки, тому вони стали настільки цінними, що складають значну частину доходу багатьох компаній. Основні компанії, такі як Google, eBay, Salesforce.com, Amazon та Expedia - це лише деякі з компаній, які заробляють гроші за допомогою API [8 – 9].

Зараз дуже широко використовується такий тип API як веб API. Цей тип застосовується у веб-розробці та має вигляд набору HTTP-запитів, а також відповідні структури HTTP-відповідей. Веб API визначає, як програмні компоненти повинні взаємодіяти один з одним за допомогою протоколу Інтернету (HTTP) як проміжний інтерфейс. Клієнту не потрібно знати, яку процедуру викликати на сервер. Натомість він використовує набір команд, які вбудовані в HTTP, і коли команда надходить на інший кінець, система приймаючої системи знає, що з ним робити. Перевагою веб API є гнучкість. Клієнтська система та обслуговуюча система настільки незалежні один від одного, що кожен може використовувати різні мови: Java, Python, Ruby та ін.

Веб API практично є синонімом веб-сервісів. При їх порівнянні значення полягає в обсязі роботи, яку повинні зробити клієнтська система та обслуговуюча система. Пакетування та розпакування даних у сценарії веб API зазвичай вимагає набагато меншої роботи, що, в свою чергу, прирівнює до кращої продуктивності та меншого обсягу обчислювальних циклів. Це одна з причин того, чому веб-інтерфейси відмінно підходять для передачі інформації на мобільних пристроях і планшетах, на відміну від настільних комп'ютерів та сервісів, де вони мають обмежене середовище обробки [8 – 9]. І навпаки, веб-сервіси полегшують взаємодію між двома системами і майже завжди залежать від XML-подібного інтерфейсу, щоб спілкуватися один з одним.

Подібно до всього ПЗ, API має бути протестовані. Мета тестування – перевірка опублікованих API за специфікаціями, які користувачі цих API будуть використовувати при форматуванні своїх запитів. Це тестування, як правило, виконується як частина керування життєвим циклом додатків як для ПЗ, яке публікує API, так і для всього ПЗ, яке їх використовує. API також повинні бути перевірені в опублікованій формі, щоб забезпечити належний доступ до них [9].

1.2 Поняття «веб-сервіс»

Веб-сервіси – це системи обміну інформацією на базі XML, які використовують Інтернет для прямої взаємодії між програмами [1]. Ці системи можуть включати програми, об'єкти, повідомлення або документи. Веб-сервіси не прив'язані до жодної операційної системи або мови програмування. Вони містять кінцеві точки – це URL-адреса, за якою клієнти певної служби можуть отримати доступ до неї. Посилаючись на цю URL-адресу, клієнти отримують доступ до операцій, що надаються цією службою [1].

Під комунікацією з веб-сервісом мається на увазі розбір XML- повідомлення при його отриманні, виконання необхідних операцій, упаковку результатів знову у XML-повідомлення і його відправлення [4]. Комунікація з веб-сервісами виконується за допомогою транспортних протоколів,таких як HTTP, HTTPS, FTP, SMTP, BEEP, при цьому технології веб-сервісів можна поділити на чотири види:

а) SOAP веб-сервіси – це веб-сервіси, взаємодія з якими виконується за допомогою XML- повідомлень по SOAP протоколу та які мають інтерфейси, описані у форматі WSDL. Ці веб-сервіси орієнтовані на виклик видалених процедур [4];

б) XML веб-сервіси – це веб-сервіси, орієнтовані на повідомлення. Вони забезпечують низькорівневу обробку XML- повідомлень, при цьому веб-сервіс обробляє отримані XML- повідомлення цілком та повністю формує XML- повідомлення у відповідь [3];

в) REST веб-сервіси – це веб-сервіси, які представляють віддалені ресурси, доступні за допомогою HTTP-запитів. Вони забезпечують взаємодію з віддаленими ресурсами, передаючи клієнту їх уявлення. REST веб-сервіси ідентифікуються URL-адресою та обробляють HTTP- методи GET, PUT, POST і DELETE у відповідь на запит клієнта [3 – 4];

г) UDDI веб-сервіси – це веб-сервіси, які представляють собою всесвітній реєстр веб-сервісів для реклами, пошуку та зберігання. UDDI надає структуру для подання ділових відносин, веб-сервісів, технічних метаданих і точок доступу до веб-сервісів [4].

Веб-сервіси мають клієнтів двох типів:

– перший тип клієнта – це клієнтська програма, де взаємодію з веб-сервісом ініціює реальний користувач. Це можуть бути додатки типу веб-браузер чи десктопні додатки з графічним інтерфейсом користувача;

– другий тип клієнта – це програмний компонент, який автоматично здійснює запит веб-сервісу без участі людини.

Взаємодія клієнта з веб-сервісом може бути синхронною чи асинхронною. У

випадку синхронної взаємодії, після відправки клієнтом запиту веб-сервісу, усі його дії блокуються до тих пір, поки не буде отримана відповідь. У випадку асинхронної взаємодії, дії клієнта не блокуються, а обробка відповіді здійснюється після її отримання від веб-сервісу [1]. Таким чином, існують наступні сценарії взаємодії з веб-сервісом [1]:

1) Односпрямований запит, де клієнт відправляє повідомлення веб-сервісу, який його обробляє, але не генерують відповідь;

2) Синхронний «запит-відповідь», де здійснюється синхронний обмін повідомленнями між клієнтом і веб-сервісом;

3) Асинхронний «запит-відповідь», де здійснюється асинхронний обмін повідомленнями між клієнтом і веб-сервісом;

4) Помилки, де виконання процедури інтерфейсу веб-сервісу завершується помилкою, і клієнт отримує повідомлення про помилку;

5) Передача через посередників, де клієнтське повідомлення передається кінцевому веб-сервісу через проміжні веб-сервіси;

6) Кешування, де у випадку схожого клієнтського запиту відповідь йому надходить з кешу;

7) Діалог, де повідомлення, які беруть участь у обміні з веб-сервісами, групуються у логічні набори зі збереженням стану;

8) Додаткова функціональність, де обмін повідомленнями виконується с додатковими вимогами, такими як кодування, аутентифікація, цілісність, транзакції [3 – 4].

При створенні веб-сервісу з метою надання послуг для широкого кола споживачів необхідно, щоб потенційні користувачі веб-сервісу змогли його знайти. Для звичайних користувачів інтернету веб-сервіси становляться доступними за допомогою елементів керування, розміщених на сторінках сайтів. Для розробників веб-сервіс необхідно зареєструвати в якому-небудь закритому реєстрі, який дозволяє здійснювати як статичний, так і динамічний пошук необхідного веб-сервісу [1].

1.3 Протоколи веб-сервісів

Системою класифікації мережевих протоколів є модель OSI. Відповідно до неї протоколи діляться на сім рівнів за своїм призначенням [7]:

Прикладний рівень – це верхній (7-й) рівень моделі, який забезпечує взаємодію мережі й користувача. Рівень дозволяє додаткам користувача доступ до мережевих служб,

таким як обробник запитів до баз даних, доступ до файлів, пересилання електронних повідомлень. Також відповідає за передачу службової інформації, надає додаткам інформацію про помилки і формує запити до рівня уявлення [7];

Рівень уявлення – це 6-й рівень,який відповідає за перетворення протоколів і кодування , декодування даних. Запити програм, отримані з рівня додатків, цей рівень перетворює в формат для передачі по мережі, а отримані з мережі дані перетворює у формат, зрозумілий додаткам. На рівні уявлення може здійснюватися стиснення, розпакування або кодування, декодування даних, а також перенаправлення запитів іншому мережному ресурсу, якщо вони не можуть бути оброблені локально [7].

Сеансовий рівень – це 5-й рівень моделі, який відповідає за підтримання сеансу зв'язку, що дозволяє додаткам взаємодіяти між собою тривалий час. Сеансовий рівень управляє створенням, завершенням сеансу, обміном інформацією, синхронізацією завдань, визначенням права на передачу даних і підтримкою сеансу в періоди неактивності додатків. Синхронізація передачі забезпечується поміщенням в потік даних контрольних точок, починаючи з яких поновлюється процес при порушенні взаємодії [3].

Транспортний рівень – це 4-й рівень моделі, призначений для доставки даних без помилок, втрат і дублювання в тій послідовності, як вони були передані. При цьому неважливо, які дані передаються, звідки і куди, тобто він надає сам механізм передачі. Блоки даних він розділяє на фрагменти, розмір яких залежить від протоколу, короткі об'єднує в один, а довгі розбиває.

Мережевий рівень – це 3-й рівень, призначений для визначення шляху передачі даних. Відповідає за трансляцію логічних адрес і імен у фізичні, визначення найкоротших маршрутів, комутацію і маршрутизацію, відстеження неполадок і заторів в мережі. На цьому рівні працює такий мережний пристрій, як маршрутизатор [7].

Канальний рівень – це 2-й рівень, призначений для забезпечення взаємодії мереж на фізичному рівні і контролю за помилками, які можуть виникнути. Дані, отримані з фізичного рівня, він упаковує у фрейми, перевіряє на цілісність, якщо потрібно виправляє помилки і відправляє на мережевий рівень. Канальний рівень може взаємодіяти з одним або декількома фізичними рівнями, контролюючи і керуючи цим взаємодією [7].

Фізичний рівень – це найнижчий рівень моделі, призначений безпосередньо для передачі потоку даних. Здійснює інтерфейс між мережним носієм і мережним пристроєм. Функції фізичного рівня реалізуються на всіх пристроях, підключених до мережі. З боку комп'ютера функції фізичного рівня виконуються мережевим адаптером або послідовним портом.

Протокол – це набір угод інтерфейсу логічного рівня, які визначають обмін даними між різними програмами [3]. Веб-сервіси можуть взаємодіяти один з одним і зі сторонніми

додатками за допомогою повідомлень, заснованих на певних протоколах, таких як:

– TCP / IP – сімейство протоколів, призначених для взаємодії між електронними пристроями. Визначає, як вони повинні бути пов'язані через інтернет і як потрібно передавати дані між ними;

– TCP – відповідає за розбиття даних на невеликі пакети перед тим, як пересилати їх по мережі, і за збірку цих пакетів в початковий стан після отримання [7];

– IP – відповідає за обмін даними між пристроями, тобто за адресацію, відправлення та одержання пакетів даних через інтернет;

– DNS (Domain Name Server) – ієрархічна система імен, побудована на розподілених базах даних. Коли користувач відвідує сайт, то ім'я сайту перетворюється в числове уявлення за допомогою DNS. А коли реєструється новий домен разом з TCP / IP адресою, DNS по всьому світу фіксують цю інформацію [7];

– HTTP (HyperText Transfer Protocol) – протокол рівня додатка, який використовується в основному в WWW. HTTP використовує клієнт-серверну модель, де браузер є клієнтом і спілкується з веб-сервером;

– HTTPS (HyperText Transfer Protocol Secure) – різновид HTTP протоколу, який додає даним, які передаються рівень безпеки через SSL протокол або TLS протокол;

– FTP – протокол, який використовується для передачі або обміну файлами між комп'ютерами. FTP часто використовується для завантаження веб-сторінок та інших документів з приватного пристрою розробки на відкриті сервера хостингу [7];

– SSL (Secure Socket Layer) – стандартний протокол, який використовується для захищеної передачі документів через мережу;

– SNMP (Simple Network Management Protocol) – стандартний інтернет-протокол для управління пристроями в IP-мережах на основі архітектур UDP / TCP;

– LDAP (Lightweight Directory Access Protocol) – використовується для збору інформації про користувачів і електронні поштові адреси з інтернету [7];

– POP (Post Office Protocol) – використовується програмами з поштовою функціональністю для відшукання потрібної пошти з поштового сервера;

– IMAP (Internet Message Access Protocol) – діє так само, як і POP, тільки не завантажує відразу все запитувані пошти, а дає можливість подивитися на повідомлення, які видає поштовий сервер або видалити їх з бази [7];

– SMTP (Simple Mail Transfer Protocol) – піклується про відправлення повідомлень в пошту. Зазвичай повідомлення надсилаються на поштовий сервер, а потім в інші сервери і тільки потім в пункт призначення. SMTP може передавати тільки текстові дані;

– MIME (Multi-purpose Internet Mail Extensions) – виконує такі ж функції, що і SMTP, тільки може ще передавати в повідомленнях двійкові дані, тобто аудіо, відео,

зображення [3, 7];

– NFS (Network File System) – протокол мережевого доступу до файлових систем, який дозволяє підключати вилучені файлові системи через мережу [7].

1.4 WSDL – мова опису веб-сервісів

WSDL – це XML-документ, що описує веб-сервіс [4]. Кожен документ має свою структуру (рис. 1.1-1.2), що складається з таких частин:

1) <wsdl:types> – визначення виду відправлених і отриманих сервісом XML повідомлень;

2) <wsdl:message> – повідомлення, що використовуються веб-сервісом;

3) <wsdl:portType> – список операцій, які можуть бути виконані з повідомленнями [4];

4) <wsdl:binding> – спосіб, яким повідомлення буде доставлено;

5) <wsdl:import> – посилається на окремий документ WSDL 1.1 з описами, що підлягають включенню в цей документ;

6) <wsdl:service> – визначає сервіс в цілому, як правило, включає один або кілька елементів <wsdl:port> з інформацією доступу для елементів <wsdl:binding> [4].

```
<wsdl:definitions ... xmlns:tns="http://sosnoski.com/ws/library/Bool
    targetNamespace="http://sosnoski.com/ws/library/BookServerInter
  <wsdl:document>Book service interface definition.</wsdl:document>
  <wsdl:types>
    <xs:schema ...
        targetNamespace="http://sosnoski.com/ws/library/BookServerI
      <xs:import namespace="http://sosnoski.com/ws/library/types"
          schemaLocation="book-types.xsd"/>
      ...
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="getBookMessage">
    <wsdl:part name="part" element="tns:getBook"/>
  </wsdl:message>
  <wsdl:message name="getBookResponseMessage">
    <wsdl:part name="part" element="tns:getBookResponse"/>
  </wsdl:message>
  ...
  <wsdl:message name="addBookMessage">
    <wsdl:part name="part" element="tns:addBook"/>
  </wsdl:message>
  <wsdl:message name="addBookResponseMessage">
    <wsdl:part name="part" element="tns:addBookResponse"/>
  </wsdl:message>
  <wsdl:message name="addDuplicateFault">
    <wsdl:part name="fault" element="tns:addDuplicate"/>
  </wsdl:message>
  <wsdl:portType name="BookServerPortType">
    <wsdl:documentation>
      Book service implementation. This creates an initial library
      class is loaded, then supports method calls to access the lib
      (including adding new books).
    </wsdl:documentation>
    <wsdl:operation name="getBook">
      <wsdl:documentation>
        Get the book with a particular ISBN.
      </wsdl:documentation>
      <wsdl:input message="tns:getBookMessage"/>
      <wsdl:output message="tns:getBookResponseMessage"/>
    </wsdl:operation>
    ...
    <wsdl:operation name="addBook">
      <wsdl:documentation>Add a new book.</wsdl:documentation>
      <wsdl:input message="tns:addBookMessage"/>
      <wsdl:output message="tns:addBookResponseMessage"/>
      <wsdl:fault message="tns:addDuplicateFault" name="addDuplicat
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

Рисунок 1.1 – Приклад WSDL документу

```
<wsdl:definitions ... xmlns:ins="http://sosnoski.com/ws/library/Bool
    xmlns:tns="http://sosnoski.com/ws/library/BookServer"
    targetNamespace="http://sosnoski.com/ws/library/BookServer">
  <wsdl:document>
    Definition of actual book service implementation.
  </wsdl:document>
  <wsdl:import namespace="http://sosnoski.com/ws/library/BookServer
      location="BookServerInterface.wsdl"/>
  <wsdl:binding name="BookServerBinding" type="ins:BookServerPortTy
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" s
    <wsdl:operation name="getBook">
      <soap:operation soapAction="urn:getBook"/>
      <wsdl:input>
        <soap:body/>
      </wsdl:input>
      <wsdl:output>
        <soap:body/>
      </wsdl:output>
    </wsdl:operation>
    ...
    <wsdl:operation name="addBook">
      <soap:operation soapAction="urn:addBook"/>
      <wsdl:input>
        <soap:body/>
      </wsdl:input>
      <wsdl:output>
        <soap:body/>
      </wsdl:output>
      <wsdl:fault name="addDuplicateFault">
        <soap:fault name="addDuplicateFault"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="BookServer">
    <wsdl:port name="BookServerPort" binding="tns:BookServerBinding
      <soap:address location="http://localhost:8080/cxf/BookServer",
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Рисунок 1.2 – Приклад WSDL документу

Секція <wsdl:types> включає визначення XML у вигляді елементів <s: schema>. Компонентам документа WSDL присвоюються окремі імена з використанням атрибута name. При використанні в кореневому елементі документа <wsdl:definitions> атрибута targetNamespace, імена цих компонентів визначено в цьому просторі імен. Це означає, що при визначенні імені досить привласнити просту, або «локальну», частину імені, але посилання на цей компонент повинні уточнювати ім'я за допомогою префікса простору імен або за допомогою простору імен за замовчуванням [4].

Елементи <wsdl:message> є описами XML-даних, переданих між клієнтом і постачальником послуг, які розташовані в ядрі описів сервісів WSDL. Кожен елемент <wsdl:message> містить, як правило, один дочірній елемент <wsdl:part>, для якого потрібно власний унікальний в межах <wsdl: message> атрибут name і один з атрибутів element або type, який посилається на визначення схеми XML-даних [1].

Елементи <wsdl:portType> визначають абстрактний інтерфейс сервісу в частині повідомлень, переданих сервісу та прийнятих від нього, і містять будь-яку кількість дочірніх елементів <wsdl:operation>. Дочірній елемент <wsdl:operation> має власний атрибут name і включає один або кілька дочірніх елементів з описом повідомлення. Залежно від способу використання можуть бути включені такі дочірні елементи, як: <wsdl:input>, де містяться вхідні дані, що відправляються клієнтом постачальника послуг; <wsdl:output>, де повертаються клієнтові постачальником послуг дані; <wsdl:fault>, де повертаються клієнтові постачальником послуг дані при виникненні помилки [1, 4].

Дочірні елементи <wsdl:binding> містять інформацію про спосіб реалізації. Елементи з простору імен WSDL відповідають елементам <wsdl: portType> і повинні використовувати те ж значення name, а не посилання з уточненням простору імен. Та ж зв'язок по імені відноситься і до дочірнім елементам <wsdl:input>, <wsdl:output>, <wsdl: fault> елементів <wsdl: operation> [4]. Незважаючи на повторне використання одних і тих самих імен елементів, зміст цих елементів істотно відрізняється для дочірніх елементів <wsdl:binding> щодо елементів <wsdl:portType>.

Елемент <wsdl:service> складається з групи елементів <wsdl:port>. Кожен елемент <wsdl:port> пов'язує адрес доступу з <wsdl:binding>. Адреса доступу визначається атрибутом <soap:address> [4]. Зв’язок між компонентами WSDL представлений на рисунку 1.3.

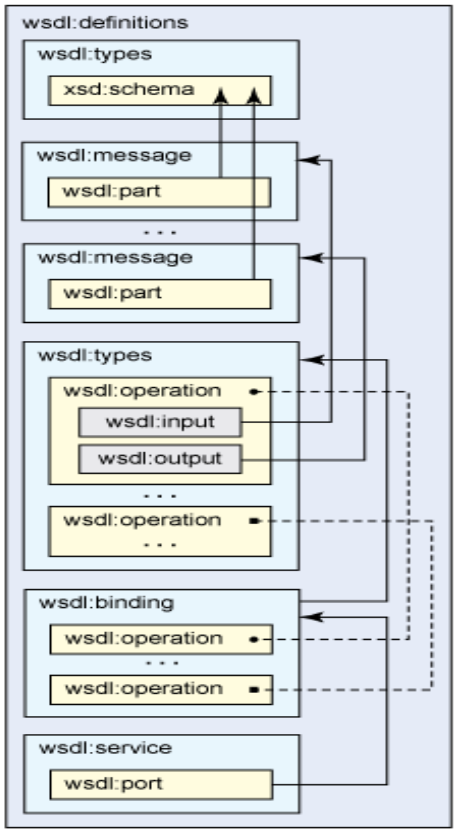


Рисунок 1.3 – Зв’язок між компонентами WSDL

1.5 Елементи тестування веб-сервісів

Кожен XML-документ містить запити, які відправляються за URL- адресою та відповіді на ці запити. Для того, щоб перевірити правильність запитів і відповідей необхідно тестувати такі елементи: дані, типи даних, їх порядок та повноту. Також

необхідно перевіряти клієнтську частину, яку використовує цей веб-сервіс, http-статуси, авторизацію, тайм-аут відповіді, виконувати навантажувальне тестування та тестування безпеки [1].

Розглянемо приклад тестування веб-сервісу, у якому підраховується сума чисел (рис. 1.4).

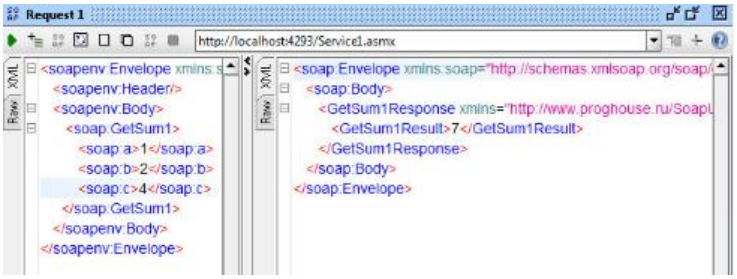


Рисунок 1.4 – Тестування даних

Для того, щоб перевірити дані, необхідно заповнити усі обов’язкові поля у запиті. Оскільки даний крок – це позитивний тест, у відповіді підраховується сума [1]. Далі необхідно зробити перевірку типів даних, для цього дані в одному обов’язковому полі повинні містити не числове значення (рис. 1.5). Даний крок – це негативний тест, у результаті запиту буде помилка. Порядок елементів у запиті не обов’язковий, тобто не має значення у якій послідовності будуть розташовані елементи, у відповіді ми отримаємо суму даних чисел (рис. 1.6). Щоб зробити перевірку повноти запиту, необхідно одне обов’язкове поле залишити порожнім. Даний крок – це також негативний тест, тому у результаті запиту буде помилка (рис.1.7).

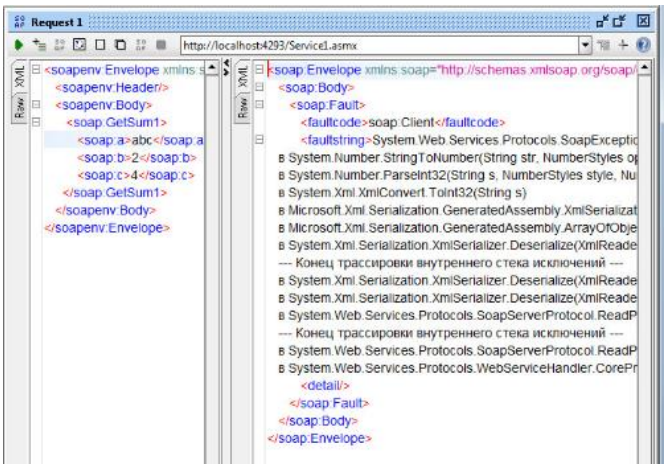


Рисунок 1.5 – Тестування типів даних

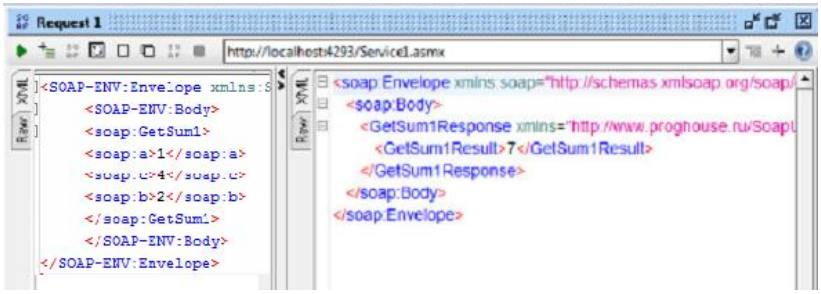


Рисунок 1.6 – Тестування порядку даних у запиті

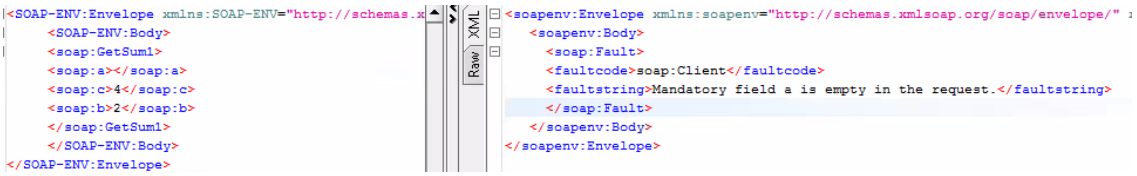


Рисунок 1.7 – Тестування повноти даних

1.6 Постановка задачі дослідження

Метою даної атестаційної роботи є дослідження результатів якості веб-сервісів за допомогою автоматизованого тестування. Для цього необхідно виконати наступні завдання:

– дослідити програмний продукт;

– дослідити методи автоматизованого тестування веб-сервісів;

– розробити набір тест-кейсів;

– розробити і програмно реалізувати тест-кейси за допомогою функцій автоматизованого тестування у SoapUI;

– провести дослідження та аналіз розроблених автоматизованих тестів.

Для тестування програмного продукту була обрана система Oracle Identity Manager (OIM).

2 АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ ВЕБ-СЕРВІСІВ

2.1 Тестові стратегії веб-сервісів

Як і в веб-системах, для веб-сервісів потрібні різні види тестових стратегій. Існують такі підходи до тестування [6]:

1) Модульне тестування;
2) Базове тестування;
3) Тестування SOA;
4) Тестування сумісності;
5) Навантажувальне тестування.

Веб-сервіси подібні до будь-яких інших програм, тому модульне тестування є обов’язковим. Тестові кроки модульного тестування повинні бути написані перед розробкою програми. Коли програма розроблена, тестові кроки виконуються для перевірки коду, а саме окремих невеликих частин програми, які можна досліджувати ізольовано від інших подібних частин [6].

У базовому тестуванні основною метою є перевірка доступності веб-сервісу. Основна увага на цьому етапі повинна полягати у виконанні наступних процедур:

– необхідно отримати wsdl файл і перевірити, чи він добре сформований та відповідає усім вимогам з специфікації wsdl;

– використовуючи отриманий wsdl файл, необхідно згенерувати сторонні позиції клієнта, які обробляють взаємодію з веб-сервісом;

– необхідно перевірити функціональність веб-сервісу, зробити пробний виклик,передаючи йому необхідні параметри та перевірити чи він відповідає до запитів, надісланих йому правильно [3];

– якщо є будь-які перевірки безпеки, такі як ім'я користувача та пароль, потрібно перевірити їх ефективність. У результаті цього кроку повинно бути порушення в системи і отримання несанкціонованого доступу.

Оскільки організації створюють інтерфейс веб-сервісу та долають їх проблеми безпеки, вони можуть обмінюватися даними з суб'єктами, такими як клієнти, постачальники та партнери в більш непривабливій формі [6]. Для тестування таких спільних веб-сервісів необхідно наступне:

– у системі, де веб-сервіси взаємодіють один з одним, потрібно протестувати складові можливості веб-сервісів;

– для деяких SOAP-повідомлень, може бути призначений одержувач, але також може існувати один або декілька посередників у маршруті повідомлення.SOA тестування

має підтвердити правильність функціонування цих посередників також [4].

У вільно пов'язаному середовищі сервіс-орієнтованої архітектури не повинні знати подробиці один одного, але потрібно, щоб вони мали достатньо інформації для надійного обміну повідомленнями без помилок або непорозуміння. Стандартизовані технічні умови допомагають створити такі умови, але відмінності в реалізації можуть викликати проблеми в спілкування. Сумісність – це коли сервіси можуть взаємодіяти між собою, не стикаючись з такими проблемами. Відсутність типів даних у запиті чи відповіді є стандартом зв'язку між веб-сервісами. SOAP запити – це XML-документи, а XML забезпечує гнучкість щодо типу даних. Гнучкість може стати проблемою для SOAP взаємодії [6]. Великі цифрові типи даних використовуються для представлення великих чисел. Існують відмінності у максимальній точності, що підтримуються основними платформи програмування. Тому може так статися, що запит або відповідь містять номери, що мають вищу точність ніж ті, що можуть оброблятися рідною мовою сервісу або клієнт інтерпретація цих чисел залежить від того, як такі випадки обробляються рідною мовою. Це може мати вплив на програму, де важлива точність, такі як банківська або фінансова програми [6].

Щоб протестувати веб-сервіси на сумісність, потрібно надіслати виклики службі, в якій клієнт надсилає параметр певного типу, і сервер повертає параметр одного і того ж типу та значення. Потім клієнту необхідно перевірити повернуте значення, щоб переконатися, що воно відповідає відправленому значенню [6].

Тестування навантаження повинно проводитися, щоб зрозуміти статистику ефективності веб-сервісу. Тестування навантаження дає нам уявлення про те, як користувачі будуть взаємодіяти з веб-сервісом. Тестування навантаження передбачає потрапляння в веб-сервіс багатьох запитів одночасно і отримання вимірів часу для різних параметрів, таких як час підключення до веб-сервісу та час отримання відповіді від веб-сервісу. Також повинна бути перевірена правильність відповіді у випадку одночасних запитів. При навантажувальному тестуванні ігнорується процес перевірки функціональності та перевіряється виключно швидкість завантаження [3, 6].

Тестова стратегія для веб-сервісів може не відрізнятися від тестової стратегії додатків, але буде виконуватися в іншому інтерфейсі:

– необхідно виконувати тестування меж та даних;

– досліджувати дані, послідовність, умови;

– для перевірки даних необхідно виконати аналіз даних для кожного параметра, виконати еквівалентне розділення класів, потім граничний аналіз значень [6];

– Вимушено перевіряти помилки. Залишити обов'язкові поля порожніми, перевірити необов'язкові поля з даними або без даних, заповнити поля з неправильними

типами даних.

Існують такі типи помилок, як: збій обробки помилкових умов, невикористані прапори, відсутній або дублюється функціонал, труднощі при підключенні і отриманні відповіді від API, проблеми з безпекою, питання многопоточності, проблеми з продуктивністю, некоректна обробка валідних значень, дані відповіді некоректно структуровані [3, 6].

Веб-сервіси складаються з безлічі класів, функцій та процедур, які представляють собою грунт бізнес-логіки. Якщо API не перевіряється належним чином, то це може викликати проблеми не тільки в застосуванні API, але і в викликаних додатках.

2.2 Проблеми у тестуванні веб-сервісів

Вільно пов’язані сервіси без користувацького інтерфейсу можуть викликати проблеми у ході розробки і тестування, такі як масштабованість і безпека, відсутність користувацького інтерфейсу та розповсюдження по мережі, комбінація і вибір параметрів, валідація та верифікація вихідних даних в різних системах, обов’язкова перевірка обробки винятків [3 – 4].

Середовище розробки та розгортання веб-сервісів дуже відрізняються одне від одного. Якщо веб-сервіс використовується для внутрішньої приватної мережі деякої організації, то ми маємо максимальну кількість користувачів, які підключаються до сервісу, а також є контроль над тим, хто може отримати доступ до веб-сервісу, тому ми маємо певну безпеку. Але сценарій для інтернет веб-сервісу інший. Там не можна зробити припущення щодо кількості користувачів, підключених до сервісу, безпеки або способу, за допомогою якого користувачі матимуть доступ до веб-сервісу. Також необхідно знати заздалегідь вплив продуктивності у випадку великої кількості користувачів, що підключаються до веб-сервісу [3].

На відміну від традиційних веб-систем, веб-сервіси не мають інтерфейсу користувача. Тому вони не можуть бути перевірені вручну, але вимагають написання тестових кроків. Для цього тестувальник повинен мати навички програмування та бути ознайомлений з основами веб-сервісів.

Програми зазвичай створюються шляхом інтеграції багатьох веб-сервісів для використання існуючої функціональності веб-сервісу. Ці веб-сервіси можуть бути розроблені одними й тими ж розробниками або можуть бути надані третьою стороною. Тому повинно бути виконано ретельне тестування чорної скриньки. Також ці послуги

поширюються по мережі та можуть бути розміщений на різних операційних системах і розгортатися в різних середовищах. Отже, під час тестування повинні братися до уваги питання доступності, продуктивність, надійності та безпеки [6].

2.3 Інструменти для тестування веб-сервісів

Оскільки веб-сервіси доступні в інтернеті і поширюються у мережах, вони уразливі до вірусів і загроз безпеки, які впливають на процеси, засновані на них. Отже, тестування веб-сервісів або API-інтерфейсів стає необхідним для забезпечення належного функціонування та коректної відповіді на запити [1].

Звичайні GET запити можна надсилати за допомогою браузера. Існує кілька безкоштовних інструментів для тестування веб-сервісів: їх можливостей підключення, відповіді та продуктивності [10]. Вони надають можливість не тільки відправляти різні типи запитів, але і зберігати запити, показувати результати в різних форматах, виступати в ролі proxy сервера, розробляти автоматизовані тести для конкретного тестового сценарію. Серед таких інструментів:

а) Postman – це пагін у Google Chrome, який можна використовувати для відправки POST-запитів до серверу і отримання відповідей. Він дозволяє налаштувати всі заголовки і тимчасові файли, які очікує API, і перевірити відповідь по отриманню. Ключові особливості:

– можна використовувати для автоматизованого і дослідницького тестування;
– можна використовувати для Mac, Windows, Linux;
– має функції запуску, тестування, документації і відстеження;
– дозволяє організовувати API в функції;
– полегшує спільну роботу і спільне використання даних API і засобів контролю;
– дозволяє записувати логічні тести в Postman Interface [3].

б) JMeter – це інструмент, який окрім навантажувального тестування надає можливість функціонального тестування API. Наприклад, він вміє автоматично працювати з csv- файлами, дозволяючи швидко створювати унікальні значення параметрів для тестів. Ключові особливості:

– JMeter повністю написаний на Java. Це десктопна програма, тому її можна запускати на різних платформах;

– результати тестування можуть бути представлені у формі таблиць, системного журналу або дерева файлів;

– інсталяцію робити не потрібно. Необхідно скопіювати і запустити файл * .bat.

– JMeter підтримує багато базових протоколів, таких як LDAP, SOAP, FTP, HTTP [1].

в) Fiddler – це інструмент, який дозволяє відстежувати, змінювати і повторно використовувати http-запити. Для перевірки поведінки веб-сервісів можна використовувати бібліотеку FiddlerCore.NET Class Library і створити інфраструктуру для API-тестування. Ключові особливості:

– дозволяє перевіряти трафік, встановлювати точки зупинки та оперувати вхідними і вихідними даними;

– дозволяє налагоджувати трафік з практично будь-якою програмою, що підтримує проксі, включаючи Internet Explorer, Google Chrome, Apple Safari, Mozilla Firefox, Opera;

– можна налагоджувати трафік із популярних пристроїв, таких як Windows Phone, iPod, iPad [1].

г) SoapUI – це інструмент, який дозволяє автоматизувати функціональні, регресійні, узгоджені та навантажувальні тести як SOAP, так і REST-сервісів. Він простий у використанні та підтримує передові технології та стандарти для моделювання та стимулювання поведінки веб-сервісів. Ключові особливості:

– надає звіти для друку, експорту та html на рівні Project, TestSuite, TestCase або LoadTest;

– є можливість взаємодії з Hudson, Bamboo, Maven, ANT і JUnit;

– дозволяє розробляти власний набір функцій у вигляді плагінів SoapUI;

– записує, контролює і відображає всі дані.

– підтримує ws-security і ssl-розшифровки.

д) Runscope – це інструмент для перевірки і моніторингу продуктивності API. Підтримує тестування API-інтерфейсів і бекенда мобільних додатків. Ключові особливості:

– дозволяє створювати тести з динамічними даними навіть для складних випадків;

– відображає показники і аналітику для виявлення проблем;

– дозволяє повторно використовувати і виконувати тести в декількох місцях;

– полегшує управління тестуванням для поліпшення спільної роботи [3].

е) SOAtest – це інструмент для тестування і перевірки API-інтерфейсів і додатків, керованих API. Він забезпечує надійну підтримку функціонального блоку, інтеграцію, безпеку, симуляцію, проведення навантажувального тестування за допомогою таких технологій, як REST, JSON, MQ, JMS, TIBCO, HTTP і XML. Ключові особливості:

– забезпечує end-to-end тестування;

– підтримує більше ста двадцяти протоколів та типів повідомлень;

– дозволяє створювати складні та багаторазові тести без кодування;

– підтримує безперервне інтеграційне тестування.

є) vRest – це інструмент, призначений для тестування REST APIS і веб-сервісів. Він також підтримує тестування веб-систем, мобільних і десктопних додатків, які взаємодіють зі сторонніми API-інтерфейсами або службами HTTP. Ключові особливості:

– має розширення Chrome для запису і відтворення тестових кроків;

– дозволяє експортувати та імпортувати тест-кейси і звіти з зовнішніх інструментів;

– підтримує інтеграцію з Jenkins для безперервної роботи серверів і Jira для відстеження помилок [5].

ж) TestingWhiz – це інструмент автоматизації тестування, сумісний з веб-сервісами. Він дозволяє проводити функціональне тестування, тестування сумісності, тестування навантаження і працювати з веб-сервісами REST і SOAP за допомогою WSDL-інтерфейсу через HTTP і FTP. Ключові особливості:

– підтримує порівняння рядків для перевірки відповіді API;

– допомагає у пошуку дефектів за допомогою інтегрованих інструментів відслідковування помилок, таких як Jira, Mantis і Fogbugz;

– створює звіти про проведення тесту за допомогою електронної пошти;

– підтримує тестування, засноване на даних і ключових словах [1].

В якості тестового середовища було вирішено застосувати платформу для автоматизації тестування SoapUI.

2.4 SoapUI платформа для автоматизації тестування веб-сервісів

SoapUI – це програма для тестування веб-сервісів з відкритим кодом для сервіс-орієнтованих архітектур (SOAP) та державних переказів (REST). Завдяки зручному використанню графічного інтерфейсу та особливостям корпоративного класу, SoapUI дозволяє легко та швидко створювати та виконувати функціональні, регресійні та навантажувальні тести [10]. У єдиному середовищі тестування SoapUI забезпечується повне тестування - від веб-сервісів SOAP і REST до корпоративних повідомлень JMS Enterprise, баз даних, Rich Internet Applications та багато іншого [5].

У SoapUI робота організована в проекти, які відображаються під кореневим вузлом у навігаторі робочого середовища. Проект може містити будь-яку кількість

функціональних тестів, тестів навантаження та моделювання служб, необхідних для цілей тестування [2].

SoapUI пропонує два формати проекту: автономні проекти та складні проекти. Формат проекту за замовчуванням є автономним. SoapUI зберігає їх у вигляді єдиного XML-файлу, який містить всі артефакти проекту, такі як інтерфейси, тести, mock служби, скрипти тощо.

У SoapUI існує три основних типи проектів:

– SOAP проекти;
– REST проекти;
– загальні проекти.

SOAP проекти створюються з файлу WSDL або одного службового виклику. Можна використовувати ці проекти, щоб протестувати кожен аспект сервісів SOAP, перевіряти, чи підтримують служби стандарти, такі як WS-Security, WS-адресація, створювати тести функціональності та навантаження [5].

REST проекти створюються з файлу WADL або, безпосередньо, URI та його параметрів. Можна використовувати ці проекти, щоб протестувати служби RESTful, створювати різні запити та перевіряти отриману інформацію, спробувати використовувати безліч методів та операцій [2, 5].

Загальні проекти є універсальним проектом для сервісів з різноманітними інтерфейсами та методами. Вони можуть поєднувати тести, які ви створюєте для служб REST та SOAP, а також тестування на основі даних, mockups та всієї функціональності, яку надає SoapUI [5].

SoapUI – це інструмент, який має такі особливості тестування API з відкритим кодом:

– функціональне тестування;
– моделювання служб;
– тестування безпеки;
– навантажувальне тестування;
– технологічна підтримка;
– автоматизоване тестування;
– аналітика.

Функціональне тестування у SoapUI являє собою тести, кожен з яких має три рівня: набір тест-кейсів, тест-кейси та тестові кроки, які створюються всередині проекту SoapUI для підтримки масових сценаріїв тестування.

Моделювання служб (Mocking) дозволяє імітувати роботу певного веб-сервіса і створити надійні прототипи, перш ніж вони будуть реалізовані. Вони усувають витрати на

побудову повноцінних копій виробничих систем і дають змогу споживачам отримувати доступ до цих послуг, не чекаючи того, що вони будуть побудовані або доступні. Можна імітувати будь-яку бажану поведінку, незалежно від того, наскільки вона складна, і повністю налаштувати сервісні відповіді [2].

Тестування безпеки у SoapUI використовується для сканування цільових служб для загальних вразливостей безпеки, наприклад, SQL Injections і XML Bombs. Тести безпеки додаються до тест-кейсів та після запуску виконують фактичне сканування та виявлення уразливості [5].

Навантажувальне тестування у SoapUI базується на існуючих функціональних тестових програмах. Навантажувальні тести багаторазово запускають тест-кейси протягом заданого часу з потрібною кількістю потоків (або віртуальних користувачів). Навантажувальні тести також додаються до тест-кейсів. Після запуску тестів навантаження можна перевірити продуктивність веб-сервіса за допомогою різних стратегій завантаження, перевірити, чи його функціональні можливості не порушуються під навантаженням, одночасно запускати декілька тестів навантаження, щоб побачити, як вони впливають один на одного [2, 5].

За допомогою сучасних технологій SoapUI підтримує всі загальні протоколи та стандарти.

Автоматизоване тестування у SoapUI здійснюється за допомогою передових автоматичних функцій, що дозволяють різко скоротити витрати на робочу силу. Для створення автоматизованих тестів використовуються інструменти командного рядка в комплекті з SoapUI. Є можливість налаштувати виконання тесту, щоб перекрити параметри тесту, контролювати, які тести виконувати або виводити [5].

За допомогою потужної та інтегрованої аналітики SoapUI робить тестування швидше, ніж коли-небудь, і заощаджує безліч годин. Є можливість створюйте всеосяжні, але прості для розуміння звіти для функціональних та тестових навантажень з інтерфейсу користувача на рівні проекту. Можна експортувати свої звіти в будь-який стандартний формат і налаштовувати їх будь-яким засобом.

SoapUI – найпоширеніший в світі інструмент автоматизованого тестування для SOAP та REST API. Щоб мати глибокий досвід в області автоматизації тестування веб-сервісів потрібно використовувати саме цей інструмент для запису, запуску, інтеграції та автоматизації розширених тестів API [2, 5].

2.5 Налаштування середовища SoapUI

В якості тестового середовища використовується SoapUI 5.1.3. SoapUI має як комерційну, так і безкоштовну версію для роботи з веб-сервісами. Завантажити необхідну версію програми можна на офіційній сторінці продукту. Завантаження файлу виконується для 64 бітної операційної системи Windows [10]. Після завантаження, файл слід розпакувати використовуючи функції адміністратору комп'ютера, прийняти ліцензійні умови та вибрати папку для розташування програми. Після установки програма доступна на робочому столі комп'ютера. На рисунку 2.1 зображено головне вікно SoapUI після запуску [5]. При першому відкритті SoapUI не буде завантажено жодного проекту.

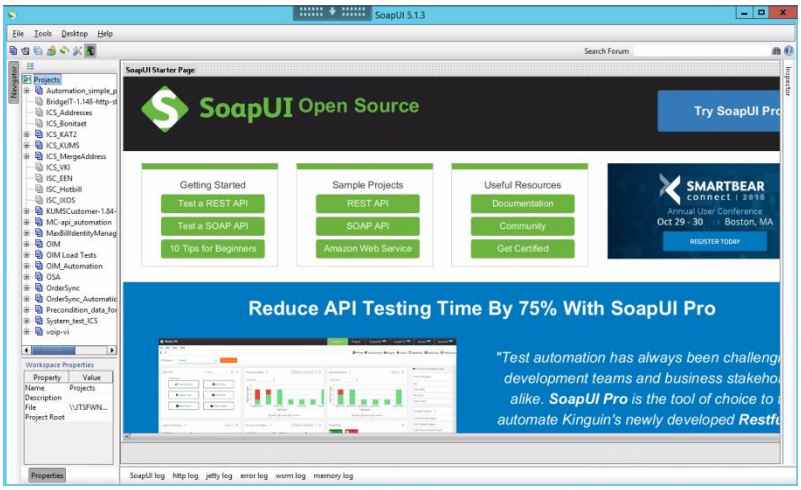


Рисунок 2.1 – Головне вікно SoapUI після запуску

SoapUI – це лідер ринку серед інструментів тестування API [2]. У SoapUI створити проект можна двома засобами:

– створити новий проект;

– імпортувати існуючий проект.

Для того, щоб створити автоматизовані тести необхідно імпортувати XML файл відповідного SOAP проекту, для якого будуть додаватися автоматизовані тести (рис. 2.2).

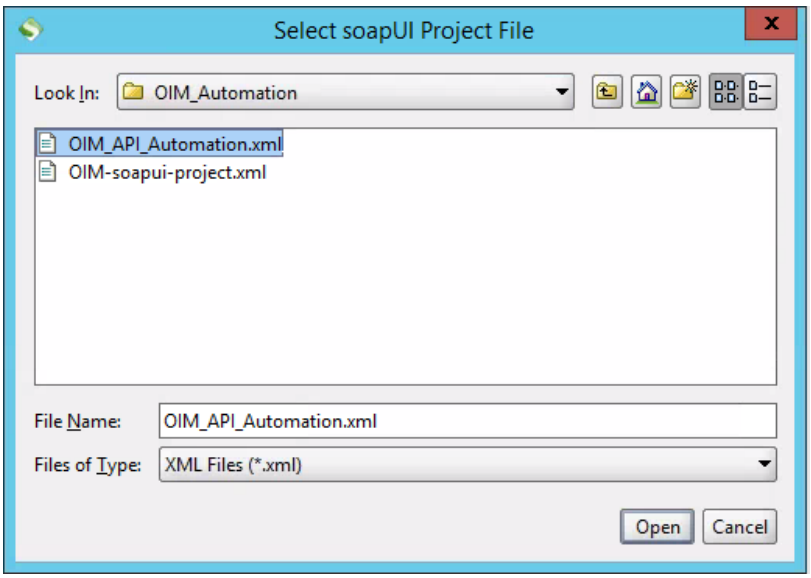


Рисунок 2.2 – Імпорт XML файлу у SoapUI

Для того, щоб створити новий SOAP проект необхідно вказати назву нового проекту та файл WSDL, який SoapUI використовуватиме для початкової конфігурації, а потім вибрати потрібні параметри (рис. 2.3).

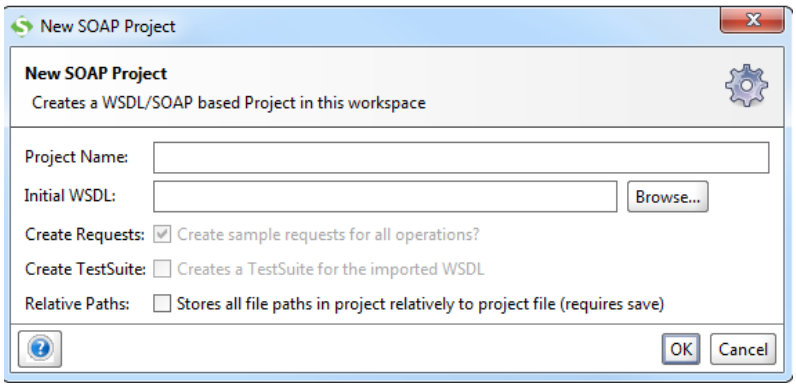


Рисунок 2.3 – Створення нового SOAP проекту

Після імпорту XML файлу необхідно створити набір тест-кейсів, самі тест-кейси та тестові кроки (рис. 2.4).

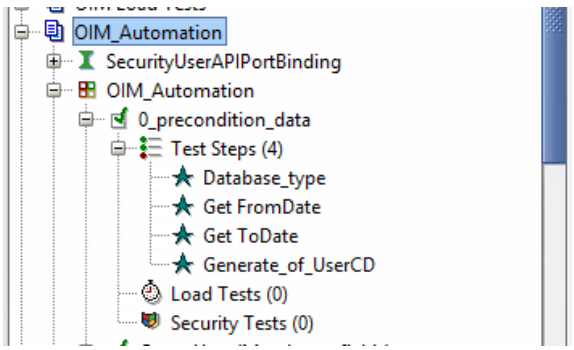


Рисунок 2.4 – Створення набору тест-кейсів і тестових кроків

SoapUI підтримує такий набір тестових кроків, як:

а) SOAP Request;

б) REST Request;

в) HTTP Request;

г) AMF Request;

д) JDBC Request;

Кожен request має формат запиту та відповіді, який відкривається у відповідному редакторі запиту та має XML структуру (рис. 2.5). Для виклику операції можна додавати будь-яку кількість об'єктів запиту до операції в дереві навігатора [10]. SoapUI за замовчуванням створює приклад для кожної операції під час імпорту. Щоб отримати відповідь на запит, необхідно запустити операцію виконання запиту. Подібний редактор доступний для усіх типів запитів [5].

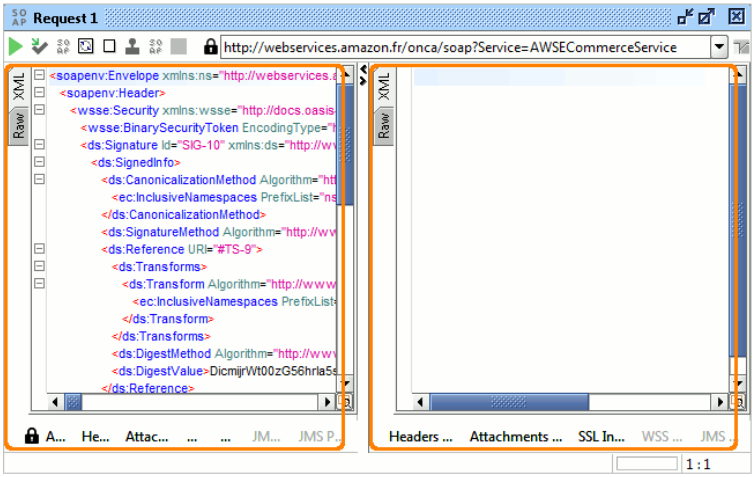


Рисунок 2.5 – Редактор запиту у SoapUI

е) Properties;

Властивості тестових кроків використовуються для визначення користувацьких властивостей, які будуть використовуватися в тест-кейсі (рис. 2.6). Основними перевагами перед визначенням властивостей на рівні тест-кейсу є:

– можна організувати властивості в декількох властивостях тестових кроків;

– можна вказати вихідні та цільові імена файлів, які будуть використовуватися для читання та запису зазначених властивостей при виконанні тестових кроків [2, 5].

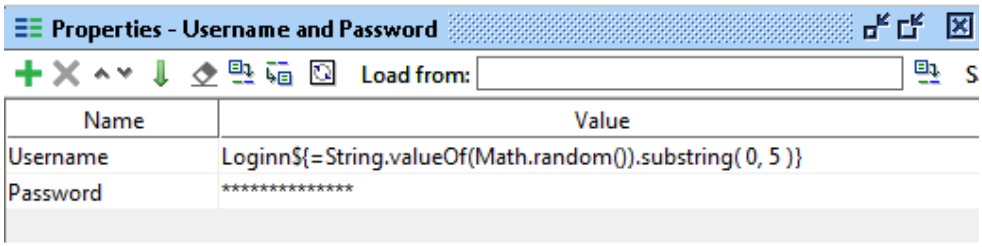


Рисунок 2.6 – Властивості тестових кроків у SoapUI

є) Property transfer;

Передача властивостей тестових кроків використовуються для передачі властивостей між тестовими кроками та їх вмістом тест-кейсів, набору тест-кейсів та проекту (рис. 2.7). Вони надзвичайно корисні в ряді ситуацій, особливо якщо вони включають властивості, що містять XML, наприклад, коли вам потрібно:

– витягнути значення з повідомлення XML;
– написати значення в XML-повідомлення;
– передати складний XML зміст між властивостями [5].

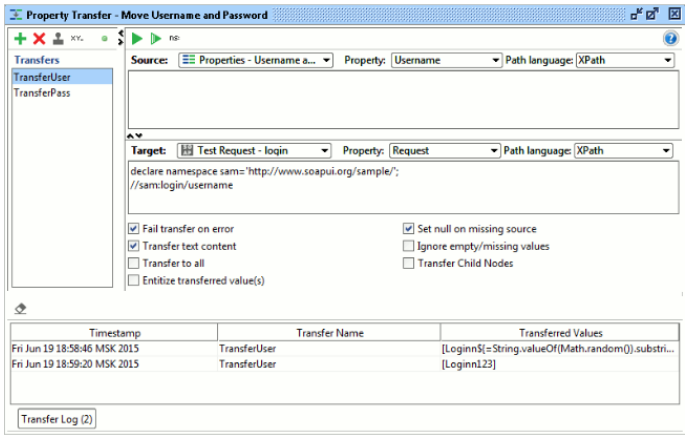


Рисунок 2.7 – Передача властивостей у SoapUI

ж) Conditional Goto;

Умовний Goto містить довільну кількість виразів XPath до відповідного тестового кроку (рис. 2.8). Вони застосовуються до найближчої відповіді попереднього тестового кроку; налаштовані вирази XPath застосовуються зверху до низу, а коли вираз XPath дає значення true, умовне Goto переведе виконання до вказаного тестового кроку [2].

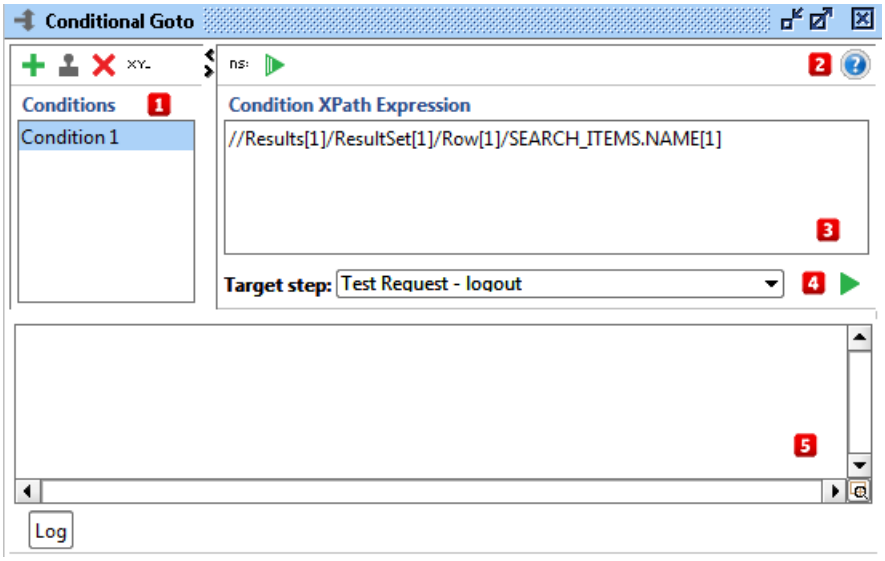


Рисунок 2.8 – Умовний Goto у SoapUI

з) Run TestCase;

Щоб отримати результат тест-кейсу з усіма тестовими кроками, необхідно запустити тест-кейс на виконання (рис. 2.9).

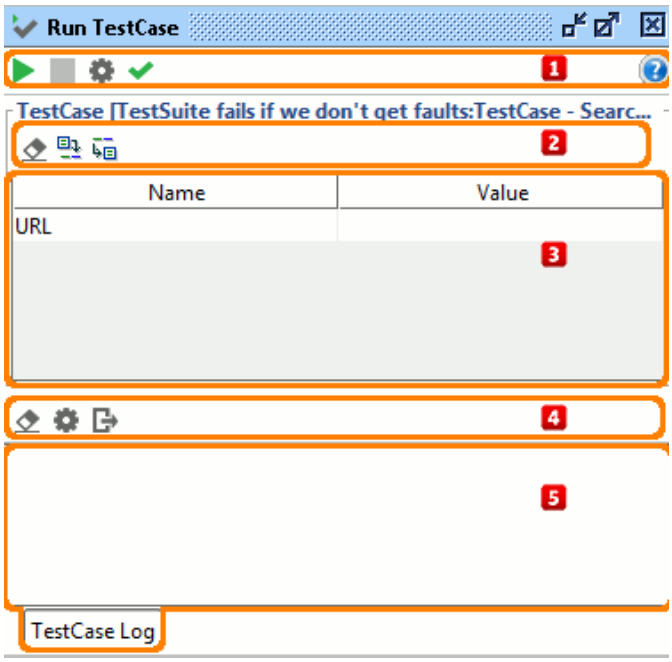


Рисунок 2.9 – Запуск тест-кейсу у SoapUI

и) Groovy Script;

Groovy - це скриптова мова, яка внутрішньо включає в себе всі бібліотеки Java, тому всі ключові слова та функції, пов'язані з java, можуть бути використані безпосередньо в groovy script (рис. 2.10). Бібліотеки Java поставляються з SoapUI і інтегровані під час самого встановлення SoapUI [10].

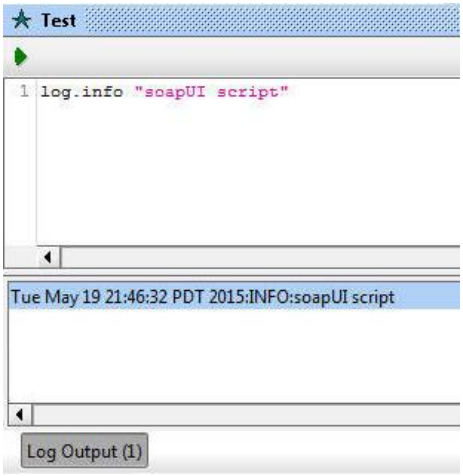


Рисунок 2.10 – Groovy Script у SoapUI

і) Delay;

Затримка тестового кроку затримує виконання тест-кейсу за заданою кількістю мілісекунд (рис. 2.11). У списку тестових кроків мітка відображає назву затримки і її значення, яке буде йти у зворотному напрямку під час його виконання [5].

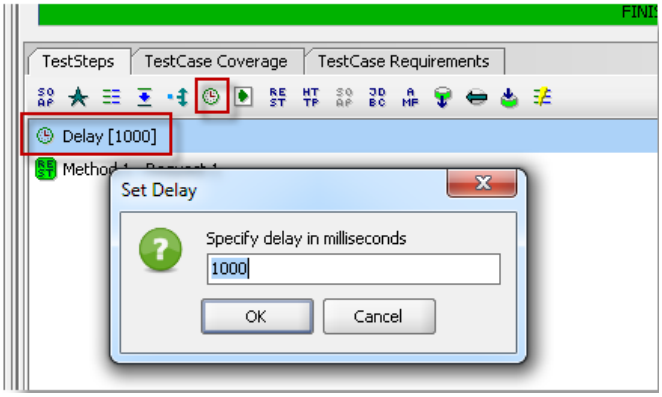


Рисунок 2.11 – Затримка у SoapUI

ї) Soap Mock Response;

Mock Response містить повідомлення, яке Mock Service повинен повернути на виклик клієнта (рис. 2.12). Mock Service може містити користувальницький вміст, заголовки та вкладення, що дозволяє імітувати будь-який тип дійсної (або невірної) HTTP-відповіді, а додаткові можливості сценаріїв дозволяють легко додавати динамічний вміст до вихідної відповіді.

Це в основному те ж саме, що і стандартне вікно редактора запиту SOAP, але орієнтація на користувачів тепер знаходиться у правій частині редактора [2]. Тут налаштується відповідне повідомлення, яке потрібно буде повернути, включаючи власні заголовки HTTP та вкладення через інспекторів у нижній частині екрана, так само як і в редакторі запиту, налаштуйте ваші повідомлення запитів. Ліва частина вікна показує

останній запит, фактично відправлений до цієї конкретної відповіді, з усіма можливостями перегляду вхідних заголовків HTTP, вкладень [2, 5].

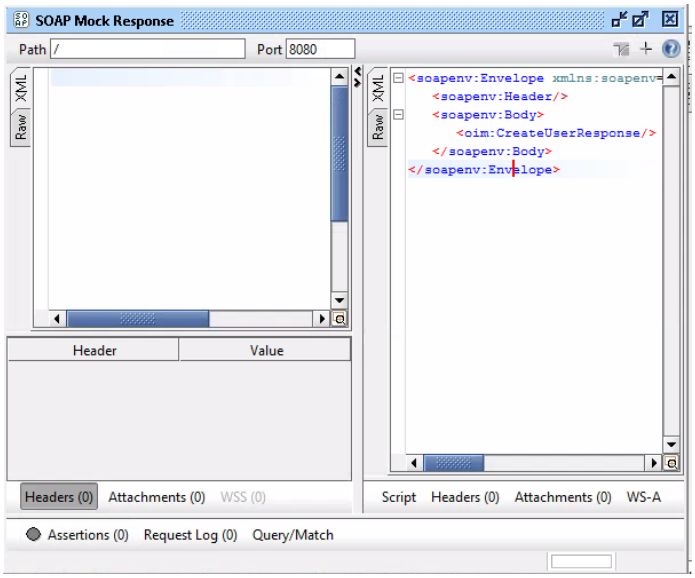


Рисунок 2.12 – Mock Response у SoapUI

й) Manual TestStep.

Ручний тестовий крок використовується, коли ми хочемо взаємодіяти з користувачами в тесті. Ці тест-кейси, які мають ручні тестові кроки, пропускаються при запуску через text runner, оскільки взагалі неможливо взаємодіяти з спливаючими вікнами (рис. 2.13).

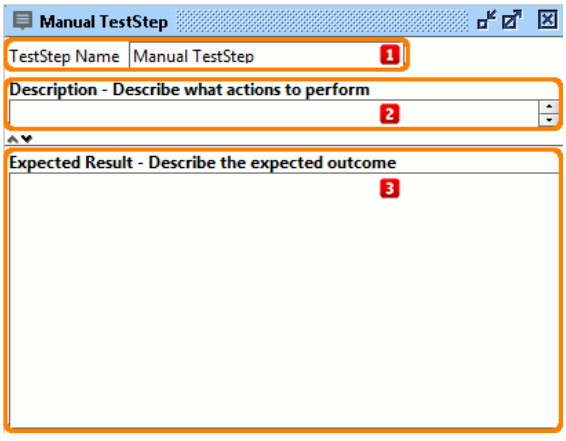


Рисунок 2.13 – Manual TestStep у SoapUI

У ході створення автоматизованих тестів слід пам’ятати про такі умови:

– тест-кейси повинні бути згруповані за тестовими категоріями;
– кожен тест повинен описувати тестовані функції;
– вибір параметрів повинен бути явно згаданий в самому тесті;

– встановити пріоритети виклику функцій API;

– кожен тест повинен бути самодостатнім і незалежним один від одного;

– особливої обережності слід дотримуватися при зверненні до функцій видалення, закриття вікна;

– виклик послідовності дій повинен бути добре спланований і здійснений;

– для забезпечення повного тестового покриття необхідно створювати тестові випадки для всіх можливих комбінацій вхідних даних [5].

Дослідження якості буде проводитися для одного веб-сервіса. В якості веб-сервіса було прийнято рішення протестувати і перевірити якість інтерфейсу Oracle Identity Manager.

3 АНАЛІЗ РЕЗУЛЬТАТІВ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБ-СЕРВІСІВ

3.1 Написання тестових кроків

Інтерфейс Oracle Identity Manager – це веб-сервіс, який призначений для управління користувачами у веб-додатку. Щоб перевірити якісь веб-сервісу, треба перевірити працездатність його функцій. Тестові сценарії веб-сервісу, які необхідно перевірити:

– перевірка створення користувача;

– перевірка можливості зміни параметрів користувача;

– додавання списку груп користувачу;

– перевірка можливості зміни статусу користувача;

– отримання повної інформації про створеного користувача, про користувача зі зміненими параметрами, про групу користувача.

3.1.1 Написання тестових кроків для перевірки створення користувача

Для того, щоб перевірити можливість створення користувача, було написано декілька позитивних та негативних тест кейсів, які представлені відповідно на рисунках 3.1 - 3.2.

Project	Use Case	Test Case ID	Test Case Name	Test Case Description	Expected Result	Version
0059 Oracle Identity Manager Integration (finished)	UC_205106329_4	TC_212867075_32	Create new user account Note: operation <*CreateUser*>	Precondition: 1) Operation <*CreateUser*> is registered in ESB; 2) The User with the same User Code is not exist; 3) Not to specify list of security groups in the request. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	1) The new SecurityUser's account with specified attributes is available in the **securityuser** table. Note: The <*Last Modified By*> contains supplied OIM user.	R7.8.0

Рисунок 3.1 – Позитивний тест кейс для перевірки створення користувача у OIM

0059 Oracle Identity Manager Integration (finished)	UC_205106329_4	TC_212867075_41	Create new user account **if the same User Code is exist** Note: operation *<CreateUser>*	Precondition: 1) Operation *<CreateUser>* is registered in ESB; 2) The User with the same User Code is exist; 3) Not to specify list of security groups in the request. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	The following error message is inserted into log file: Message Code: **E98002** Message Text: **User "user from request" already exists**

Рисунок 3.2 – Негативний тест кейс для перевірки створення користувача у OIM

Умови для виконання першого тесту наступні:

– функція <CreateUser> зареєстрована у системі ESB;

– користувач з таким <UserID> відсутній у базі даних;

– список <UserGroups> відсутній у запиті WSDL файлу.

Кроки першого тесту наступні:

– відправляємо WSDL файл на обробку з заповненими обов’язковими полями.

За допомогою першого тесту перевіряється можливість створення користувача. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– новий користувач присутній у таблиці <securityuser> бази даних;

– поле <LastModifiedBy> містить інформацію про систему, яка створила користувача.

Умови для виконання другого тесту наступні:

– функція <CreateUser> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– список <UserGroups> відсутній у запиті WSDL файлу.

Кроки другого тесту наступні:

– відправляємо WSDL файл на обробку з заповненими обов’язковими полями.

За допомогою другого тесту перевіряється можливість створення користувача, якщо такий <UserID> присутній у базі даних. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу містить помилку «Користувач вже існує»;

– новий користувач не присутній у таблиці <securityuser> бази даних.

Умови для виконання третього тесту наступні:

– функція <CreateUser> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– список <UserGroups> відсутній у запиті WSDL файлу.

Кроки третього тесту наступні:

– відправляємо WSDL файл на обробку з заповненими обов’язковими полями.

За допомогою третього тесту перевіряється можливість створення користувача, якщо список <UserGroups> відсутній у запиті WSDL файлу. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– новий користувач не присутній у таблиці <securityuser> бази даних;

– список <UserGroups> відсутній.

3.1.2 Написання тестів для перевірки можливості зміни параметрів користувача

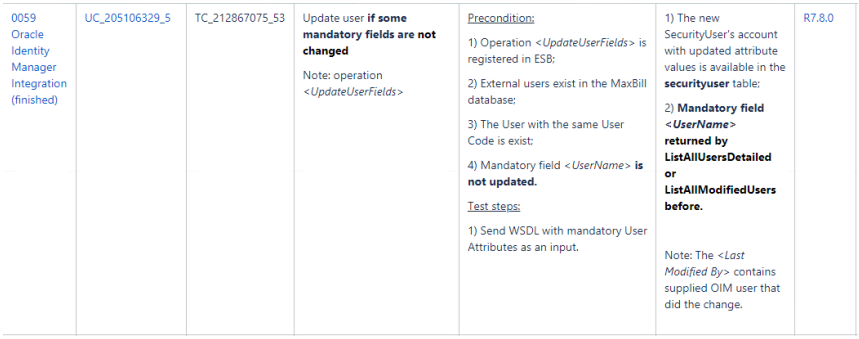
Для того, щоб перевірити можливість зміни параметрів користувача, було написано декілька позитивних та негативних тест кейсів, які представлені відповідно на рисунках 3.3 - 3.6.

0059 Oracle Identity Manager Integration (finished)	UC_205106329_5	TC_212867075_43	Update user Note: operation <*UpdateUserFields*>	Precondition: 1) Operation <*UpdateUserFields*> is registered in ESB; 2) External users exist in the MaxBill database; 3) The User with the same User Code is exist. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	1) The new SecurityUser's account with updated attribute values is available in the **securityuser** table. Note: The <*Last Modified By*> contains supplied OIM user that did the change.	R7.8.0

Рисунок 3.3 – Позитивний тест кейс для перевірки зміни параметрів користувача у OIM

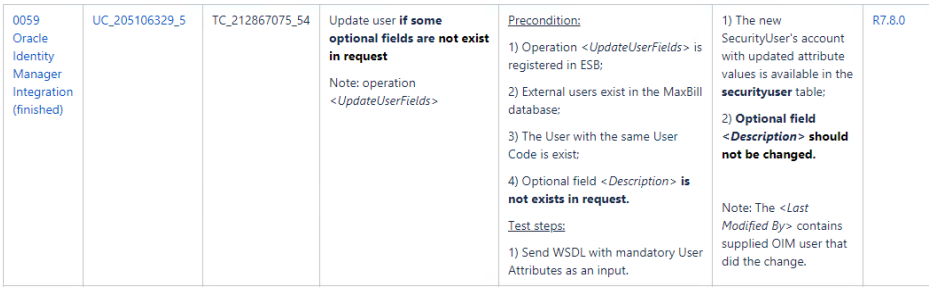
0059 Oracle Identity Manager Integration (finished)	UC_205106329_5	TC_212867075_52	Update user **if the same User Code is not exist** Note: operation <*UpdateUserFields*>	Precondition: 1) Operation <*UpdateUserFields*> is registered in ESB; 2) External users exist in the MaxBill database; 3) The User with the same User Code is **not** exist. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	The following error message is inserted into log file: Message Code: **E98003** Message Text: **User "user from request" not found**	R7.8.0

Рисунок 3.4 – Негативний тест кейс для перевірки зміни параметрів користувача у OIM



0059 Oracle Identity Manager Integration (finished)	UC_205106329_5	TC_212867075_53	Update user **if some mandatory fields are not changed** Note: operation *<UpdateUserFields>*	Precondition: 1) Operation *<UpdateUserFields>* is registered in ESB; 2) External users exist in the MaxBill database; 3) The User with the same User Code is exist; 4) Mandatory field *<UserName>* **is not updated.** Test steps: 1) Send WSDL with mandatory User Attributes as an input.	1) The new SecurityUser's account with updated attribute values is available in the **securityuser** table; 2) **Mandatory field *<UserName>* returned by ListAllUsersDetailed or ListAllModifiedUsers before.** Note: The *<Last Modified By>* contains supplied OIM user that did the change.	R7.8.0

Рисунок 3.5 – Негативний тест кейс для перевірки обов'язкових параметрів користувача у OIM, якщо поле не було змінено



0059 Oracle Identity Manager Integration (finished)	UC_205106329_5	TC_212867075_54	Update user **if some optional fields are not exist in request** Note: operation *<UpdateUserFields>*	Precondition: 1) Operation *<UpdateUserFields>* is registered in ESB; 2) External users exist in the MaxBill database; 3) The User with the same User Code is exist; 4) Optional field *<Description>* **is not exists in request.** Test steps: 1) Send WSDL with mandatory User Attributes as an input.	1) The new SecurityUser's account with updated attribute values is available in the **securityuser** table; 2) **Optional field *<Description>* should not be changed.** Note: The *<Last Modified By>* contains supplied OIM user that did the change.	R7.8.0

Рисунок 3.6 – Негативний тест кейс для перевірки не обов'язкових параметрів користувача у OIM у разі їх відсутності у запиті

Умови для виконання першого тесту наступні:

– функція <UpdateUserFields> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних.

Кроки першого тесту наступні:

– відправляємо WSDL файл на обробку з зміненими заповненими обов'язковими полями.

За допомогою першого тесту перевіряється можливість зміни параметрів користувача. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– користувач з зміненими обов'язковими полями присутній у таблиці <securityuser> бази даних;

– поле <LastModifiedBy> містить інформацію про систему, яка зробила зміни параметрів користувача.

Умови для виконання другого тесту наступні:

– функція < UpdateUserFields> зареєстрована у системі ESB;

– користувач з таким <UserID> відсутній у базі даних.

Кроки другого тесту наступні:

– відправляємо WSDL файл на обробку з зміненими заповненими обов'язковими

полями.

За допомогою другого тесту перевіряється можливість зміни параметрів користувача, якщо такий <UserID> відсутній у базі даних. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу містить помилку «Користувача не знайдено;

– користувач з зміненими обов'язковими полями не присутній у таблиці <securityuser> бази даних.

Умови для виконання третього тесту наступні:

– функція < UpdateUserFields> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– обов'язкове поле <UserName> не змінено у запиті WSDL файлу.

Кроки третього тесту наступні:

– відправляємо WSDL файл на обробку з зміненими заповненими обов'язковими полями, окрім поля <UserName>.

За допомогою третього тесту перевіряється можливість зміни параметрів користувача, якщо обов'язкове поле <UserName> не змінено у запиті WSDL файлу. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– користувач з зміненими заповненими обов'язковими полями присутній у таблиці <securityuser> бази даних;

– поле <UserName> містить інформацію, яка повертається функціями <ListAllUsersDetailed> або <ListAllModifiedUsers>;

– поле <LastModifiedBy> містить інформацію про систему, яка зробила зміни параметрів користувача.

Умови для виконання четвертого тесту наступні:

– функція < UpdateUserFields> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– не обов'язкове поле <Description> відсутнє у запиті WSDL файлу.

Кроки четвертого тесту наступні:

– відправляємо WSDL файл на обробку з зміненими заповненими обов'язковими полями, окрім поля <Description>.

За допомогою четвертого тесту перевіряється можливість зміни параметрів користувача, якщо не обов'язкове поле <Description> відсутнє у запиті WSDL файлу. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– користувач з зміненими заповненими обов'язковими полями присутній у таблиці

<securityuser> бази даних;

– поле <Description> повертається у відповіді WSDL файлу з незміненою інформацією;

– поле <LastModifiedBy> містить інформацію про систему, яка зробила зміни параметрів користувача.

3.1.3 Написання тестів для додавання списку груп користувачу

Для того, щоб перевірити можливість додавання списку груп користувачу, було написано декілька позитивних та негативних тест кейсів, які представлені відповідно на рисунках 3.7 - 3.12.

0059 Oracle Identity Manager Integration (finished)	UC_205106329_6	TC_212867075_55	Assign a Security Group List to the user Note: operation *<SetUserGroup>*	Precondition: 1) Operation *<SetUserGroup>* is registered in ESB; 2) External users exist in the MaxBill database; 3) Groups configured in the MaxBill DB; 4) The User with the same User Code is exist. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	**1) To field USR_Grouplist is added the list of groups in the SECURITYUSER table;** 2) The SecurityUser's account is included to the specified list of Security Groups in the SECURITYUSERTOGRP table. Note: The *<Last Modified By>* contains supplied OIM user that did the change.	R7.8.0

Рисунок 3.7 – Позитивний тест кейс для перевірки додавання списку груп користувачу у OIM

0059 Oracle Identity Manager Integration (finished)	UC_205106329_6	TC_212867075_60	Assign a Security Group List to the user **if the same User Code is not exist** Note: operation *<SetUserGroup>*	Precondition: 1) Operation *<SetUserGroup>* is registered in ESB; 2) External users exist in the MaxBill database; 3) Groups configured in the MaxBill DB; 4) The User with the same User Code is **not** exist. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	The following error message is inserted into log file: Message Code: **E98003** Message Text: **User "user from request" not found**	R7.8.0

Рисунок 3.8 – Негативний тест кейс для перевірки додавання списку груп користувачу у OIM, якщо такий користувач не існує

0059 Oracle Identity Manager Integration (finished)	UC_205106329_6	TC_212867075_61	Assign a Security Group List to the user **if the Security Groups are not exist** Note: operation *<SetUserGroup>*	Precondition: 1) Operation *<SetUserGroup>* is registered in ESB; 2) External users exist in the MaxBill database; 3) Groups **are not** configured in the MaxBill DB; 4) The User with the same User Code is exist. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	The following error message is inserted into log file: Message Code: **E98004** Message Text: **Group "group from request" not found**	R7.8.0

Рисунок 3.9 – Негативний тест кейс для перевірки додавання списку груп користувачу OIM, якщо група не існує

0059 Oracle Identity Manager Integration (finished)	UC_205106329_6	TC_212867075_58	The **<UserGroups> <Group>** list is empty Note: operation *<SetUserGroup>*	Precondition: 1) Operation *<SetUserGroup>* is registered in ESB; 2) External users exist in the MaxBill database; 3) Groups configured in the MaxBill DB; 4) The User with the same User Code is exist. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	1) The interface cleans up the old list of Security Groups for the Security User's account in the SECURITYUSERTOGRP table. Note: The *<Last Modified By>* contains supplied OIM user that did the change.	R7.8.0

Рисунок 3.10 – Негативний тест кейс для перевірки додавання списку груп користувачу у OIM, якщо список груп порожній у запиті

0059 Oracle Identity Manager Integration (finished)	UC_205106329_6	TC_212867075_63	Assign **the same** Security Group List to the user Note: operation *<SetUserGroup>*	Precondition: 1) Operation *<SetUserGroup>* is registered in ESB; 2) External users exist in the MaxBill database; 3) **The same Group exists in the Group list in request**; 4) The User with the same User Code is exist. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	The following error message is inserted into log file: Message Code: **E98005** Message Text: **List of groups contains duplicates**	R7.8.0

Рисунок 3.11 – Негативний тест кейс для перевірки додавання списку груп користувачу, якщо користувач вже має таку групу

0059 Oracle Identity Manager Integration (finished)	UC_205106329_6	TC_212867075_64	Assign an **one** Security Group to the user Note: operation <*SetUserGroup*>	Precondition: 1) Operation <*SetUserGroup*> is registered in ESB; 2) External users exist in the MaxBill database; 3) Group configured in the MaxBill DB; 4) The User with the same User Code is exist. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	1) The interface cleans up the old list of Security Groups for the Security User's account and assigns the supplied in the request list of groups in the SECURITYUSERTOGRP table; 2) The SecurityUser's account is included to the specified Security Group in the SECURITYUSERTOGRP table. Note: The <*Last Modified By*> contains supplied OIM user that did the change.	R7.8.0

Рисунок 3.12 – Позитивний тест кейс для перевірки додавання однієї групи користувачу у OIM

Умови для виконання першого тесту наступні:

– функція <SetUserGroup> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– групи з такими <UserGroups> присутні у базі даних.

Кроки першого тесту наступні:

– відправляємо WSDL файл на обробку зі списком груп, які необхідно додати користувачу.

За допомогою першого тесту перевіряється можливість додавання списку груп користувачу. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– інтерфейс видаляє групи користувача з таблиці <securityusertogrp>;

– користувач з новими групами присутній у таблиці <securityusertogrp> бази даних;

– поле <LastModifiedBy> містить інформацію про систему, яка додала список груп користувачу.

Умови для виконання другого тесту наступні:

– функція <SetUserGroup> зареєстрована у системі ESB;

– користувач з таким <UserID> відсутній у базі даних;

– групи з такими <UserGroups> присутні у базі даних.

Кроки другого тесту наступні:

– відправляємо WSDL файл на обробку зі списком груп, які необхідно додати користувачу.

За допомогою другого тесту перевіряється можливість додавання списку груп

користувачу, якщо такий <UserID> відсутній у базі даних. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу містить помилку «Користувача не знайдено;

– користувач з новими групами відсутній у таблиці <securityusertogrp> бази даних.

Умови для виконання третього тесту наступні:

– функція <SetUserGroup> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– групи з такими <UserGroups> відсутні у базі даних.

Кроки третього тесту наступні:

– відправляємо WSDL файл на обробку зі списком груп, які необхідно додати користувачу.

За допомогою третього тесту перевіряється можливість додавання списку груп користувачу, якщо групи з такими <UserGroups> відсутні у базі даних. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу містить помилку «Група не знайдена»;

– користувач з новими групами відсутній у таблиці <securityusertogrp> бази даних.

Умови для виконання четвертого тесту наступні:

– функція <SetUserGroup> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– список груп з такими <UserGroups> порожній у запиті.

Кроки четвертого тесту наступні:

– відправляємо WSDL файл на обробку з порожнім списком груп.

За допомогою четвертого тесту перевіряється можливість додавання списку груп користувачу. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– інтерфейс видаляє групи користувача з таблиці <securityusertogrp>;

– користувач з новими групами відсутній у таблиці <securityusertogrp> бази даних;

– поле <LastModifiedBy> містить інформацію про систему, яка додала список груп користувачу.

Умови для виконання п’ятого тесту наступні:

– функція <SetUserGroup> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– список груп з такими <UserGroups> має дублікати у запиті.

Кроки п’ятого тесту наступні:

– відправляємо WSDL файл на обробку з списком груп, який має дублікати у запиті.

За допомогою п’ятого тесту перевіряється можливість додавання списку груп користувачу, якщо список містить дублікати у запиті. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу містить помилку «Список груп містить дублікати»;

– користувач з новими групами відсутній у таблиці <securityusertogrp> бази даних.

Умови для виконання шостого тесту наступні:

– функція <SetUserGroup> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– список груп з такими <UserGroups> містить одну групу.

Кроки шостого тесту наступні:

– відправляємо WSDL файл на обробку з однією групою у списку.

За допомогою шостого тесту перевіряється можливість додавання однієї групи користувачу. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– інтерфейс видаляє групи користувача з таблиці <securityusertogrp>;

– користувач з новою групою присутній у таблиці <securityusertogrp> бази даних;

– поле <LastModifiedBy> містить інформацію про систему, яка додала групу користувачу.

3.1.4 Написання тестів для перевірки можливості зміни статусу користувача

Для того, щоб перевірити можливість зміни статусу користувача, було написано декілька позитивних та негативних тест кейсів. Кейси представлені відповідно на рисунках 3.13 - 3.16.

0059 Oracle Identity Manager Integration (finished)	UC_205106329_7	TC_212867075_65	Activate user Note: operation <*SetUserStatus*>	Precondition: 1) Operation <*SetUserStatus*> is registered in ESB; 2) External users exist in the MaxBill database; 3) The User with the same User Code is exist; 4) The field <*UserState*> contains value <*Disabled*>. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	1) The SecurityUser's account is activated in the **securityuser** table. Note: The <*Last Modified By*> contains supplied OIM user that did the change.	R7.8.0

Рисунок 3.13 – Позитивний тест кейс для активації користувача у OIM

0059 Oracle Identity Manager Integration (finished)	UC_205106329_7	TC_212867075_66	Deactivate user Note: operation <*SetUserStatus*>	Precondition: 1) Operation <*SetUserStatus*> is registered in ESB; 2) External users exist in the MaxBill database; 3) The User with the same User Code is exist; 4) The field <*UserState*> contains value <*Enabled*>. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	1) The SecurityUser's account is deactivated in the **securityuser** table. Note: The <*Last Modified By*> contains supplied OIM user that did the change.	R7.8.0

Рисунок 3.14 – Позитивний тест кейс для деактивації користувача у OIM

0059 Oracle Identity Manager Integration (finished)	UC_205106329_7	TC_212867075_71	Activate user **if User account is already activated** Note: operation <*SetUserStatus*>	Precondition: 1) Operation <*SetUserStatus*> is registered in ESB; 2) External users exist in the MaxBill database; 3) The User with the same User Code is exist; 4) The field <*UserState*> **for the same User Code already contains values** <Enabled>. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	The following error message is inserted into log file: Message Code: **E98007** Message Text: **User "user from request" is already enabled**	R7.8.0

Рисунок 3.15 – Негативний тест кейс для активації користувача у OIM, якщо користувач вже активований

0059 Oracle Identity Manager Integration (finished)	UC_205106329_7	TC_212867075_72	Deactivate user **if User account is already deactivated** Note: operation <*SetUserStatus*>	Precondition: 1) Operation <*SetUserStatus*> is registered in ESB; 2) External users exist in the MaxBill database; 3) The User with the same User Code is exist; 4) The field <*UserState*> **for the same User Code already contains values** <Disabled>. Test steps: 1) Send WSDL with mandatory User Attributes as an input.	The following error message is inserted into log file: Message Code: **E98006** Message Text: **User "user from request" is already disabled**	R7.8.0

Рисунок 3.16 – Негативний тест кейс для деактивації користувача у OIM, якщо користувач вже деактивований

Умови для виконання першого тесту наступні:

– функція <SetUserStatus> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– поле <UserState> містить значення <Disabled> у базі даних.

Кроки першого тесту наступні:

– відправляємо WSDL файл на обробку з полем <UserState>, яке містить значення <Enabled> у запиті.

За допомогою першого тесту перевіряється можливість активації користувача. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– користувач активований у таблиці <securityuser> бази даних;

– поле <LastModifiedBy> містить інформацію про систему, яка змінила статус користувача.

Умови для виконання другого тесту наступні:

– функція <SetUserStatus> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– поле <UserState> містить значення <Enabled> у базі даних.

Кроки другого тесту наступні:

– відправляємо WSDL файл на обробку з полем <UserState>, яке містить значення <Disabled> у запиті.

За допомогою другого тесту перевіряється можливість деактивації користувача. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– користувач деактивований у таблиці <securityuser> бази даних;

– поле <LastModifiedBy> містить інформацію про систему, яка змінила статус користувача.

Умови для виконання третього тесту наступні:

– функція <SetUserStatus> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– поле <UserState> містить значення <Enabled> у базі даних.

Кроки третього тесту наступні:

– відправляємо WSDL файл на обробку з полем <UserState>, яке містить значення <Enabled> у запиті.

За допомогою третього тесту перевіряється можливість активації користувача, якщо він вже активований. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу містить помилку «Користувач вже активований».

Умови для виконання четвертого тесту наступні:

– функція <SetUserStatus> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– поле <UserState> містить значення <Disabled> у базі даних.

Кроки четвертого тесту наступні:

– відправляємо WSDL файл на обробку з полем <UserState>, яке містить значення <Disabled> у запиті.

За допомогою четвертого тесту перевіряється можливість деактивації користувача, якщо він вже деактивований. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу містить помилку «Користувач вже деактивований».

3.1.5 Написання тестів для отримання повної інформації про створеного користувача, про користувача зі зміненими параметрами, про групу користувача

Для того, щоб отримати повну інформацію про створеного користувача, було написано декілька позитивних тест кейсів, один з яких представлений на рисунку 3.17.

Project	Use Case	Test Case ID	Test Case Name	Test Case Description	Expected Result	Version
0059 Oracle Identity Manager Integration (finished)	UC_205106329_1	TC_212867075_01	Get **all user accounts** Note: operation *<ListAllUsersDetailed>*	Precondition: 1) External users exist in the MaxBill database; 2) Operation *<ListAllUsersDetailed>* is registered in ESB. Test steps: 1) Send WSDL with **no** input parameters.	1) The interface returns list of all active and not active accounts of external type available in the **securityuser** table.	R7.8.0

Рисунок 3.17 – Позитивний тест кейс для перевірки виводу повної інформації про користувачів у OIM

Умови для виконання першого тесту наступні:

– функція <ListAllUsersDetailed> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних.

Кроки першого тесту наступні:

– відправляємо WSDL файл на обробку без вхідних параметрів.

За допомогою першого тесту перевіряється отримання повної інформації про користувачів у OIM. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– у відповіді WSDL файлу міститься інформація про всіх користувачів з таблиці <securityuser>.

Для того, щоб отримати повну інформацію про користувача зі зміненими параметрами, було написано декілька позитивних тест кейсів, один з яких представлений на рисунку 3.18.

0059 Oracle Identity Manager Integration (finished)	UC_205106329_2	TC_212867075_15	Get **all user accounts** that have been **changed** since a certain time Note: operation *<ListAllModifiedUsers>*	Precondition: 1) External users exist in the MaxBill database; 2) Operation *<ListAllModifiedUsers>* is registered in ESB. Test steps: 1) Send WSDL with the date since the changed accounts are to be returned.	1) The interface returns list of all active and not active accounts of external type available in the **securityuser** table that were modified since specified input date.	R7.8.0

Рисунок 3.18 – Позитивний тест кейс для перевірки виводу повної інформації про користувачів зі зміненими параметрами у OIM

Умови для виконання другого тесту наступні:

– функція <ListAllModifiedUsers> зареєстрована у системі ESB;

– користувач з таким <UserID> та зі зміненими параметрами присутній у базі даних.

Кроки другого тесту наступні:

– відправляємо WSDL файл на обробку з датою, з якої необхідно повернути користувачів зі зміненими параметрами.

За допомогою другого тесту перевіряється отримання повної інформації про користувачів зі зміненими параметрами у OIM. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– у відповіді WSDL файлу міститься інформація про всіх користувачів з таблиці <securityuser>, параметри яких було змінено з дати, вказаної у запиті WSDL файлу.

Для того, щоб отримати повну інформацію про групу користувача, було написано декілька позитивних тест кейсів, один з яких представлений на рисунку 3.19.

0059 Oracle Identity Manager Integration (finished)	UC_205106329_3	TC_212867075_30	Get **all authorization elements** Note: operation *<ListGroups>*	Precondition: 1) External users exist in the MaxBill database; 2) Operation *<ListGroups>* is registered in ESB; 3) Groups configured in the MaxBill DB. Test steps: 1) Send WSDL with **no** input parameters.	1) The interface returns list of all security groups available in the **securityuser** table that are not deleted.	R7.8.0

Рисунок 3.19 – Позитивний тест кейс для отримання повної інформації про групу користувача у OIM

Умови для виконання третього тесту наступні:

– функція <ListGroups> зареєстрована у системі ESB;

– користувач з таким <UserID> присутній у базі даних;

– група з таким <UserGroups> присутня у базі даних.

Кроки третього тесту наступні:

– відправляємо WSDL файл на обробку без вхідних параметрів.

За допомогою третього тесту перевіряється отримання повної інформації про групу користувача у OIM. Очікуваний результат даного тесту наступний:

– відповідь WSDL файлу не містить помилки;

– у відповіді WSDL файлу міститься інформація про всі групи з таблиці <securitygroup>.

3.2 Покриття тестів (кейсів) автоматизованими тестами

3.2.1 Написання авто-тестів для перевірки створення користувача

Для того, щоб перевірити можливість створення користувача, якщо користувач з таким <UserID> відсутній у базі даних були створені змінні за допомогою groovy script, які відправляються у запиті wsdl файлу. Після отримання відповіді зроблена перевірка, чи дійсно такий користувач присутній у базі даних. У результаті проходження тесту було виявлено, що існує можливість створити користувача з <UserID>, який відсутній у базі даних. Результат позитивного авто-тесту представлений на рисунках 3.20 -3.22.

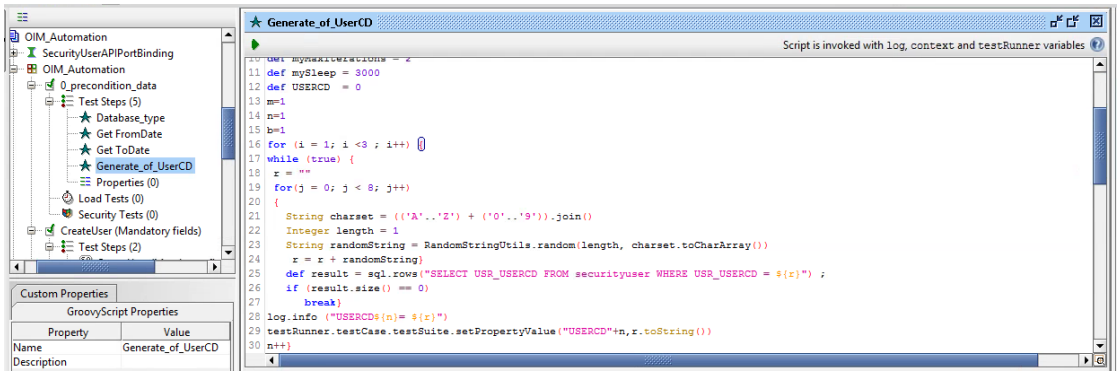


Рисунок 3.20 – Створення змінної <UserID> у groovy script

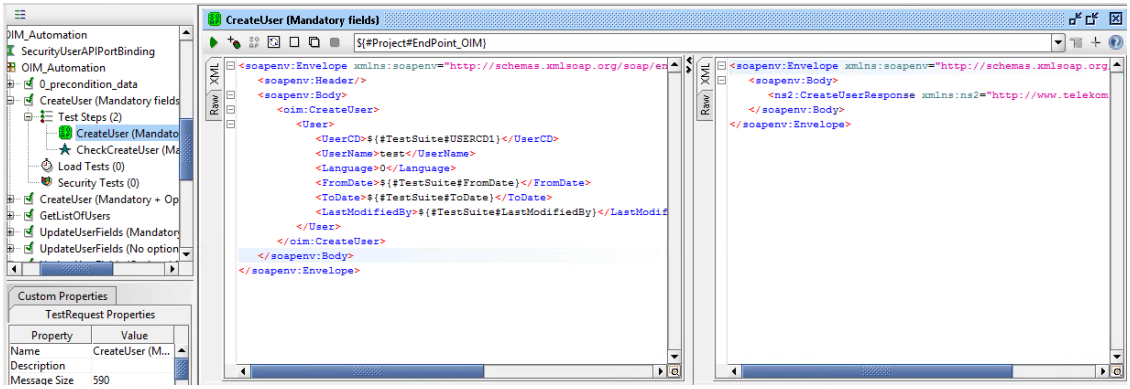


Рисунок 3.21 – Результат проходження першого тесту для перевірки створення користувача у OIM

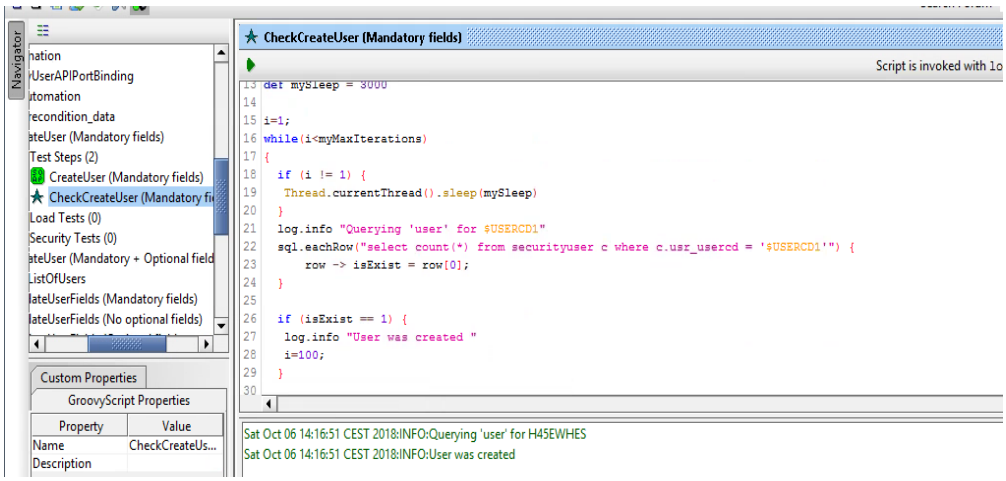


Рисунок 3.22 – Перевірка, що користувач дійсно існує у базі даних у groovy script

Для перевірки негативного тест кейсу, якщо користувач з таким <UserID> присутній у базі даних запит був відправлений ще раз. У результаті проходження тесту було виявлено, що відсутня можливість створити користувача з <UserID>, який вже присутній у базі даних. Результат авто- тесту представлений на рисунку 3.23.

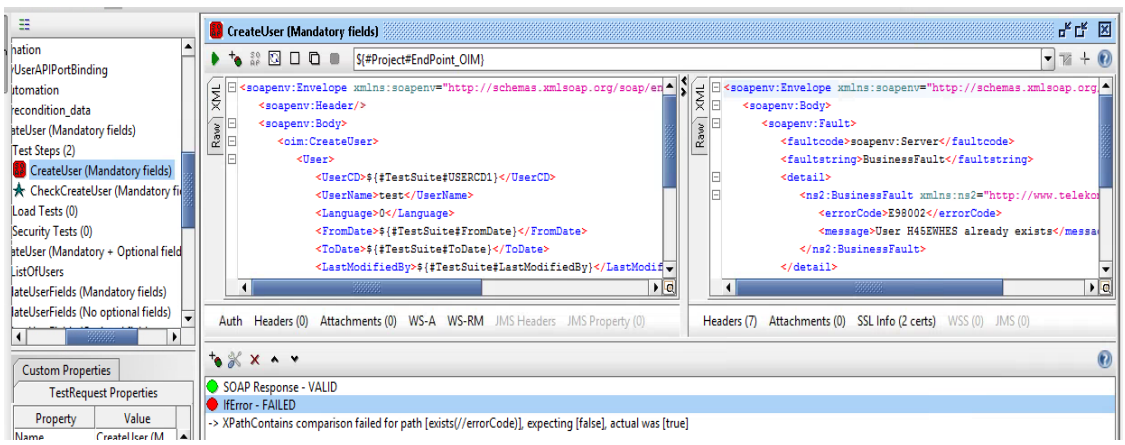


Рисунок 3.23 – Результат проходження другого тесту для перевірки створення користувача у OIM

3.2.2 Написання авто-тестів для перевірки можливості зміни параметрів користувача

Для того, щоб перевірити можливість зміни параметрів для створеного користувача у запиті wsdl файлу були змінені поля <UserName> та <Language>. Після отримання відповіді зроблена перевірка, чи дійсно параметри користувача були змінені у базі даних за допомогою groovy script. У результаті проходження тесту було виявлено, що існує можливість змінити параметри з <UserID>, який присутній у базі даних. Результат позитивного авто-тесту представлений на рисунках 3.24 - 3.25.

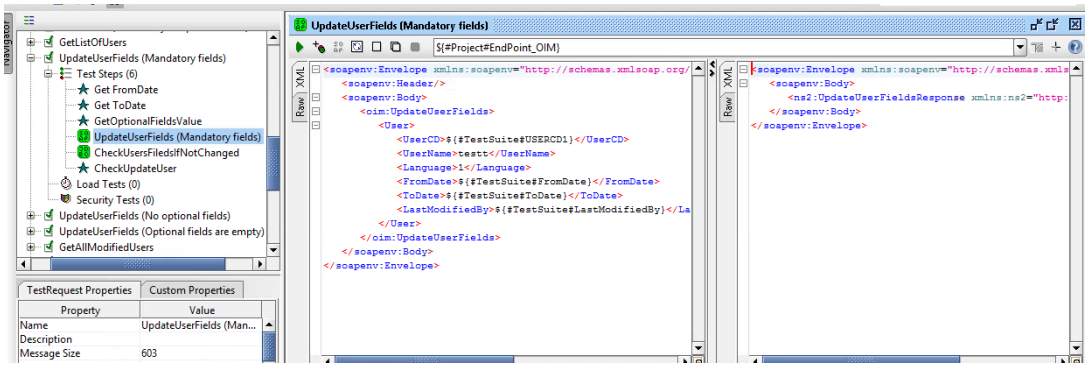


Рисунок 3.24 – Результат проходження першого тесту для перевірки зміни параметрів користувача у OIM

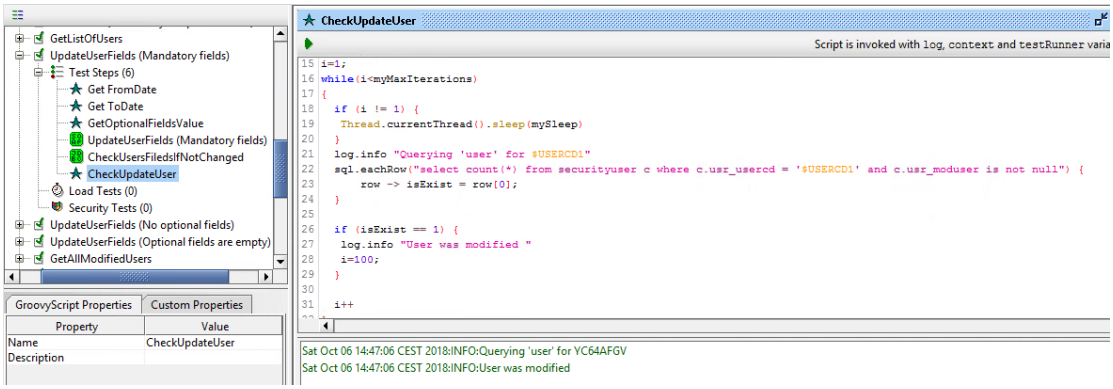


Рисунок 3.25 – Перевірка, що параметри користувача були змінені groovy script

Для перевірки негативного тест кейсу, якщо користувач з таким <UserID> відсутній у базі даних запит був відправлений ще раз з неіснуючим <UserID>. У результаті проходження тесту було виявлено, що відсутня можливість змінити параметри користувача з <UserID>, який відсутній у базі даних. Результат авто-тесту представлений на рисунку 3.26.

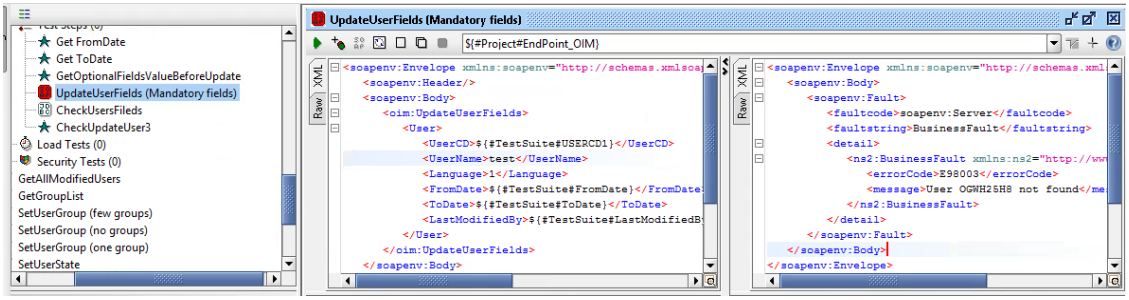


Рисунок 3.26 – Результат проходження другого тесту для перевірки зміни параметрів користувача у OIM

Для перевірки негативного тест кейсу, якщо обов'язкові поля запиту не були змінені запит був відправлений ще раз з незміненими полями <UserName> та <Language>. Після отримання відповіді зроблена перевірка, чи дійсно параметри користувача не були змінені у базі даних за допомогою jdbc request. У результаті проходження тесту було виявлено, що відповідь не повертає помилки та усі поля містять не оновлені значення. Результат авто- тесту представлений на рисунку 3.27.

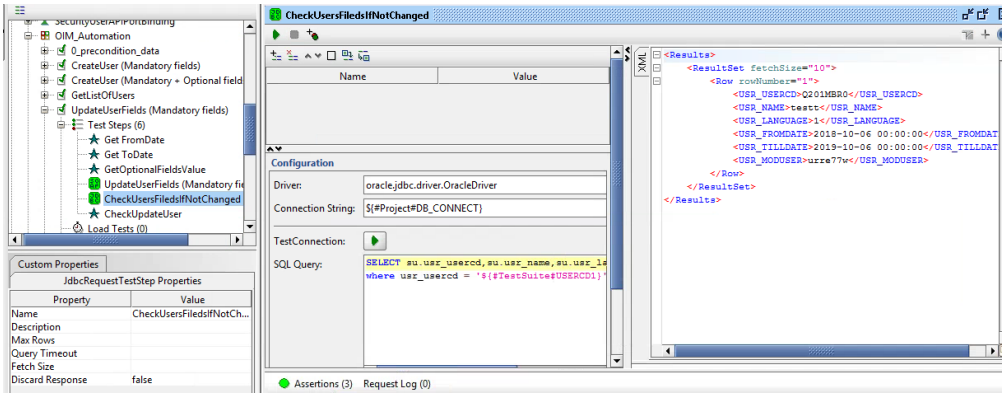


Рисунок 3.27 – Результат проходження третього тесту для перевірки зміни параметрів користувача у OIM

Для перевірки негативного тест кейсу, якщо не обов'язкові поля запиту <Description> та <Email> відсутні у запиті wsdl файлу було змінено поле <Language>. Після отримання відповіді зроблена перевірка, чи дійсно параметри користувача не були змінені у базі даних за допомогою jdbc request. У результаті проходження тесту було виявлено, що відповідь не повертає помилки та не обов'язкове поле містить не оновлене значення. Результат авто-тесту представлений на рисунку 3.28.

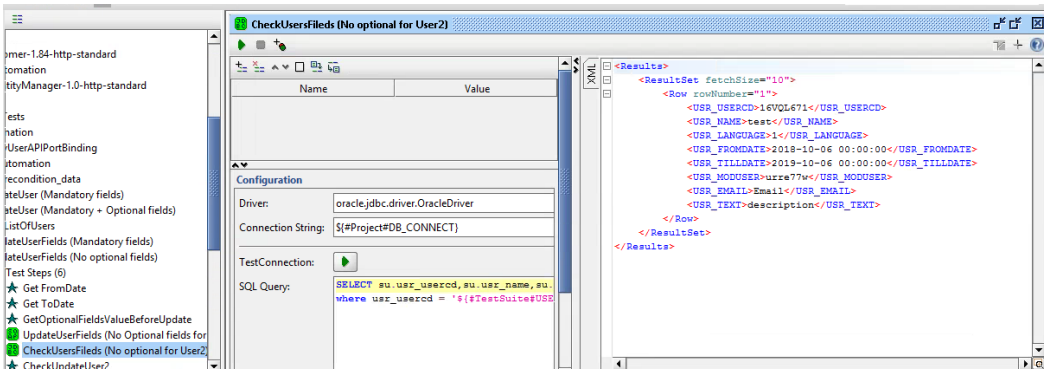


Рисунок 3.28 – Результат проходження четвертого тесту для перевірки зміни параметрів користувача у OIM

3.2.3 Написання авто-тестів для перевірки можливості додавання списку груп користувачу

Для того, щоб перевірити можливість додавання списку груп існуючому користувачу були створені змінні зі значеннями груп на рівні тест кейсу. Після отримання відповіді зроблена перевірка, чи дійсно користувач має такі групи за допомогою jdbc request. У результаті проходження тесту було виявлено, що існує можливість додавати декілька груп користувачу. Результат позитивного авто-тесту представлений на рисунках 3.29 - 3.30.

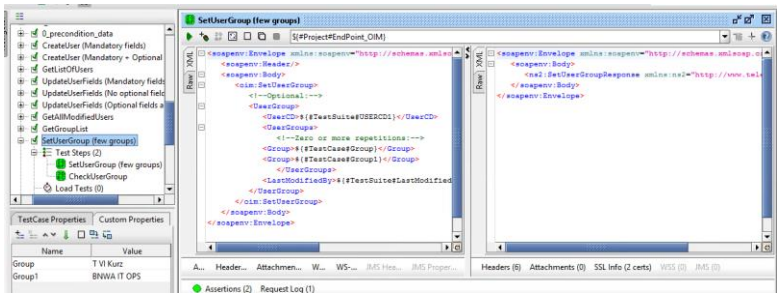


Рисунок 3.29 – Результат проходження першого тесту для перевірки додавання списку груп користувачу у OIM

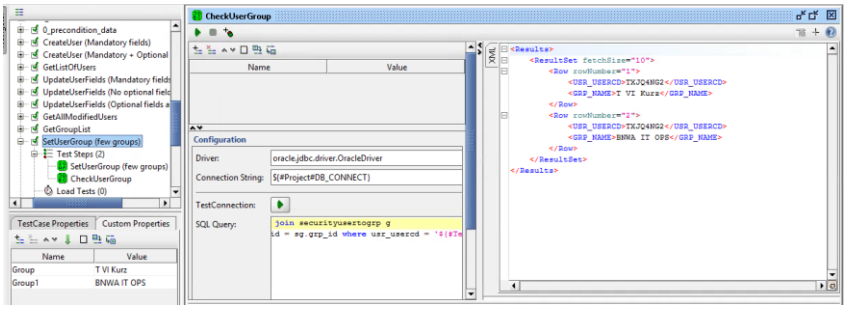


Рисунок 3.30 – Перевірка груп користувача у OIM

Для того, щоб перевірити можливість додавання списку груп не існуючому користувачу був відправлений запит ще раз з неіснуючим <UserID>. У результаті проходження тесту було виявлено, що не має можливості додавати декілька груп не існуючому користувачу. Результат позитивного авто-тесту представлений на рисунку 3.31.

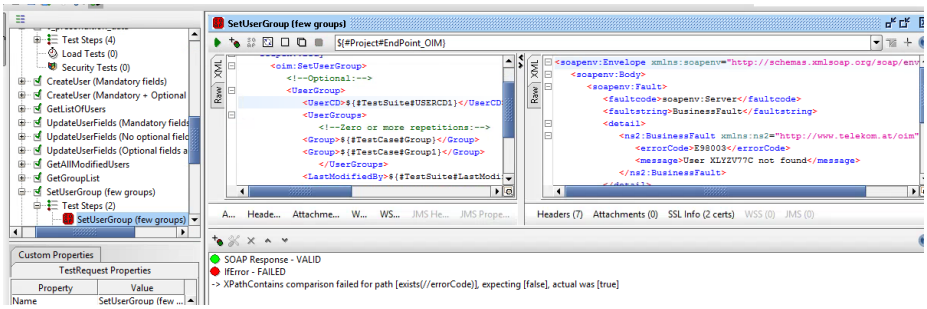


Рисунок 3.31 – Результат проходження другого тесту для перевірки додавання списку груп не існуючому користувачу у OIM

Для того, щоб перевірити можливість додавання не існуючого списку груп існуючому користувачу були створені змінні зі значеннями груп на рівні тест кейсу. У результаті проходження тесту було виявлено, що не має можливості додавати декілька не існуючих груп користувачу. Результат негативного авто-тесту представлений на рисунку 3.32.

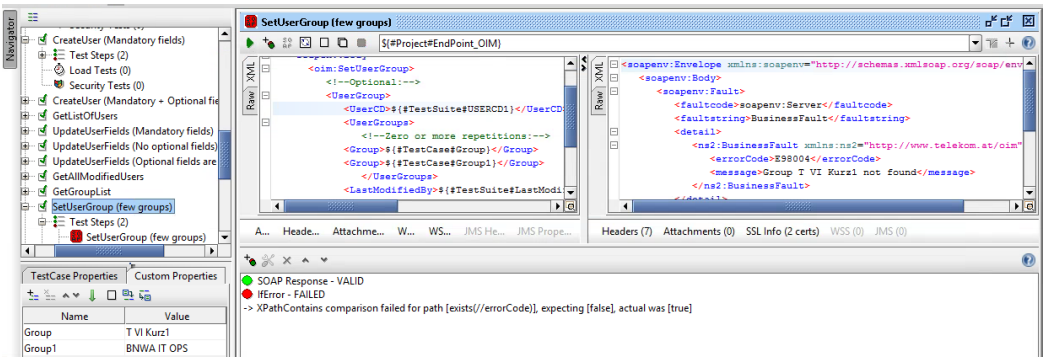


Рисунок 3.32 – Результат проходження третього тесту для перевірки додавання неіснуючого списку груп існуючому користувачу у OIM

Для перевірки четвертого тест кейсу, якщо список груп порожній у запиті wsdl файлу були видалені групи. Після отримання відповіді зроблена перевірка, які групи має користувач за допомогою jdbc request.У результаті проходження тесту було виявлено, що попередні групи користувача видаляються з бази даних. Результат негативного авто-тесту представлений на рисунках 3.33 - 3.34.

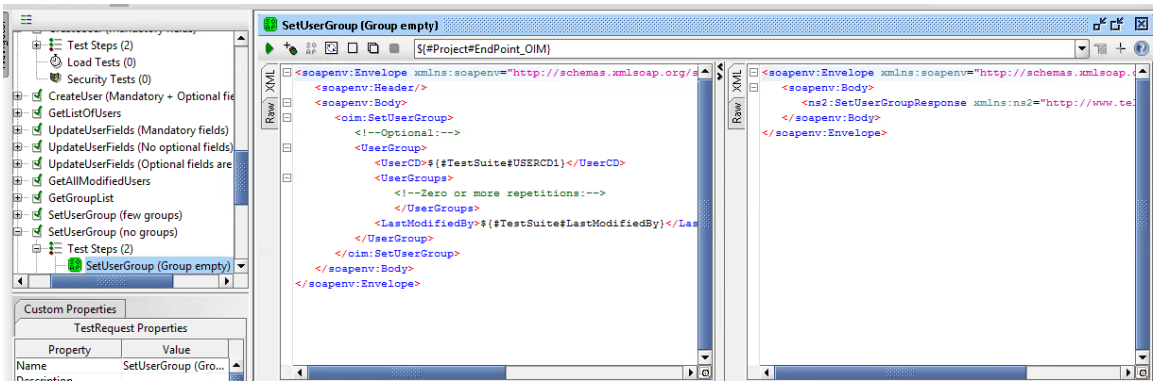


Рисунок 3.33 – Результат проходження четвертого тесту для перевірки додавання порожнього списку груп існуючому користувачу у OIM

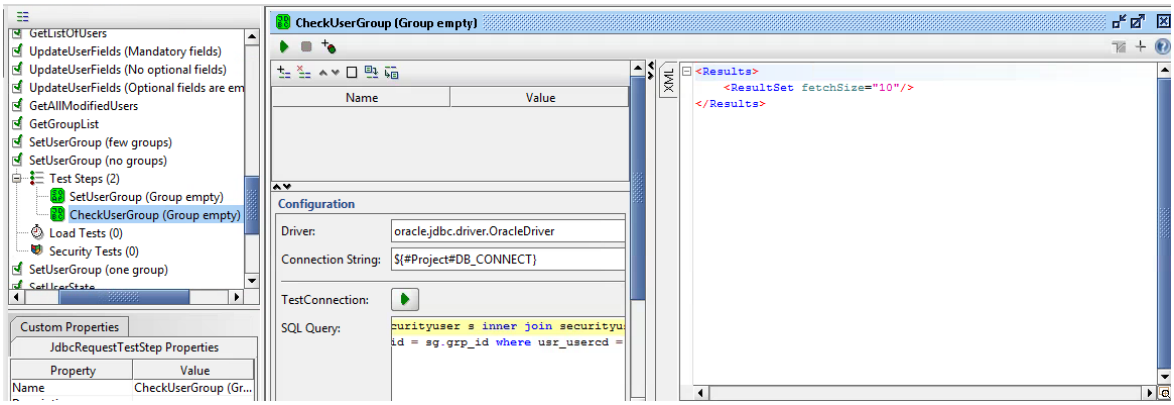


Рисунок 3.34 – Перевірка, що групи користувача видалені у OIM

Для перевірки п’ятого тест кейсу, якщо список груп містить дві однакові групи у запиті wsdl файлу була додана ще одна існуюча група. У результаті проходження тесту було виявлено, що не має можливості додавати дві однакові групи користувачу в один момент часу. Результат негативного авто-тесту представлений на рисунку 3.35.

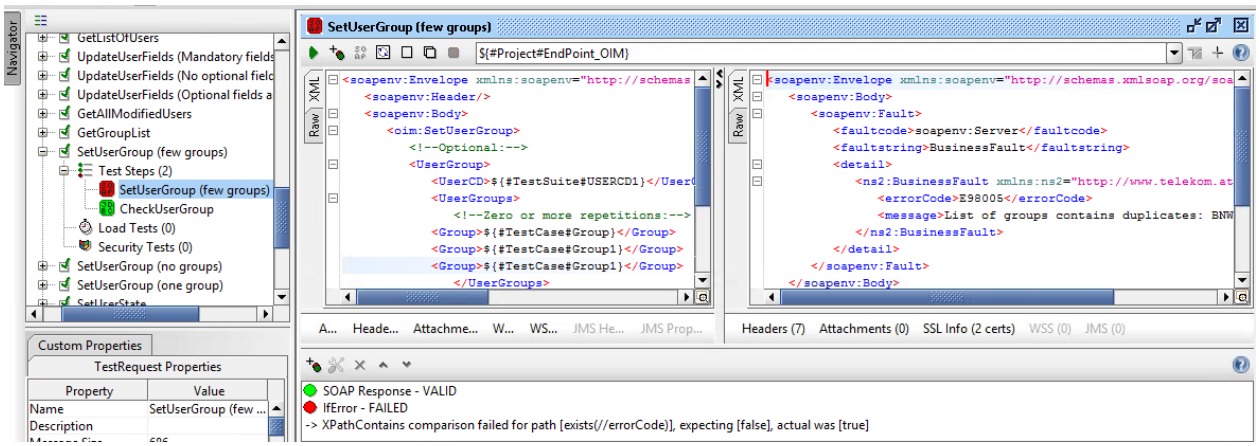


Рисунок 3.35 – Результат проходження п’ятого тесту для перевірки додавання однакових груп існуючому користувачу у OIM

Для того, щоб перевірити можливість додавання однієї групи існуючому користувачу у запиті wsdl файлу було видалено список, окрім однієї групи. Після отримання відповіді зроблена перевірка, чи дійсно користувач має таку групу за допомогою jdbc request. У результаті проходження тесту було виявлено, що існує можливість додавати одну групу користувачу. Результат позитивного авто-тесту представлений на рисунках 3.36 - 3.37.

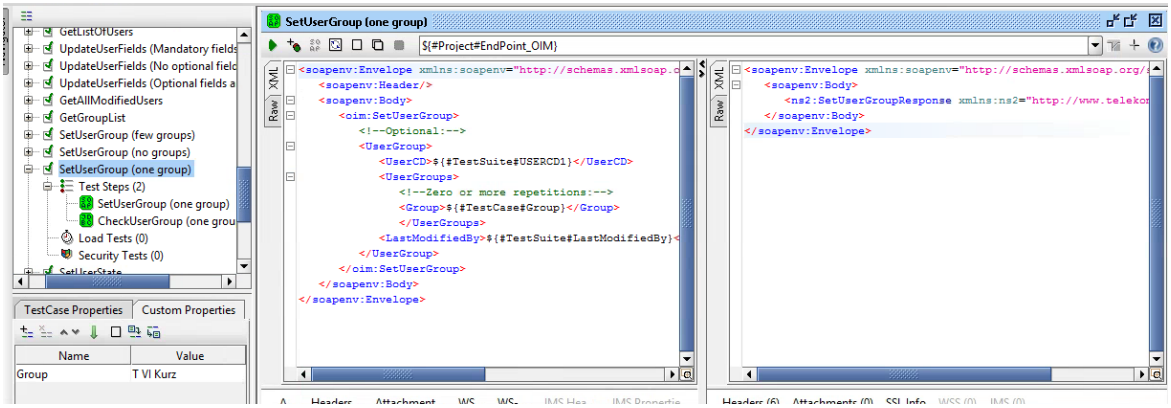


Рисунок 3.36 – Результат проходження шостого тесту для перевірки додавання однієї групи існуючому користувачу у OIM

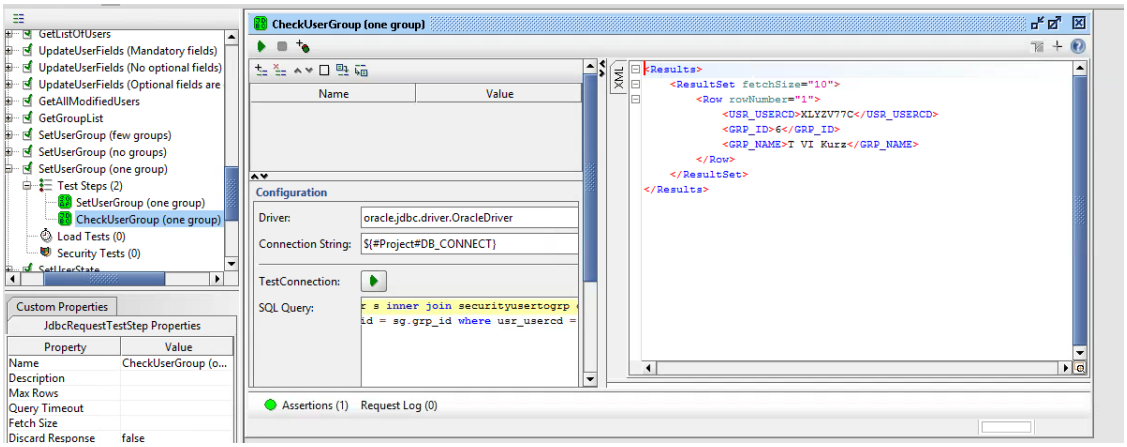


Рисунок 3.37 – Перевірка груп користувача у OIM

3.2.4 Написання авто-тестів для перевірки можливості зміни статусу користувача

Для того, щоб перевірити можливість зміни статусу користувача у запиті wsdl файлу поле <UserState> містить значення Disabled. Після отримання відповіді зроблена перевірка, чи дійсно користувач деактивований за допомогою jdbc request. У результаті

проходження тесту було виявлено, що існує можливість деактивації користувача. Результат позитивного авто- тесту представлений на рисунках 3.38 - 3.39.

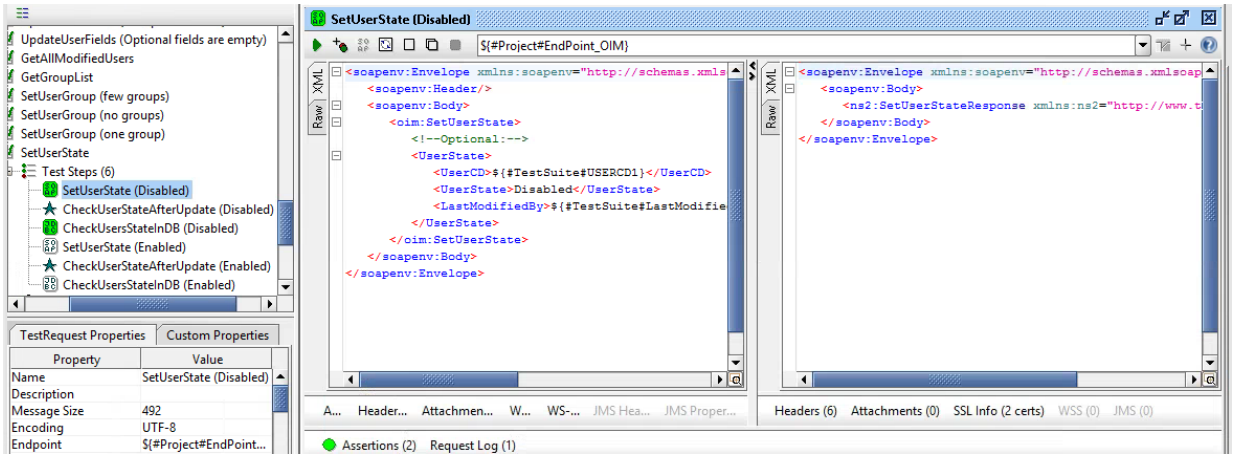


Рисунок 3.38 – Результат проходження першого тесту для перевірки статусу користувача у OIM

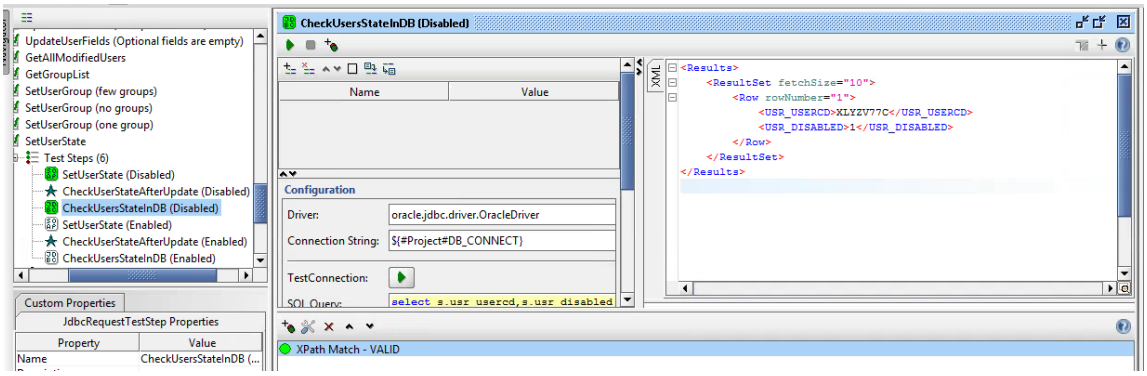


Рисунок 3.39 – Перевірка, що користувач деактивований

Для того, щоб перевірити можливість активації користувача у запиті wsdl файлу поле <UserState> містить значення Enabled. Після отримання відповіді зроблена перевірка, чи дійсно користувач активований за допомогою jdbc request. У результаті проходження тесту було виявлено, що існує можливість активації користувача. Результат позитивного авто-тесту представлений на рисунку 3.40.

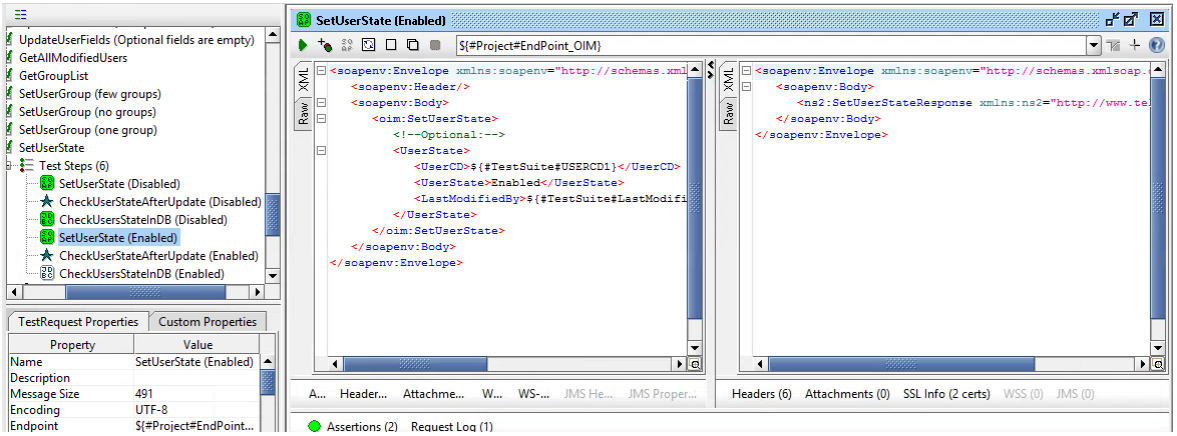


Рисунок 3.40 – Результат проходження другого тесту для перевірки активації користувача у OIM

Для перевірки деактивації користувача, якщо він вже деактивований запит зі значенням Disabled у полі <UserState> був відправлений ще раз. У результаті проходження тесту було виявлено, що не існує можливості деактивації користувача, якщо він вже має такий статус. Результат негативного авто-тесту представлений на рисунку 3.41.

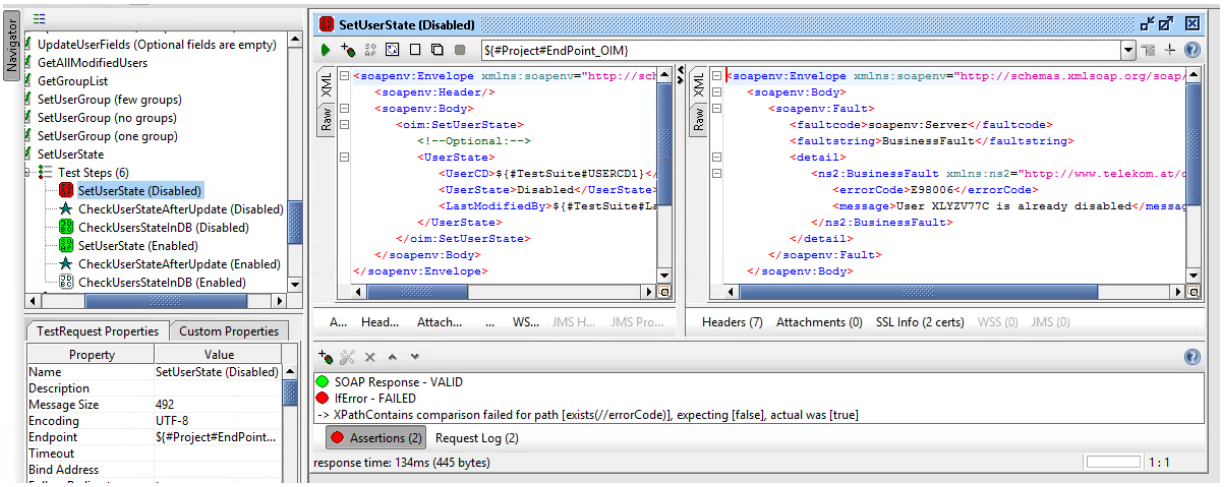


Рисунок 3.41 – Результат проходження третього тесту для перевірки деактивації користувача у OIM

Відповідно, щоб перевірити активацію користувача, якщо він вже активований запит зі значенням Enabled у полі <UserState> був відправлений ще раз. У результаті проходження тесту було виявлено, що не існує можливості активації користувача, якщо він вже має такий статус. Результат негативного авто-тесту представлений на рисунку 3.42.

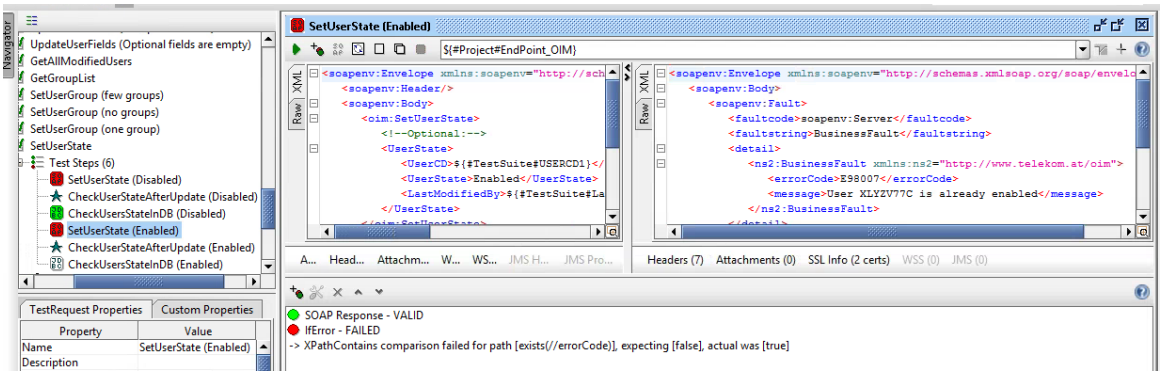


Рисунок 3.42 – Результат проходження четвертого тесту для перевірки активації користувача у OIM

3.2.5 Написання авто-тестів для отримання повної інформації про створеного користувача, про користувача зі зміненими параметрами, про групу користувача

Для перевірки першого тест кейсу було написано декілька groovy script, для отримання інформації з бази даних. У першому скрипті перевіряється кількість користувачів, які отримано у відповіді, у другому скрипті перевіряється відповідність інформації користувача, яку отримана у відповіді та яка існує у базі даних. Значення полів записані на рівні тест кейсу після виконання groovy script. Також було додано декілька перевірок на рівні soap request, у яких порівнюється інформація отримана з відповіді wsdl файлу та groovy script. У результаті проходження тесту було виявлено, що отримана інформація у відповіді wsdl файлу співпадає з інформацією з бази даних. Результат авто-тесту представлений на рисунках 3.43 - 3.45.

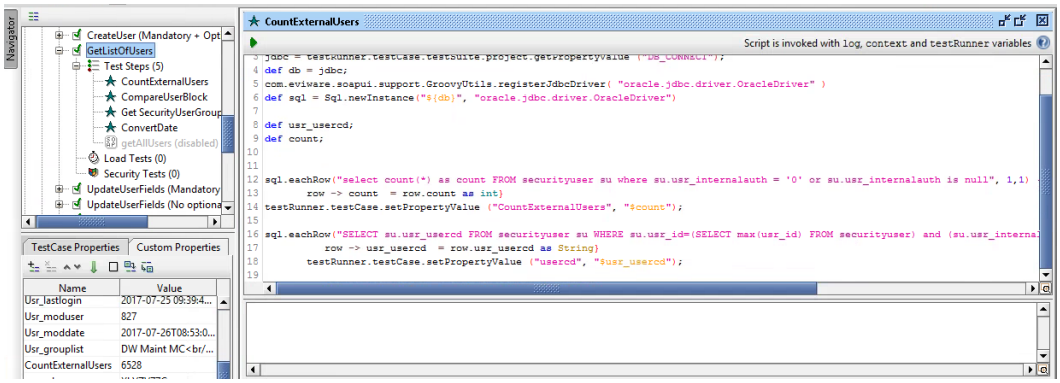


Рисунок 3.43 – Groovy script для отриманні інформації про кількість користувачів у базі даних

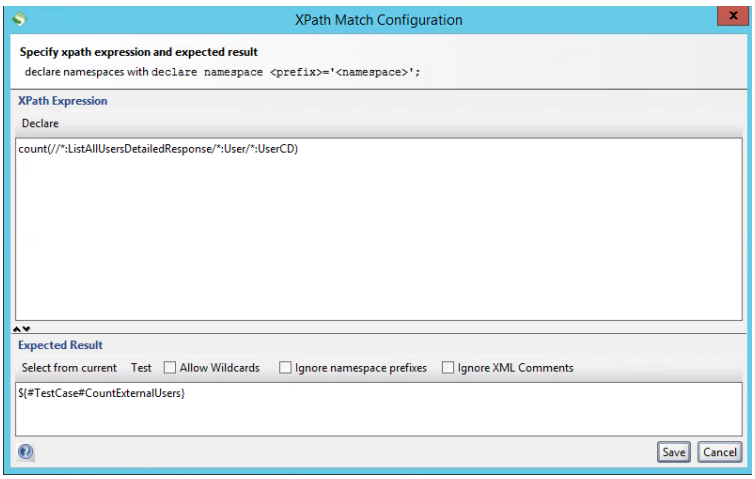


Рисунок 3.44 – XPAth для перевірки відповідності кількості користувачів у відповіді та groovy script

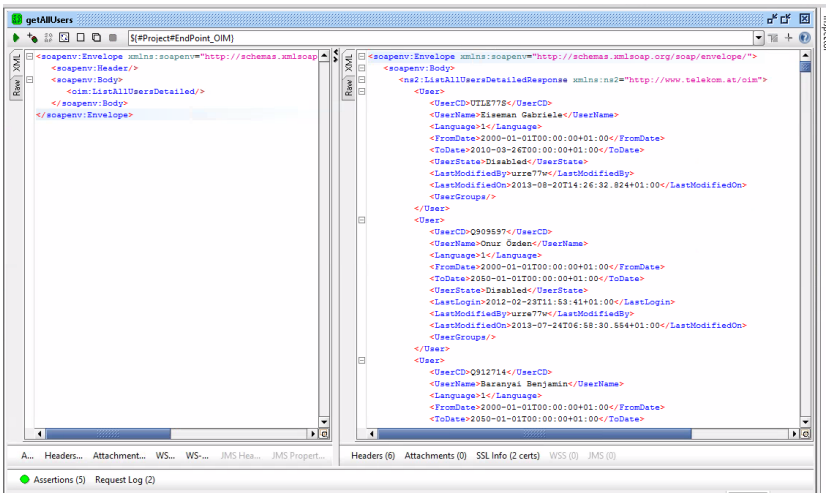


Рисунок 3.45 – Результат проходження тест кейсу у SoapUI для отримання повної інформації про користувача

Для перевірки другого тест кейсу було написано декілька groovy script, для отримання інформації з бази даних. У першому скрипті перевіряється кількість користувачів, параметри яких було змінено з відповідної дати у запиті wsdl файлу, у другому скрипті перевіряється відповідність інформації користувача, яку отримана у відповіді та яка існує у базі даних. Змінну зі значенням дати було додано на рівні тест кейсу. Значення полів також записані на рівні тест кейсу після виконання groovy script. Також було додано декілька перевірок на рівні soap request, у яких порівнюється інформація отримана з відповіді wsdl файлу та groovy script. У результаті проходження тесту було виявлено, що у відповіді міститься повна інформація про усіх користувачів, параметри яких було змінено з відповідної дати. Результат авто-тесту представлений на рисунках 3.46 - 3.48.

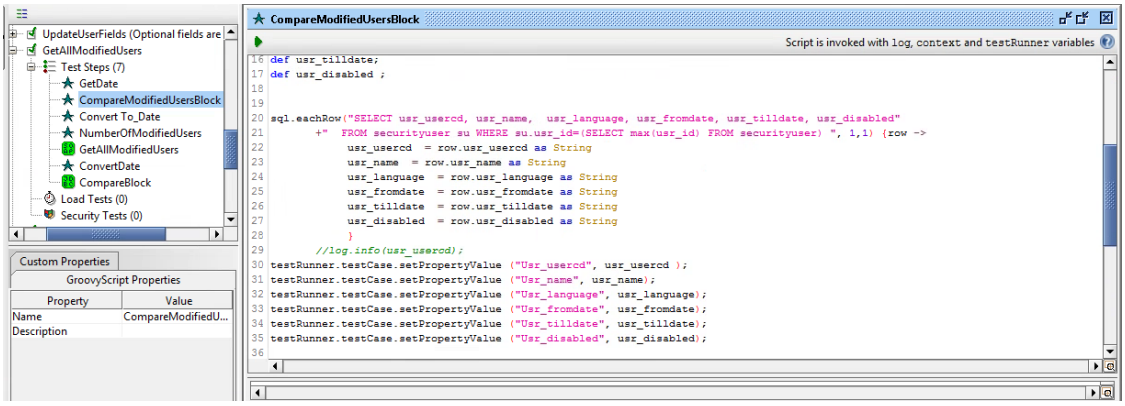


Рисунок 3.46 – Groovy script для отриманні інформації про користувача

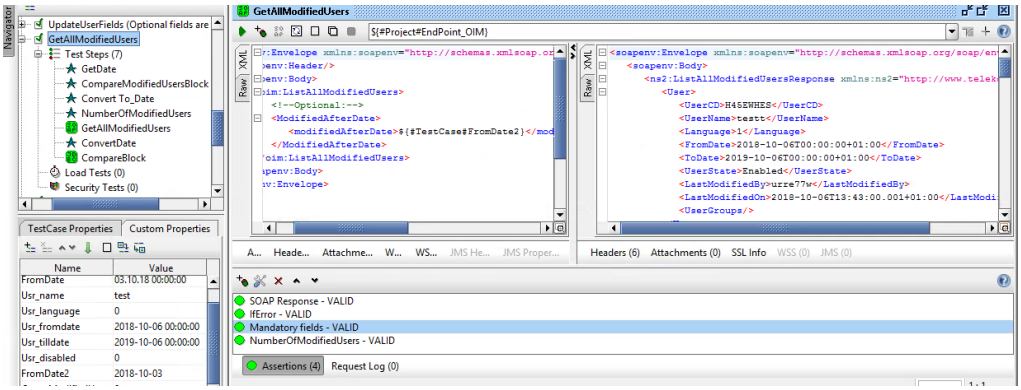


Рисунок 3.47 – Результат проходження другого тесту для отримання інформації про користувачів, параметри яких було змінено з відповідної дати

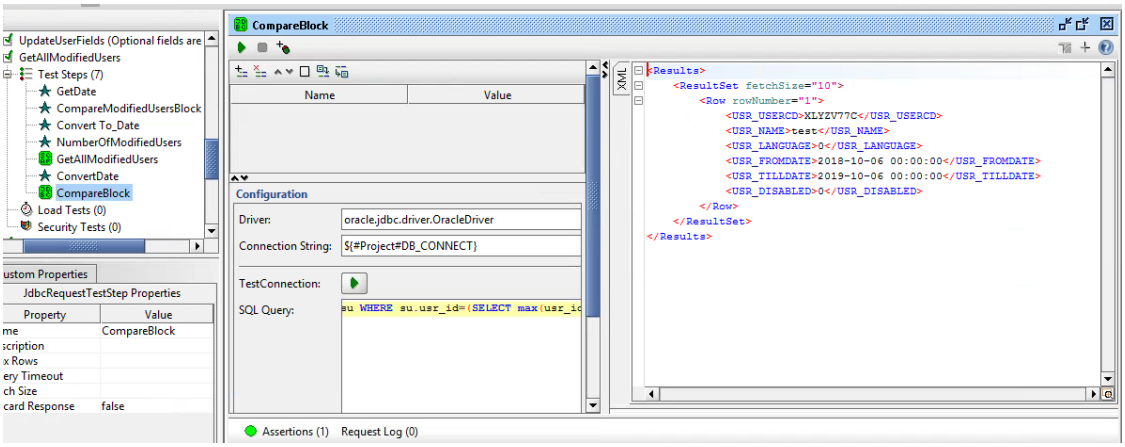


Рисунок 3.48 – JDBC request для перевірки відповідності інформації користувачів у відповіді та groovy script

Для перевірки третього тест кейсу був написаний groovy script для отримання інформації про кількість груп з бази даних. Також було додано декілька перевірок на рівні soap request, у яких порівнюється інформація отримана з відповіді wsdl файлу та groovy script. У результаті проходження тесту було виявлено, що у відповіді міститься повна інформація про усі існуючи групи. Результат авто-тесту представлений на рисунку 3.49.

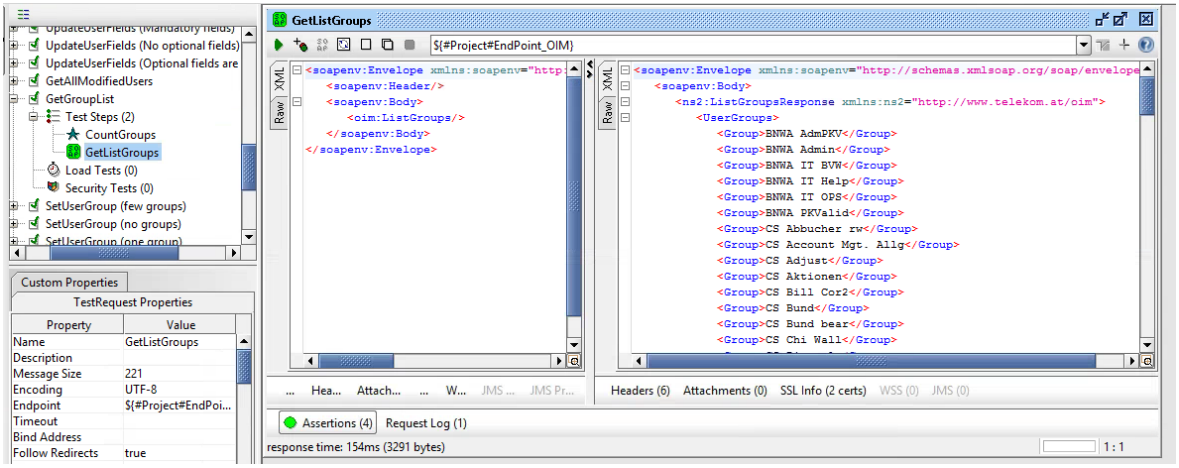


Рисунок 3.49 – Результат проходження третього тесту для отримання інформації про існуючи групи

3.3 Дослідження автоматизації тестування

Для того, щоб зробити дослідження автоматизації тестування веб-сервісу, було прийнято рішення порівняти між собою такі функції автоматизації як Groovy Script та JDBC Request разом з Conditional Goto. Слід відповісти на запитання: Ефективніше використати декілька автоматизованих тестових кроків у одному тесті для перевірки коректного додавання даних чи один тестовий крок для перевірки тієї ж функції? Для відповіді на це запитання розглянемо приклад використання цих функцій у тест кейсі SetUserGroup, результат роботи яких представлений на рисунку 3.55.

У першому тест кейсі користувачу додається одна група. Щоб перевірити, що користувач має таку групу після отримання відповіді у wsdl файлі додаємо тестовий крок delay – пауза, для того, щоб запит встиг виконатися у базі даних та jdbc request, за допомогою якого отримаємо групу з бази даних. Далі необхідно додати умову conditional goto, яка перевіряє у xml структурі попереднього кроку jdbc request значення поля grp_name. Якщо поле містить значення, відправлене у запиті wsdl файлу, переходимо до останнього кроку jdbc request, де перевіряється успішний або невдалий результат проходження усього тест кейсу, якщо поле містить інше значення, ми повертаємося до першого кроку у тест кейсі та виконуємо запит на додавання групи користувачу ще раз. Таким чином, у результаті перевірки групи ми отримаємо чотири тестових кроки автоматизації представлених на рисунках 3.50 - 3.53.

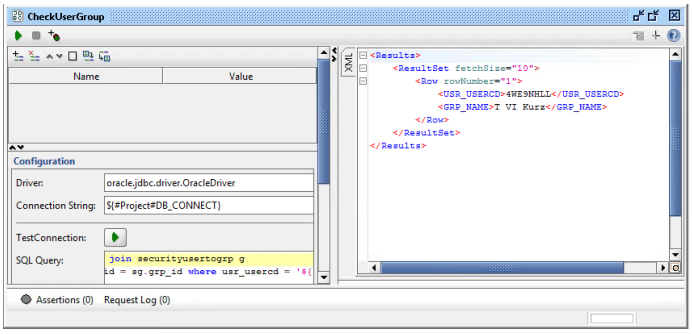


Рисунок 3.50 – JDBC Request для отримання групи з бази даних

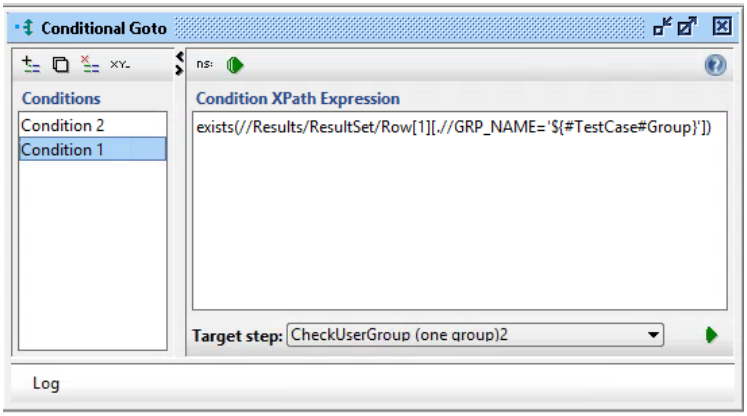


Рисунок 3.51 – Conditional Goto – перша умова для перевірки групи

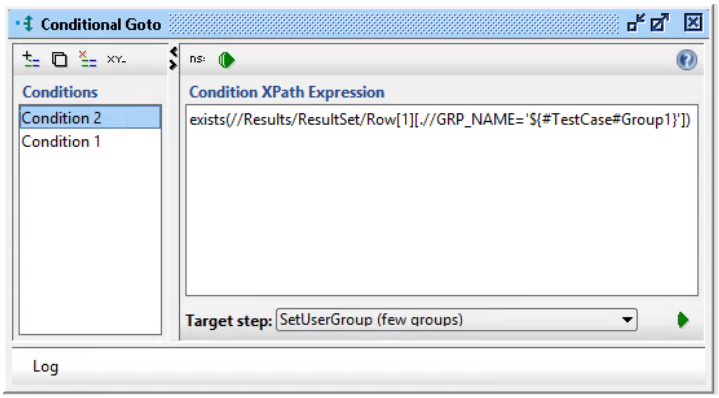


Рисунок 3.52 – Conditional Goto – друга умова для перевірки групи

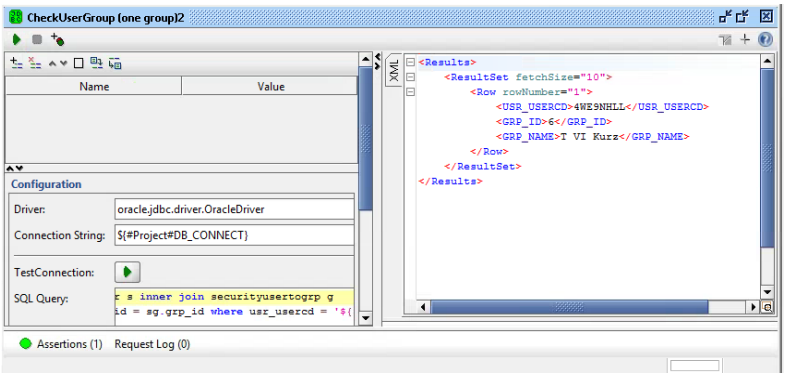


Рисунок 3.53 – JDBC Request – перевірка групи у виході з циклу для отримання кінцевого результату тест кейсу

У другому тест кейсі користувачу також додається одна група. Щоб перевірити, що користувач має таку групу після отримання відповіді у wsdl файлі додаємо тестовий крок groovy script, у якому:

– перевіряємо, що користувач має групу;

– перевіряємо, що користувач має саме таку групу, яка була відправлена у запиті wsdl файлу;

– додаємо паузу на початку циклу, щоб запит встиг обробитися у базі даних.

У разі не відповідності однієї з цих умов groovy script виходить з циклу та тестовий крок завершується невдало, як і весь тест кейс. Перевірка за допомогою groovy script представлена на рисунку 3.54.

```
import groovy.sql.Sql
jdbc = testRunner.testCase.testSuite.project.getPropertyValue ("DB_CONNECT");
def db = jdbc;
com.eviware.soapui.support.GroovyUtils.registerJdbcDriver( "oracle.jdbc.driver.OracleDriver" )
def sql = Sql.newInstance("${db}", "oracle.jdbc.driver.OracleDriver")
USERCD1 =  testRunner.testCase.testSuite.getPropertyValue ("USERCD1");
Group = testRunner.testCase.getPropertyValue ("Group");
def isExist = null
def myMaxIterations = 5
def mySleep = 3000
i=1;
while(i<myMaxIterations)
{if (i != 1) {
   Thread.currentThread().sleep(mySleep)}
  log.info "Querying 'user' for $USERCD1"
  sql.eachRow("select count(*) from securityusertogrp g where g.usr_id in (SELECT usr_id FROM securityuser where usr_usercd = '$USERCD1')") {
      row -> isExist = row[0]; }
 if (isExist == 1 ) {
   log.info "Group exists"
testRunner.testCase.setPropertyValue ("UserGroup", "$Group");
   i=100; }
 if (isExist == 0) {
   log.info "Group does not exist"
testRunner.testCase.setPropertyValue ("UserGroup", "0");
   i=100; }
  i++ }
if (i==myMaxIterations){
 log.info "ERROR: Reached MAX iterations $i "
 assert false}
Groupp = testRunner.testCase.getPropertyValue ("UserGroup")
log.info(Groupp)
if (Groupp == "$Group"){
log.info "Group is added correctly"}
else {
log.info "Group is added NOT correctly"
assert false    }
```

Рисунок 3.54 – Код тестового кроку groovy script

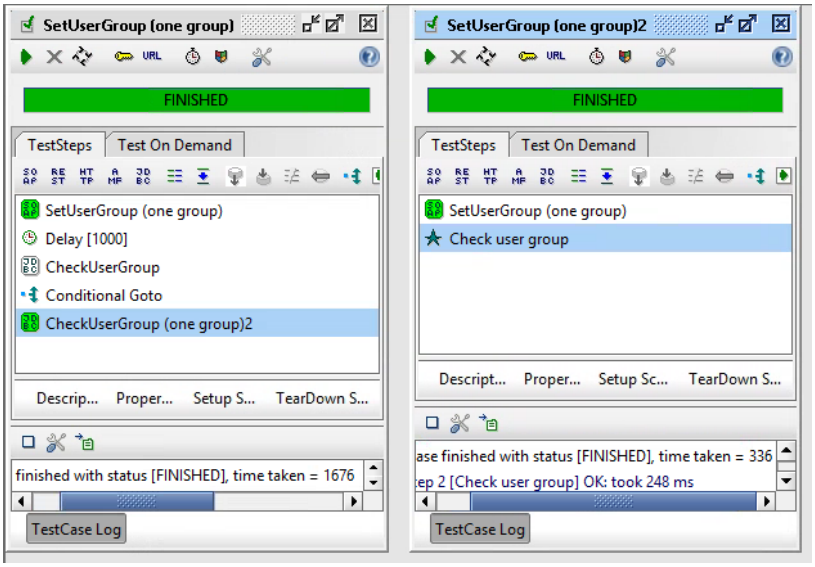


Рисунок 3.55 – Результат роботи двох тестів, виконаних за допомогою різних функцій

3.4 Аналіз результатів автоматизації у SoapUI

На основі отриманих результатів дослідження автоматизації тестування веб-сервісу, зробимо їх аналіз. Як показано на рисунку 3.55 час виконання тесту з використанням groovy script значно менший, ніж час виконання тесту з conditional goto та jdbc request. Це означає, що декілька тестових кроків, призначені для автоматизації функціонального тестування можна замінити одним скриптом для перевірки того ж функціоналу веб-сервісу, щоб покращити час виконання автоматизованих тестів та зробити їх набагато ефективнішими.

Також, автоматизацію у SoapUI для тестування головного функціоналу інтерфейсу OIM було доцільно використовувати у таких випадках, як:

- генерація тестових даних;
- регресійне тестування;
- навантажувальне тестування;
- перевірка валідаційних повідомлень;
- перевірка правильності пошуку даних;
- димове тестування;
- перевірка структури відповіді wsdl файлу.

Усі написані автоматизовані тести значно вплинули на час тестування інтерфейсу OIM. У процесі тестування було знайдено ряд дефектів, для перевірки роботи яких у регресійному тестуванні запускалися авто-тести, які дозволили швидко зробити аналіз у разі невдалого проходження тесту. Авто- тести також було запущено у процесі димового тестування, щоб швидко перевірити роботу незміненого функціоналу інтерфейсу та зосередитися на нових функціях. Автоматизація також була використана у навантажувальному тестуванні, де один тест було запущено багато разів, та при цьому не було необхідності генерувати тестові дані вручну, а лише запустити тест на виконання.

3.5 Підсумки автоматизованого тестування

Підводячи підсумки автоматизації тестування веб-сервісу у SoapUI, слід відповісти на питання: які результати були отримані в процесі автоматизації тестування та після завершення автоматизованого тестування? Чи довелося досягти мети атестаційної роботи?

Головним завданням була перевірка якості веб-сервісу з використанням SoapUI.

Для цього була здійснена перевірка найголовнішої функціональності інтерфейсу OIM за допомогою написання автоматизованих тестів у середовищі SoapUI.

Як виявилося, у результаті автоматизованого тестування було знайдено ряд дефектів (багів), які за своєю важливістю можна поділити на високі, середні та низькі. Для тестування знайдених дефектів у процесі регресійного тестування були застосовані авто-тести, за допомогою яких значно зменшився час перевірки головного функціоналу інтерфейсу. Також, було здійснено дослідження функцій автоматизації, за допомогою якого були знайдені шляхи не тільки швидкого, а й ефективного тестування веб-сервісу. Тому на базі зробленого аналізу автоматизації можна зробити висновок, що автоматизація веб-сервісів є невід'ємною частиною життєвого циклу їх тестування.

Що ж стосується середовища для розробки автоматизованих тестів, можна сказати, що воно значно вплинуло на час виконання тестів. Також, плюсом було те, що коли автоматичні скрипти вже були написані, на аналіз результатів знадобилося менше часу, ніж на проведення того ж обсягу тестування вручну.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера, на якому буде виконуватися розробка.

4.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» [30] визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

4.2 Аналіз стану умов праці

Робота над проектом проходитиме в приміщенні багатоквартирного будинку. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп’ютером.

4.2.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 4.1.

Таблиця 4.1 – Розміри приміщення

Найменування	Значення
Довжина, м	5
Ширина, м	2,3
Висота, м	3,2
Площа, м2	11,5
Об’єм, м3	36,8

Згідно з [16] розмір площі для одного робочого місця оператора персонального комп’ютера має бути не менше 6 кв. м, а об’єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

4.2.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за [14] (табл. 4.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	750	680 ÷800
Висота простору для ніг, мм	730	не менше 600
Ширина простору для ніг, мм	660	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	470	400 ÷500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	800	700 ÷800

Приміщення кабінету знаходиться на другому поверсі п’яти поверхової будівлі і

має об’єм 36,8 м^3, площу – 11,5 м^2. У цьому кабінеті обладнано одне місця праце та два укомплектовані ПК.

Температура в приміщенні протягом року коливається у межах 14–28°С, відносна вологість — близько 50%. Швидкість руху повітря не перевищує 0,2 м/с. Шум в лабораторії знаходиться на рівні 50 дБА. Система вентилювання приміщення — природна неорганізована, а опалення — автономне.

4.2.3 Навантаження та напруженість процесу праці

Під час виконання випускної роботи за фізичним навантаженням робота відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи на ведені в [14].

Роботу за дипломним проектом визнано, такою, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви - для розробників програм тривалістю 10-15 хв. через кожну годину роботи;

4.2.4 Пожежна безпека

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання.

Запобігти утворенню горючого середовища (замінити горючі речовини і матеріали на негорючі і важкогорючі) не надається технічно можливим.

Простори усередині приміщень в межах, яких можуть утворюватися або знаходиться пожежонебезпечні речовини і матеріали відповідно до [20] відносяться до пожежонебезпечної зони класу П-ІІа. Це обумовлено тим, що в приміщенні знаходяться тверді горючі та важкозаймисті речовини та матеріали. Приміщенню, у якому розташоване робоче місце, присвоюється ІІ ступень вогнестійкості.

Продуктами згорання, що виділяються на пожежі, є: окис вуглецю; сірчистий газ; окис азоту; синильна кислота; акромін; фосген; хлор і ін. При горінні пластмас, окрім звичних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідридні кислоти, формальдегіди, хлористий водень, фосген, синильна кислота, аміак, фенол, ацетон, стирол [12].

4.2.5 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.3 Гігієнічні вимоги до параметрів виробничого середовища

4.3.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. Оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають

[15] і наведені в табл. 4.3:

Таблиця 4.3 – Норми мікроклімату робочої зони об’єкту

Період року	Категорія робіт	Температура C^0	Відносна вологість %	Швидкість руху повітря, м/с
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

Дане приміщення обладнане системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до [15]. Рівні позитивних і негативних іонів у повітрі мають відповідати [15].

Контроль параметрів мікроклімату в холодний і теплий період року здійснюється не менше 3-х разів на зміну (на початку, середині, в кінці).

4.3.2 Освітлення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає [13]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n, \qquad (4.1)$$

де Sb – площа віконних прорізів, м2;

Sn – площа підлоги, м2.

Sn = a·b = 5·2,3 = 11,5 м2,

S = 1/8·11,5 = 2,3 м2

Приймаємо 1 вікно площею S=3,08 м2.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні.

Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \qquad (4.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 11,5 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо:

$$n = \frac{300*11.5*1.1*1.5}{5400*0.575*2} \approx 0{,}916$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.3.3 Шум та вібрація, електромагнітне випромінювання

Рівень шуму, що супроводжує роботу користувачів персональних комп'ютерів (зумовлений як роботою системних блоків, клавіатури, так і друкуванням на принтерах, а також зовнішніми чинниками), коливається у межах 50–65 дБА [4]. У залах опрацювання інформації та комп'ютерного набору рівні шуму не повинні перевищувати 65 дБА.

Віброізоляція можливо здійснювати за допомогою спеціальної прокладки під системний блок, який послаблює передачу вібрацій робочого столу. Вібрація на робочому місці в приміщенні, що розглядається, відповідає нормам [15].

4.3.4 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти), тобто при V приміщення > 40 м3 на одного працюючого допускається природна вентиляція. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП.

Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

4.3.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

1) Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

– правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;

– експлуатацію сертифікованого обладнання;

– дотримання заходів електробезпеки;

– забезпечення оптимальних параметрів мікроклімату;

– забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);

– облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря.

2) Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

– постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, лампових патронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під'єднують до електромережі;

– постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;

– не тягнути за мережевий кабель, щоб витягти вилку з розетки;

– не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;

– не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;

– не залишати включені електроприлади без нагляду;

– не допускати потрапляння всередину електроприладів крізь вентиляційні отвори рідин або металевих предметів, а також не закривати їх та підтримувати в належній чистоті, щоб уникнути перегрівання та займання приладу;

– не ставити на електроприлади матеріали, які можуть під дією теплоти, що виділяється, загорітися (канцелярські товари, сувенірну продукцію тощо).

Вимоги безпеки при надзвичайних ситуаціях:

1) При раптовому припиненні подачі електричної енергії вимкнути всі пристрої ПК в такій послідовності: периферійні пристрої, ВДТ, системний блок, стабілізатор (або блок безперервного живлення). Витягнути вилки з розеток. При наявності ознак горіння (дим, запах горілого) необхідно вимкнути всі пристрої ПК, знайти місце загоряння і виконати всі можливі заходи для його ліквідації, попередивши терміново про це керівництво.

2) При замиканні, перевантаженні електричного струму на електричному обладнанні, внаслідок ураження грозової блискавки та ймовірної небезпеки ураженням електричним струмом, приймають наступне:

– попередження замикання здійснюється правильним вибором, монтажем експлуатації мереж;

– застосування захисту схем у вигляді швидкодіючих реле, а також вимикачів, плавких запобіжників.

4.4 Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом [21], приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Коефіцієнт використання вертикальних заземлювачів ηв в залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4…0,99. Взаємну екрануючу дію горизонтального заземлювача (з'єднувальної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача ηс.

Послідовність розрахунку.

1) Визначається необхідний опір штучних заземлювачів Rшт.з.:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \qquad (4.4)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.}=R_д$.

Підставивши числові значення у формулу (4.4), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом•м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр..} = \psi \cdot \rho, \qquad (4.5)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом·м.

$\rho_{розр.в} = 1,7 \cdot 40 = 68$ Ом•м

$\rho_{розр.г} = 5,5 \cdot 40 = 220$ Ом•м

4) Розраховується опір розтікання струму вертикального заземлювача $R_в$, Ом, за (4.5).

$$R_{в} = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_{в}} \cdot \left(\ln \frac{2 \cdot l_{в}}{d_{ст}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_{в}}{4 \cdot t - l_{в}} \right), \quad (4.6)$$

де $l_{в}$ – довжина вертикального заземлювача (для труб - 2−3 м; $l_{в}$=3 м);

$d_{ст}$ – діаметр стержня (для труб - 0,03−0,05 м; $d_{ст}$=0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за ф. (4.7):

$$t = h_{в} + \frac{l_{в}}{2}, \quad (4.7)$$

де $h_{в}$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді

$$t = 0,8 + \frac{3}{2} = 2,3 \text{ м}$$

Підставивши числові значення у формулу (4,6) отримуємо:

$$R_{в} = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом}$$

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_{в}$:

$$n = \frac{2 \cdot R_{в}}{R_{д}} = \frac{2 \cdot 18,5}{4} = 9,25 \quad (4.8)$$

І визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки $\eta_{в}$ =0,57 (табличне значення).

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_{в}$, шт:

$$n_{в} = \frac{2 \cdot R_{в}}{R_{д} \cdot \eta_{в}} = \frac{2 \cdot 18,5}{4 \cdot 0,57} = 16,2 \approx 16 \quad (4.9)$$

7) Визначається довжина з'єднувальної стрічки горизонтального заземлювача $l_{с}$, м:

$$l_c = 1,05 \cdot L_в \cdot (n_в - 1), \quad (4.10)$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$ – необхідна кількість вертикальних заземлювачів.

$l_c = 1,05 \cdot 3 \cdot (16 - 1) \approx 48$м

8) Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_c} \cdot \ln \frac{2 \cdot l_c^2}{d_{см} \cdot h_г}, \quad (4.11)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_г$ – глибина закладання горизантальних заземлювачів (0,5 м);

l_c - довжина з'єднувальної стрічки горизонтального заземлювача l_c, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0,95 \cdot 0,15 \cdot 0,5} = 8,1 \text{ Ом}$$

9) Визначається коефіцієнт використання горизонтального заземлювача $\eta_{с.}$ відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги $\eta_с$=0,3 (табличне значення).

10) Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_с + R_г \cdot n_в \cdot \eta_в} \le R_д. \quad (4.12)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \le R_д$$

11) При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

– іскри і дуги коротких замикань;

– перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої

перевантаження та наявність перехідного опору;

– іскри при розмиканні і розмиканні ланцюгів;

– розряди статичної електрики;

– необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

4.5 Охорона навколишнього природного середовища

Виконання дипломної роботи у купі з іншими факторами впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища» [27], Законом України «Про забезпечення санітарного та епідемічного благополуччя населення» [28], Законом України «Про відходи» [29].

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на знешкодження, утилізацію, тощо в ІТ галузі.

Вплив на атмосферне повітря при нормальних умовах праці не оказує, бо не має в приміщенні сканерів, принтерів та інших джерел викиду забруднюючих речовин в повітря робочої зони.

В процесі діяльності аналізу математичних методів оцінки надійності БСМ і програмних продуктів для імітаційного моделювання БСМ, вибіру найбільш підходящої системи для оцінки працездатності БСМ та оцінки впливу перешкод і потужності передачі радіосигналу на працездатність БСМ.виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

– відпрацьовані люмінесцентні лампи - і клас небезпеки;

– батарейки та акумулятори (малі) -ііі клас небезпеки;

– акумулятор для джерел безперебійного живлення -ііі клас небезпеки;

– змінні носії інформації - іv клас небезпеки;

– відходи друкуючих пристроїв - іv клас небезпеки;

– макулатура - іv клас небезпеки;

– матеріали пакувальні пластмасові забруднені (ємності з-під тонеру, фарби, інш.) - іv клас небезпеки;

– побутові відходи - іv клас небезпеки.

ВИСНОВКИ

У результаті виконання атестаційної роботи було проведено дослідження якості в процесі автоматизованого тестування веб-сервісу у SoapUI. Для цього були вирішені наступні завдання:

- був досліджений сучасний стан питання якості веб-сервісу;
- були досліджені методи тестування;
- розроблено автоматизовані тести для тестування якості веб-сервісу;
- була досліджена автоматизація тестування веб-сервісу;
- був проведений аналіз результатів автоматизованого тестування.

Було проведено позитивне та негативне тестування веб-сервісу, тобто виконувалися коректні та некоректні операції, використовувалися різні вхідні дані, генерація яких була розроблена у авто-тестах. Отримані результати автоматизованого тестування повинні бути використані для виконання регресійного та навантажувального тестування, для усунення дефектів веб-сервісу так, як тестування якості веб-сервісу направлено на перевірку відповідності вимогам або потребам та очікуванням користувача.

Дослідження функцій для автоматизації тестування було проведено для вимірювання продуктивності процесу автоматизації тестування, витрачених ресурсів та часу. На основі цього дослідження та аналізу результатів можна зробити висновок, що поставлена задача перевірки якості була виконана швидко та ефективно, тому що були виконані усі тести, які входять до складу перевірки найважливіших функцій інтерфейсу OIM за допомогою найбільш використовуваних функцій автоматизації SoapUI. Результати роботи обговорювалися на конференції «Майбутній науковець-2019».

В результаті проведеної роботи в розділі «4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ» було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника. Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері. А також розглянуті екологічні аспекти впливу на діяльність з розробки магістерської роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Introduction to testing WebServices [Электронный ресурс] – Режим доступа: www/ URL : http://www.qaielearning.com/KnowledgePapers/Introduction-to-Testing-Webservices.pdf – 10.03.2017 г. – Загл. с экрана.

2) Тестирование веб-сервисов с SoapUI [Электронный ресурс] – Режим доступа: www/ URL : https://automated-testing.info/t/testirovanie-veb-servisov-s-soapui/ – 10.09.2018 г. – Загл. с экрана.

3) Машнин, Т. Web-сервисы Java [Текст] / Т. Машнин – Санкт-Петербург, 2011. – 560 с.

4) Ньюкомер, Э. Веб-сервисы. XML, WSDL, SOAP и UDDI. Для профессионалов [Текст] / Э. Ньюкомер – Издательский дома «Питер», 2003. – 256 с.

5) SoapUI Documentation [Электронный ресурс] – Режим доступа: www/ URL : https://www.soapui.org/ – 20.09.2018 г. – Загл. с экрана.

6) Methods for testing WebServices [Электронный ресурс] – Режим доступа: www/ URL : https://docplayer.net/16566176-Methods-for-testing-webservices.html – 15.09.2018 г. – Загл. с экрана.

7) Таненбаум, Э. Компьютерные сети [Текст] / Э. Таненбаум, Д. Уэзеролл – Издательский дома «Питер», 2012. – 960 с.

8) Массе, М. Rest API [Текст] / М. Массе – Издательство «O'Reilly Media», 2011. – 116 с.

9) Освоение тестирования Rest API [Электронный ресурс] – Режим доступа: www/ URL : http://quality-lab.ru/rest-api-testing/ – 02.10.2018 г. – Загл. с экрана.

10) Смит, Г. Grails. Гибкость Groovy и надежность Java [Текст] / Г. Смит, П. Ледбрук – Издательство «Символ-Плюс», 2010. – 656с.

11) Наука, исследования, развитие. Техника и технология [Текст] : тез. докл. науч.-практ. конф. (сен. 2018) / отв. ред. В. Окулич-Козарин. – Познань: Diamond trading tour, 2018. – 52 с.

12) ГОСТ 12.1.044-89 ССБТ. Пожежовибухонебезпека речовин і матеріалів. Номенклатура показників і методи їх визначенняя. Постанова від 12.12.1989 № 3683 МВС СРСР. Режим доступу: www. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=51048

13) ДБН В.2.5-28:2018 Природнє і штучне освітлення. Режим доступу: www. URL: https://dbn.co.ua/load/normativy/dbn/dbn_v_2_5_28/1-1-0-1188

14) ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними

дисплейними терміналами електронно-обчислювальних машин. Постанова №14 від 14.07.1999. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/v0014410-96

15) ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку. Постанова N 37 від 01.12.99. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/va037282-99

16) ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. Постанова N 42 від 01.12.99. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/va042282-99

17) НПАОП 0.00-4.12-05 Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці. Наказ №15 від 26.01.05. Режим доступу: www. URL:https://dnaop.com/html/33829/doc-%D0%9D%D0%9F%D0%90%D0%9E%D0%9F_0.00-4.12-05

18) НПАОП 0.00-4.15-98 Про розробку інструкцій з охорони праці. Наказ №9 від 29.01.98 року. Режим доступу: www. URL: https://dnaop.com/html/64/doc-%D0%9D%D0%9F%D0%90%D0%9E%D0%9F_0.00-4.15-98

19) НПАОП 0.00-6.03-93 Порядок опрацювання та затвердження власником нормативних актів про охорону праці. Наказ №132 від 21.12.93. Режим доступу: www. URL: https://dnaop.com/html/32357/doc-%D0%9D%D0%9F%D0%90%D0%9E%D0%9F_0.00-6.03-93

20) ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою. Наказ від 15.06.2016 № 158. Режим доступу: www. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=65419

21) НПАОП 40.1-1.01-97 Правила безопасной эксплуатации электроустановок. Наказ №257 від 06.10.97. Режим доступу: www. URL: https://dnaop.com/html/1691/doc-%D0%9D%D0%9F%D0%90%D0%9E%D0%9F_40.1-1.01-97

22) НПАОП 40.1-1.32-01 Правила устройства электроустановок. Электрооборудование специальных установок. Наказ №272 від 21.06.2001. Режим доступу: www. URL: https://dnaop.com/html/1692/doc-%D0%9D%D0%9F%D0%90%D0%9E%D0%9F_40.1-1.32-01

23) ДСН 3.3.6.039-99 Санітарні норми виробничої загальної та локальної вібрації. Постанова №39 від 01.01.99 МОЗ України. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/va039282-99

24) ДБН В.2.5-67:2013 Опалення, вентиляція та кондиціонування. Режим доступу: www. URL: https://dbn.co.ua/load/normativy/dbn/1-1-0-1018

25) ГОСТ 12.1.018-93 ССБТ.Пожаровзрывобезопасность статического

электричества. Общие требования. Дата прийняття 21.10 93. Режим доступу: www. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=48681

26) НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями. Наказ від 14.02.2018 № 207 Режим доступу: www. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=77160

27) Закон України «Про охорону навколишнього природного середовища» . Вводиться в дію Постановою ВР № 1268-XII від 26.06.91, ВВР, 1991, № 41, ст.547. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/1264-12

28) Закон України «Про забезпечення санітарного та епідемічного благополуччя населення». Відомості Верховної Ради України (ВВР), 1994, № 27, ст.218. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/4004-12

29) Закон України «Про відходи». Відомості Верховної Ради України (ВВР), 1998, № 36-37, ст.242. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/187/98-вр

30) Закон України "Про охорону праці". Вводиться в дію Постановою ВР № 2695-XII від 14.10.92, ВВР, 1992, № 49, ст.669. - Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/2694-12

ДОДАТОК А

Електронні плакати

АКТУАЛЬНІСТЬ АВТОМАТИЗАЦІЇ ВЕБ-СЕРВІСУ

- *Скорочує час на перевірку простих сценаріїв;*
- *У більшості випадків для користувача інтерфейс не передбачений;*
- *При регресійному тестуванні;*
- *При перевірці структури відповіді*
- *wsdl файлу;*
- *Виключає людський фактор при тестуванні;*
- *При перевірці валідаційних повідомлень;*

- *При перевірці правильності пошуку даних;*
- *При димовому тестуванні;*

- *При навантажувальному тестуванні.*

SOAPUI

SoapUI - це додаток для тестування веб-сервісів з відкритим вихідним кодом для сервіс-орієнтованих архітектур (SOA) і передачі станів уявлень (REST). Його функціональні можливості включають перевірку веб-сервісів, запуск, розробку, моделювання, функціональне тестування, автоматизоване тестування, тестування навантаження і безпеки.

Як веб-сервісу було розплянуто інтерфейс Oracle Identity Manager - веб-сервіс, призначений для управління користувачами в веб-додатку.

ВИДИ СЦЕНАРІЇВ ДЛЯ АВТОМАТИЗАЦІЇ

- ***перевірка створення користувача*** *(к-ть позитивних тестів: 2, к-ть негативних тестів: 1)*
- ***перевірка можливості зміни параметрів користувача*** *(к-ть позитивних тестів: 1, к-ть негативних тестів: 3)*
- ***додавання списку груп користувачеві*** *(к-ть позитивних тестів: 2, к-ть негативних тестів: 4)*
- ***перевірка можливості зміни статусу користувача*** *(к-ть позитивних тестів: 2, к-ть негативних тестів: 2)*
- ***отримання повної інформації про створеному користувача, про користувача зі зміненими параметрами, про групу користувача*** *(к-ть позитивних тестів: 3, к-ть негативних тестів: 0)*

ПРОЦЕС АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ

1) Створення тест кейсів

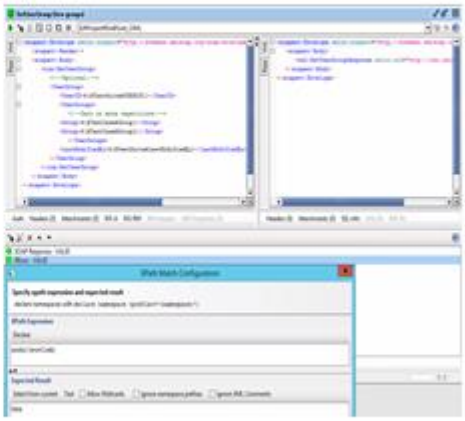
Негативний тест кейс для перевірки створення користувача в інтерфейсі OIM

Позитивний тест кейс для перевірки додавання списку груп користувачеві в інтерфейсі OIM

ПРОЦЕС АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ

2) Створення автоматизованих кейсів

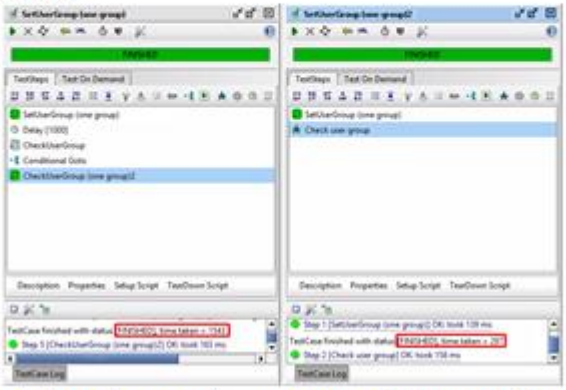
Результат перевірки створення користувача, якщо він вже існує в інтерфейсі OIM



Результат перевірки додавання списку груп користувачеві в інтерфейсі OIM

ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ

Для дослідження порівнюватися між собою такі функції автоматизації як Groovy Script і JDBC Request разом з Conditional Goto. Найефективніше використовувати кілька автоматизованих тестових кроків в одному тесті для перевірки коректного додавання даних або один тестовий крок для перевірки тієї ж функції?



Результат роботи двох тестів, виконаних за допомогою різних функцій

ВИСНОВКИ

Що ж показала автоматизація тестування?

Дослідження функцій автоматизації тестування було проведено для вимірювання продуктивності процесу автоматизації, витрачених ресурсів і часу.
За результатами досліджень видно, що час виконання тесту з використанням groovy script значно менше, ніж час виконання тесту з conditional goto і jdbc request, що робить виконання тесту швидше і ефективніше. Швидкість виконання тестів здійснюватися також за рахунок запуску всіх авто-тестів паралельно, в той час як при ручному тестуванні тести запускаться послідовно. За допомогою функцій автоматизації були створені різні тестові дані, за рахунок чого виконувалося позитивне і негативне тестування. Результати досліджень були використані при регресійному і нагрузочном тестуванні, в ході чого було знайдено ряд дефектів. Для тестування знайдених дефектів в процесі регресійного тестування були застосовані авто-тести, які значно зменшили час перевірки головного функціоналу інтерфейсу. Тому можна зробити висновок, що інтерфейс Oracle Identity Manager був протестований швидко і ефективно з використанням ключових функцій автоматизації SoapUI.

НАУКОВА КОНФЕРЕНЦІЯ

«Майбутній науковець 2019»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ТОВ НАУКОВО-ВИРОБНИЧЕ ПІДПРИЄМСТВО "Зоря"
ДУ ЛУГАНСЬКИЙ НАУКОВО-ДОСЛІДНИЙ ЦЕНТР МВС України

„Майбутній науковець – 2019"