Форма № Н-9.02

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
імені адмірала Макарова

Херсонська філія

Кафедра інформаційних технологій та фізико-математичних дисциплін

"Допущений до захисту"
Завідувач кафедри
д.т.н., доцент Гучек П.Й.
"____" _____ 2020 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

на здобуття ступеня вищої освіти "магістр"

на тему: "Дослідження засобів тестування та розробка системи автоматизації тестування веб-додатку"

Виконав: студент VI курсу, групи 6157м спеціальності
122 - "Комп'ютерні науки", ОПП - "Інформаційні управляючі системи та технології"
(шифр і назва спеціальності та освітньо-професійної програми)

Потужний І.Р.
(прізвище та ініціали)

Керівник Гайда А.Ю.
(прізвище та ініціали)

Рецензент Новіков Ю.М.
(прізвище та ініціали)

__.12 - 2020 року

.

Анотація

Дипломна робота на тему «Дослідження засобів тестування та розробка системи автоматизації тестування веб-додатку» на здобуття освітньо-кваліфікаційного рівня «Магістр» зі спеціальності «Комп'ютерні науки».

Метою роботи є підвищення якості програмного забезпечення завдяки створенню системи, що поєднує у собі кілька способів тестування веб-додатку та її програмна реалізація.

Методи досліджень: використовуючи різні методи тестування програмного забезпечення здійснюється їх порівняння та вибір найоптимальніших, що доповнюють один одного. Здійснюється реалізація системи з використанням безперервної інтеграції, управління конфігурацією. Також задіяні методи об'єктно-орієнтованого програмування для проектування програмних засобів системи.

Результати дослідження: проведено аналіз існуючих методів тестування веб-додатків, здійснено програмну реалізацію, яка використовує технологію безперервної інтеграції для ефективної роботи з різними методами тестування програмної системи. Результати роботи можуть бути використані в будь-яких системах, для яких потрібно створити та автоматизувати процес тестування веб-додатків, а також в навчальному процесі.

Орієнтовні напрямки розвитку досліджень: автоматизація тестування програмного забезпечення та методи тестування веб-додатків.

Робота викладена на 100 сторінках друкованого тесту, містить 14 рисунків, 3 таблиць, 2 додатків та перелік джерел з 31 найменувань.

Аннотация

Дипломная работа на тему «Исследование средств тестирования и разработка системы автоматизации тестирования веб-приложения» на получение образовательно-квалификационного уровня «Магистр» по специальности «Компьютерные науки».

Целью работы является повышение качества программного обеспечения благодаря созданию системы, объединяющей в себе несколько способов тестирования веб-приложения и его программная реализация.

Методы исследований: используя различные методы тестирования программного обеспечения, осуществляется их сравнение и выбор оптимальных, дополняющие друг друга. Осуществляется реализация системы с использованием непрерывной интеграции, управления конфигурацией. Также задействованы методы объектно-ориентированного программирования для проектирования программных средств системы.

Результаты исследования: проведен анализ существующих методов тестирования веб-приложений, осуществлено программную реализацию, которая использует технологию непрерывной интеграции для эффективной работы с различными методами тестирования программной системы. Результаты работы могут быть использованы в любых системах, для которых нужно создать и автоматизировать процесс тестирования веб-приложений, а также в учебном процессе.

Ориентировочные направления развития исследований: автоматизация тестирования программного обеспечения и методы тестирования веб-приложений.

Работа изложена на 100 страницах печатного теста, содержит 14 рисунков, 3 таблиц, 2 приложений и перечень источников из 31 наименований.

Annotation

Work on the topic "Research of testing tools and development of a web application testing automation system" for obtaining the educational qualification level "Master" in the specialty "Computer Science".

The aim of the work is to improve the quality of software by creating a system that combines several methods of testing a web application and its software implementation.

Research methods: using various software testing methods, their comparison and selection of the optimal ones, complementing each other, are carried out. The implementation of the system is carried out using continuous integration, configuration management. Also, methods of object-oriented programming are used for the design of system software.

Research results: the analysis of existing methods for testing web applications was carried out, a software implementation was carried out that uses continuous integration technology to effectively work with various methods of testing a software system. The results of the work can be used in any systems for which it is necessary to create and automate the process of testing web applications, as well as in the educational process.

Approximate directions of research development: software testing automation and web application testing methods.

Thesis on the topic "Research of testing tools and development of a web application testing automation system" for obtaining an educational qualification level "Master" in the specialty "Computer Science".

The aim of the work is to improve the quality of software by creating a system that combines several methods of testing a web application and its software implementation.

Research methods: using various software testing methods, their comparison and selection of the optimal ones, complementing each other, are carried out. The implementation of the system is carried out using continuous integration, configuration

management. Also, methods of object-oriented programming are used for the design of system software.

Research results: the analysis of existing methods for testing web applications was carried out, a software implementation was carried out that uses continuous integration technology to effectively work with various methods of testing a software system. The results of the work can be used in any systems for which it is necessary to create and automate the process of testing web applications, as well as in the educational process.

Approximate directions of research development: software testing automation and web application testing methods.

The work is presented on 100 pages of the printed test, contains 14 figures, 3 tables, 2 appendices and a list of sources from 31 titles.

ЗМІСТ

ВСТУП

Актуальність теми. Комп'ютерні технології все глибше проникають у наше повсякденне життя. Програмне забезпечення здійснює управління роботою безлічі речей навколо нас – від мобільних телефонів і комп'ютерів, до пральних машин, кредитних карт і автомобілів. У будь-якому випадку, всі ми зіткалися з тими чи іншими помилками у роботі програм: текстовий редактор намертво «завис» при роботі з документом, банкомат «з'їв» картку або сайт ніяк не завантажиться – все це, аж ніяк не полегшує нам життя.

Саме тому, у зв’язку з «діджиталізацією» усіх сфер людської діяльності та стрімким розвитком IT ринку, на передній план виступають питання кібербезпеки, захисту від цифрових загроз та надійності програмних продуктів.

Все, що виробляється людиною – може містити помилки. Однак не всі помилки однаково небезпечні – для різних програмних систем, рівні ризику можуть відрізнятися. Деякі з них можуть бути незначними, в той час як інші мати руйнівні наслідки. Саме тому, будь-який продукт потребує перевірки – тестування, перш ніж його можна буде ефективно і безпечно використовувати. Те ж саме справедливо і для програмного забезпечення. Всебічна перевірка працездатності програмного забезпечення, виявлення недоліків під час розробки, впровадження, функціонування та підтримки, забезпечується комплексом із заходів його тестування.

Тестування, проведене на всіх етапах розробки, дозволяє значно поліпшити якість, надійність і продуктивність системи. Під час перевірки, команда тестувальників засвідчується в тому, що програмний продукт належним чином виконує всі задокументовані функції і, одночасно, не робить того, що не повинен.

Для забезпечення високої якості кінцевого IT-продукту, критично важливим є включення тестування в життєвий цикл розробки програмного забезпечення. Особливе значення має впровадження тестування саме на на

ранніх стадіях роботи над проектом, оскільки такий підхід дозволяє значно знизити витрати на усунення виявлених помилок.

Будь-які ІТ-рішення, стосовно яких не застосовуються перевірочні заходи, можуть обернутися суттєвими економічними та іміджевими втратами. Значні обсяги об'єктів перевірки обумовлюють широке використання автоматизованих тестів. Але, для підвищення якості вихідного програмного забезпечення, зазвичай недостатньо використання одного методу тестування. Саме тому нагально постає потреба не тільки у перевірці програмного продукту, а і у розвитку методів його тестування. А саме, у побудові цілої системи тестування програмного забезпечення, з обов'язковим використанням інструментів автоматизованого тестування.

Мета і завдання дослідження. Метою даної роботи є дослідження основних існуючих методів тестування веб-додатків та створення модульної системи автоматизованих тестів для перевірки програмних продуктів. Вказана система тестів повинна мати більш високу ефективність пошуку дефектів у програмних додатках у порівнянні з існуючими рішеннями.

Об'єкт дослідження: процеси розробки веб-додатків.

Предмет дослідження: моделі та методи тестування веб-додатків.

Методи дослідження: використовуючи методи дослідження системного аналізу, для вирішення проблеми надійної перевірки функціонування веб-додатків,.

Наукова новизна одержаних результатів:

- запропоновано метод модульної системи перевірки програмного забезпечення, з поєднанням декількох методів тестування;
- розроблено модель комплексного тестування веб-додатків декількома методами.

Практичне значення отриманих результатів. Спроектовано модель системи модульного тестування веб-додатків. Розроблено алгоритм та

інструменти функціонального тестування програмного веб-продукту. Здійснено програмну реалізацію розробленої моделі за багатьма критеріями перевірки.

Впровадження результатів ДР. Розроблена у дипломній роботі модель системи автоматизованих тестів успішно впроваджена в процес розробки IT-проекту компанії “DVLOPBIZ”. Використання вказаної системи перевірки підтвердило її практичну ефективність при тестуванні веб-продуктів, що позитивно відобразилось на надійності функціонування програмного забезпечення.

					КР 122.6157м.ПЗ Р1		
Зм.	Лист	№ докум	Підпис	Дата			
Студент		Потужний І.Р.			Аналіз методологій створення веб-додатків та рівнів їх тестування	Літера	Лист 12 / Листів 19
Керівник		Гайда А.Ю.					
Консульт.		Дудченко О.М.				ХФ НУК ім. адм. Макарова	
.Рецензент		Новиков Ю.М.					
Зав. Каф		Гучек П.Й.					

1. АНАЛІЗ МЕТОДОЛОГІЙ СТВОРЕННЯ ВЕБ-ДОДАТКІВ ТА РІВНІВ ЇХ ТЕСТУВАННЯ

1.1 Види веб-додатків і технології їх створення

1.1.1 Поняття веб-додатку та його класифікація

Веб-додаток – клієнт-серверний додаток, у якому клієнт взаємодіє з веб-сервером за допомогою браузера. Структура веб-додатку розподілена між сервером і клієнтом, зберігання даних здійснюється переважно на сервері, обмін інформацією здійснюється по мережі.

На теперішній час існує велике різноманіття інтернет-додатків (сайтів). За динамікою надаваємої для клієнта інформації їх можна поділити на дві великі групи: статичні та динамічні.

Статичні – це прості сайти з незмінними веб-сторінками, які оновлюються власниками в ручному режимі. Кожна сторінка укладається в окремому документі та зберігається у готовому вигляді на сервері. Кожен клієнт отримає однакову сторінку на свій запит до сервера.

Динамічні сайти можуть змінюватись, залежно від дій клієнта і формуються на сервері на основі шаблонів та бази даних.

Архітектура клієнтсерверу найчастіше застосовується при створенні таких сайтів – інтернет-додатків. Процес роботи з веб-додатками здійснюється наступним чином. Клієнт-юзер здійснює запит на веб-сторінці веб-додатку (сайту). Додаток обробляє отриману інформацію, та надсилає її до веб-сервера через мережу. Веб-сервер, в свою чергу, обробляє запит, та через мережу повертає відповідь до веб-додатку створюючи оновлену веб-сторінку для клієнта.

За сферою застосування інтернет-додатки можна поділити на:

- системи бронювання і покупки: квитки, готелі, товари, послуги і т. ін;
- розважальні портали;

- фінансові та банківські інтернет-портали з функціями: замовлення послуг онлайн; калькулятора кредитів; конвертатора валют; інтернет-банкінгу та ін.;
- соціальні мережі;
- ігри;
- освітні, навчальні канали, сайти телепрограм, газет;
- веб-версії програмного забезпечення;
- біржі контенту, фрілансу і т.п. [3].

За призначенням серед існуючих веб-додатів можна виділити:

- інтернет магазини;
- сайт- вітрини (каталоги);
- пошукові системи;
- рейтинги;
- лендінгові сторінки;
- промо-сайти (презентаційні сайти);
- веб-галереї (портфоліо);
- особисті блоги;
- корпоративні сторінки;
- форми відправки повідомлень;
- форуми;
- upload-форми (форми завантаження файлів);
- системи опитування населення;
- content managment system (CMS);
- поштові скриньки (онлайн пошта);
- особисті системи галузі конкретного спрямування та ін.

Під час створення, впровадження і підтримки зазначених веб-продуктів, розробниками можуть бути здійснені помилки, через які веб-додатки мають

певні «слабкі місця». Серед таких типових потенційних уразливостей можна виділити:

- відсутність контролю або неповний контроль вхідної інформації від клієнта;
- перевантаження буферу обміну інформації;
- некоректна обробка поточних даних додатком або сервером;
- некоректна робота веб-додатку з файлами, під час виконання GET та POST запитів, при зовнішній передачі імені файлу;
- некоректна робота веб-додатку з правами доступу;
- помилки у кодуванні роботи з паролями (обробка, збереження, відправлення інформації);
- конфлікти при роботі програми під час завантаження файлів на сервера;
- порушення логіки виконання процесів у веб-додатку, при обробці окремих допустимих вхідних даних, що може привести до непрогнозованих наслідків у його роботі;
- доступ клієнта до бази даних програми або відображення службової інформації, не призначеної для клієнта, при помилках у роботі веб-додатку;
- помилки при роботі з паролями;
- помилки при перевищенні кількості запитів до бази даних;
- некоректна обробка вхідних даних при роботі з конкретною БД (SQL-ін’єкції);
- перевантаження серверу під час нормальної роботи або при обміні некоректною вхідною інформацією, через неоптимізований код програми;
- ураження сайтів DoS та DDoS атаками та т. ін.

1.1.2 Технології створення веб-додатків

За період існування мережі Інтернет зміст веб-додатків, задачі які вони вирішують та їх структура дуже змінилися. На теперішній час існує багато засобів для створення веб-додатків. Мова HTML є базовою в області технологій створення сайтів, оскільки відносно легка в освоєнні. Але надмірна простота є і

її недоліком. HTML (від англ. Hyper Text Markup Language – мова розмітки гіпертексту) чудово відповідала вимогам раннього періоду розвитку технологій створення веб-продуктів, але з подальшим його розвитком виникли істотні проблеми. З найпростіших засобів збереження HTML сторінок вони «виросли» до готових рішень, які орієнтовані на підтримку роботи цілих корпоративно-інформаційних систем [3].

Нині спостерігається стійка тенденція застосування таких платформ, що дозоляють створювати та ефективно керувати їх інформаційним наповненням. Серед них можна виділити такі, що базуються на серверних технологіях. Розглянемо деякі з них, а саме: AJAX , CGI, ISAPI, ASP, JSP.

AJAX (від англ. Asynchronous Javascript and XML). Насправді, AJAX не є новою технологією. Оскільки JavaScript і XML існують дуже давно, а AJAX – це синтез зазначених технологій. Він частіше всього асоціюється з терміном Web 2.0 і подається як найновіший Web-додаток.

При використані AJAX відсутня необхідність оновлювання усієї сторінки кожен раз, так як оновлюється її конкретна частина. Це набагато зручніше, адже непотрібно довго чекати, та комфортніше так, як не всі мають безлімітний інтернет. Насправді у цьому випадку розробнику необхідно стежити, щоб користувач мав поняття про те, що відбувається на цій сторінці. Це можна реалізувати з використанням індикаторів завантаження, текстових повідомлень про те, що йде обмін даних з сервером. Також потрібно пам’ятати про те, що не всі браузери підтримують AJAX і Javascript може бути вимкнений користувачем. Отже, переваги AJAX у тому, що:

- є можливість створення комфортного веб-інтерфейсу;
- є активна взаємодія з користувачем;
- є часткове перезавантаження сторінки замість повного;
- є комфортність використання.

AJAX використовує два методи роботи з веб-сторінкою:

- зміна веб-сторінки не перезавантажуючи її одночасно із динамічним зверненням до сервера;
- здійснення декількох способів, а саме XMLHttpRequest та використання техніки скритого фрейму.

Технологія CGI (від англ. Common Gateway Interface), застосовує у складі ресурсу Internet інтерактивні елементи на базі застосувань, що забезпечують передачу потоку даних від об'єкта до об'єкта. У загальному випадку принцип роботи CGI виглядає так: користувач заповнює на веб-сторінці певну форму і натискає на кнопку, після чого вбудований у HTML-код рядок виклику CGI-скрипта запускає відповідну програму CGI і передає їй управління процесом обробки інформації. Введені користувачем дані відсилаються цій програмі, а вона у свою чергу вбудовує їх в іншу сторінку, відправляє поштою або трансформує іншим способом. Скрипти CGI розміщуються на сервері у спеціально відведеній для цих цілей директорії CGI-BIN. Технологія CGI зазвичай реалізується двома методами: або з використанням програм, написаних мовою PERL (Practical Extraction and ReportLanguage), або із застосуванням мови C, оскільки більшість UNIX-сумісних платформ включають вбудований транслятор цієї мови. Подібні програми мають розширення .cgi. Необхідно зазначити, що PERL є інтерпретованою мовою, тому не вимагається додаткової компіляції. Крім згаданих можливостей за допомогою цієї технології можна організувати систему показу послідовності рекламних банерів або автозавантаження файлів на сервер, створити форму відправки електронного листа безпосередньо зі сторінки сайту або службу віртуальних листівок. Серед переваг CGI слід назвати їх незалежність від клієнтського програмного забезпечення. Головний недолік полягає в тому, що для установки і створення застосувань CGI на сервері потрібно володіти правами адміністратора, оскільки ці програми при запуску здатні порушити нормальне функціонування серверного комп'ютера і дестабілізувати роботу мережі.

ISAPI (від англ. Internet Server Application Programming Interface) – зовнішня оболонка серверу компанії Microsoft, призначена керувати цим сервером. ISAPI набула своєї індивідуальності та зручності і має підтримку майже всіх виробників програмних засобів. ISAPI-програми, насамперед є такими видами веб-додатками, завдання яких, оброблювати запити користувачів та відображати їх у вигляді потоку HTML, знаходження якого базується у клієнт-браузері.

ASP (від англ. Active Server Pages) – середовище програмування, що забезпечує можливість комбінування HTML, скриптів та компонентів для створення динамічних веб-додатків. Можливість вбудування у веб-сторінки скриптів дозволяє логічним чином об'єднати оформлення даних, отриманих з різних джерел, наприклад з баз даних.

Правильне створення сучасних веб-додатків складається з інкапсуляції бізнес-логіки в окремі компоненти, написані за технологією COM. Технологія ASP, у цьому випадку є суміжною ланкою між цими компонентами та інтерфейсом веб-додатків.

Для використання Active Server Pages не потрібні специфічні браузери. Всі ASP-скрипти запускаються та виконуються на веб-сервері, при цьому браузер отримує тільки остаточні HTML-файли.

Клієнт дає запит на ASP-сторінку веб-серверу. Сервер у свою чергу приймає цей запит та починає обробку. Розширення файлу (.asp) означає, що даний файл містить ASP-скрипт, та аналізує його зміст, послідовно адаптуючи з виконанням вставок ASP-коду. Цей код, в свою чергу, може вміщувати звертання до різних джерел даних, оброблювати отримані дані та додавати вміст генеруючої сторінки. Як результат - формується HTML-сторінка (без вмісту ASP-коду), яка, насамперед, відправляється назад до клієнта.

ASP-код, це код який потрібно виконувати на сервері, розміщуючи всередині спеціальні тєги. Тому, так як даний код обробляється на сервері, то у

користувача немає доступу до нього. Сам код може бути написаний з використанням JScript (JavaScript) або Visual Basic Scripting Edition.

З технологією ASP можна також використовувати інші мови програмування. Синтаксис VBScript є набагато простішим інших мов.

JSP (від англ. Java Server Pages) – є серверною технологією, роль якої є створення динамічних веб-сторінок. Вона розширює технологію сервлетів, аби допомогти розробникам створити динамічну сторінку з HTML подібним синтаксисом.

Створення уявлень, насамперед, підтримується також і в сервлетах, хоча в такому випадку код виглядає складно і є схильним до помилок. Також було відмічено, що велика кількість елементів веб-сторінки є статичними і саме тому JSP-сторінка більше підходить до веб-сторінок. За можливості необхідно уникати бізнес-логіки на JSP-сторінці та використовувати цю сторінку тільки у вигляді представлення. Тому рекомендується використовувати JSP-actions елементи або JSTL-теги замість написання JSP- скриптів.

Однією з переваг в JSP є «гаряча розгортка». А саме, є можливість замінити попередню сторінку на іншу в контейнері і користувачам буде відображатися нова JSP-сторінка. Отже немає необхідності компілювати цілий проект або перезавантажувати сервер для оновлення деяких сторінок.

Якщо подивитися на код з середині створеної JSP сторінки, то він буде мати вигляд HTML та відрізнятиметься від java-класу. Конвертацією JSP сторінок в HTML-код займається контейнер, який створює сервлет для використання у веб-додатку.

1.2 Аналіз розробки програмного забезпечення

1.2.1 Структура веб-додатку

Зазвичай щоб створити програмне забезпечення за основу покладають 3 основні компоненти: клієнтська частина; серверна частина; база даних.

Клієнтська частина реалізує інтерфейс користувача, формує запити до сервера, та оброблює відповіді від нього.

Серверна частина утворюється з програми або скрипту, які виконують обробку запитів користувача та відповідає за такий функціонал:

- централізовану обробку та збереження даних;
- підтримку веб-інтерфейсу користувача;
- інформування користувачів у вигляді e-mail повідомлень про події в системі;
- взаємодію з профайлами клієнтів.

Зазвичай серверна частина створюється завдяки PHP. Браузер надсилає запит до сервера, після кожного переходу клієнта за необхідним посиланням. Сервер займається обробкою цього запиту, звертаючись до певного PHP-скрипта, який і є основою формування веб-сторінки, що описується мовою розмітки HTML та надсилає її клієнту інтернет-мережею. Як наслідок, клієнт отримує дані, що відображаються певною веб-сторінкою.

База даних знаходиться на сервері та являє собою організовану структуру, призначенням якої є збереження інформації, зміни зв'язаних даних та їх надання у разі необхідності. За зв'язки з базами даних відповідає саме цей сервер. PHP-скрипт, має змогу звертатися до бази даних, отримуючи потрібні дані, що створюють сторінки за запитом клієнт-юзера. У форматі, наприклад, блогу, збережена у базі даних інформація – це, як правило, коментарі, пости, тощо.

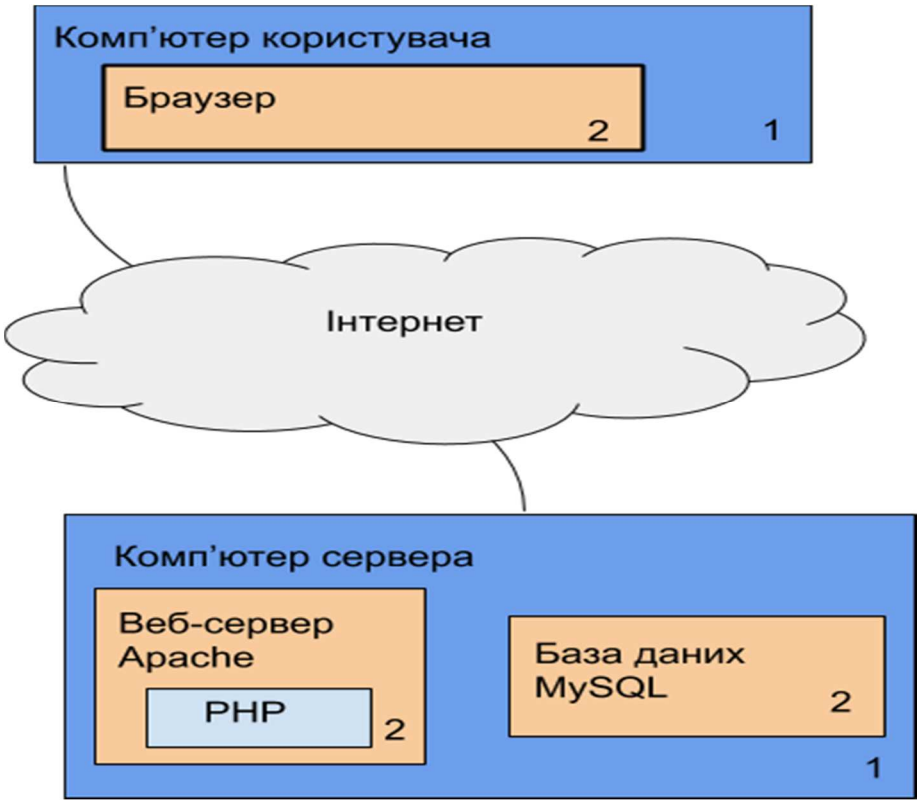

Рисунок 1.1 – Схематичне зображення зв'язку між серверною частинами і базою даних: 1 – комп'ютер, 2 – програма, 3 – зв'язок між програмами

На рисунку 1.1 проілюстрована схема зв'язків частин веб-продукту. Інтернет-браузер надсилає HTTP-запити до веб-серверу, який, в свою чергу, викликає написаний розробником PHP-скрипт. У разі необхідності, PHP-скрипт звертається до бази даних та у браузері повертає клієнту веб-сторінку.

1.2.2 Моделі розробки програмного забезпечення

При розробці програмних продуктів використовується декілька ефективних моделей, які перевірені часом та добре себе зарекомендували. Вибір, якою моделлю користуватися, залежить від специфіки проекту, розміру фінансових інвестицій, особистих переваг замовника.

Розглянемо найбільш поширені моделі розробки програмних продуктів.

Каскадна модель («Waterfall model») – у цій моделі легко керувати проектом. Вона складається з таких чітко визначених етапів: вимоги, проектування, розробка, тестування і підтримка. Завдяки її «жорсткому

фундаменту», розробка проходить швидко, вартість та строки здачі завчасно відомі. Каскадна модель дає відмінний результат у проектах з чітко та завчасно визначеними вимогами. Через чітко прописану послідовність дій, продукти, що розробляються завдяки даній моделі без обґрунтованого вибору, можуть мати недоліки, про які стає відомо лише у кінці розробки. Вартість внесення змін висока через те, що для їх впровадження, треба чекати завершення всього проекту повністю. Але, фіксована ціна, зазвичай переважає «мінуси».

Каскадна модель використовується в умовах, коли:

- всі зміни відомі та зафіксовані;
- суперечливих вимог немає;
- немає проблем з наявністю програмістів потрібної кваліфікації;
- обсяги проекту невеликі.

V-подібна модель («V-Model») – застосована до систем, яким особливо важливо безперебійне функціонування. Структура моделі успадкована від каскадної моделі, а саме її послідовність. Особливістю моделі можна вважати те, що вона спрямована на ретельну перевірку і тестування продукту, що знаходиться вже на початкових стадіях проектування. Стадія тестування проводиться одночасно з відповідною стадією розробки. Наприклад, під час кодування пишуться модульні тести.

За потреби ретельного тестування продукту, V-модель чітко виконує поставлену задачу: валідація та верифікація. Для проектів малих та середніх обсягів - вимоги чітко визначені.

Інкрементна модель (“Incremental Model”) – у цій моделі певні вимоги до системи поділяються на різні збірки.

Ці моделі мають кілька циклів розробки, які разом складають життєвий цикл - «мульти-водоспад». Цикл поділений на дрібніші, легко створювані модулі. Кожен з цих модулів проходять через певні фази визначення вимог: проектування, кодування, впровадження та тестування. Процедура розробки інкрементної моделі передбачає випуск на першому етапі продукту в базовій

функціональності, а потім оновлення та додавання нових функцій, так званих «інкрементів». Процес триває до створення повної моделі.

Інкременті моделі використовуються там, де окремі запити можуть бути формалізовані і реалізовані.

Коли основні вимоги до системи чітко визначені і зрозумілі – використовують інкрементну модель. У той же час деякі деталі можуть доопрацьовуватися поетапно. Це все обумовлюється потребою швидкого виходу продукту на ринок.

RAD-модель («rapid application development model») – різновид інкрементної моделі. У цій моделі компоненти або функції розробляються кількома командами професіоналів паралельно. Тимчасові рамки одного циклу жорстко обмежені. Створені модулі потім інтегруються в один робочий прототип. Синергія дозволяє дуже швидко надати клієнту для огляду робочий проект з метою отримання зворотного зв'язку і внесення змін.

Модель швидкої розробки додатків можна поділити на фази:

- бізнес-моделювання – визначення переліку інформаційних потоків між різними підрозділами;
- моделювання даних – інформація, зібрана на попередньому етапі, використовується для визначення об'єктів і інших сутностей, необхідних для циркуляції інформації;
- моделювання процесу – інформаційні потоки пов'язують об'єкти для досягнення цілей розробки;
- збірка програми – використовуються кошти автоматичного складання для перетворення моделей системи автоматичного проектування в код;
- тестування: тестуються нові компоненти і інтерфейси.

Таку модель можна використовувати тільки за наявності вузько-спеціалізованих фахівців. Бюджет проекту великий, адже потрібно оплачувати не тільки роботу цих фахівців, а також вартість готових інструментів автоматизованого складання.

При впевнених знаннях RAD-модель може бути обрана протягом 2-3 місяців.

Гнучка модель (“Agile Model”) – у цієї методології розробки, після кожної ітерації, замовник може спостерігати результат і розуміти, задовольняє він його чи ні. До її недоліків відносять те, що через відсутність конкретних формулювань результатів складно оцінити трудовитрати і вартість, необхідні на розробку. Екстремальне програмування (XP) є одним з найбільш відомих застосувань гнучкої моделі на практиці.

Основу такої моделі складають нетривалі щоденні зустрічі-«Scrum» і регулярно повторювані збори (раз в тиждень, раз на два тижні або раз на місяць), які називаються «Sprint».

Методологія підходить для великих або націлених на тривалий життєвий цикл проектів, які постійно адаптуються до умов ринку. Відповідно, в процесі реалізації вимоги змінюються. Варто згадати клас творчих людей, яким властиво генерувати, видавати і випробувати нові ідеї щотижня або навіть щодня. Гнучка розробка найкраще підходить для цього психотипу керівників.

Найчастіше Agile-модель застосовується в динамічному бізнесі, тобто, коли потреби користувачів постійно змінюються. Зміни у цій моделі реалізуються за меншу ціну через постійні інкременти. На відміну від каскадної моделі, у гнучкої моделі для старту проекту достатньо невеликого планування.

Ітераційна модель («Iterative Model») – ця модель життєвого циклу не потребує повної специфікації вимог. Замість цього, створення починається з реалізації частини функціоналу, що стає базою для визначення подальших вимог. Версія може бути неідеальна, головне, щоб вона працювала. Розуміючи кінцеву мету, робота планується так, щоб кожен крок був результативним, а кожна версія - працездатна.

Для оптимального використання ітераційної моделі, потрібно сформувати чітко визначені вимоги до кінцевої системи. Основне завдання повинне бути визначене, але деталі реалізації можуть змінюватися з плином часу.

Спіральна модель («Spiral Model») – ця модель схожа на інкрементну, але з акцентом на аналіз ризиків. Вона добре працює з вирішенням критично важливих бізнес-задач, коли невдача несумісна з діяльністю компанії, в умовах випуску нових продуктових лінійок, при необхідності наукових досліджень і практичної апробації.

Дана модель передбачає 4 етапи кожного витка:

1) планування;
2) аналіз ризиків;
3) конструювання;
4) оцінка результату (при задовільній якості перехід до нового витка).

Ця модель не підійде до малих проектів. Вона резонна для складних і дорогих, наприклад, таких, як розробка системи документообігу для банку, коли кожен наступний крок вимагає більшого аналізу для оцінки наслідків, ніж програмування.

1.3 Рівні і види тестування програмного забезпечення веб-додатків

1.3.1 Поняття тестування програмного забезпечення

Тестування – в широкому сенсі, це одна з технік контролю якості, що включає в себе активності з планування робіт (Test Management), проектування тестів (Test Design), виконання тестування (Test Execution) і аналізу отриманих результатів (Test Analysis).

Існує декілька визначень тестування програмного забезпечення. Це і процес дослідження та випробовування програмного продукту з метою з’ясування відповідності реального та очікуваного його функціонування, з використанням певного комплексу перевірочних заходів. І процес спостереження за виконанням в спеціальних умовах і винесення на цій основі оцінки будь-яких аспектів її роботи. А також, процес перевірки, з метою виявлення ситуацій, в яких поведінка програми є неправильною, небажаною або

не відповідає специфікації, тобто до повного переліку вимог до функціонування системи (програмного продукту), яку розроблятимуть. Специфікація складається із користувацьких сценаріїв (use cases), які описують сценарії взаємодії клієнта і ІТ-продукту.

Після проведення перевірки з'ясовується інформація про якість створеного програмного продукту.

Розглянемо основні цілі тестування, це:

- підвищити ймовірність того, що розроблений веб-додаток, буде працювати правильно при будь-яких обставинах;
- підвищити ймовірність того, що веб-додаток, буде відповідати всім описаним вимогам;
- надання актуальної інформації про стан веб-продукту на момент перевірки.

Верифікація та валідація використовуються для перевірки того факту, що програмне забезпечення насправді володіє очікуваними властивостями.

Верифікація - це процес оцінки того, наскільки веб-додаток за підсумками деякого етапу його розробки відповідає умовам (цілі, задачі і терміни), заданим на початку етапу.

Валідація - процес оцінки того, наскільки веб-додаток відповідає вимогам по його призначенню.

«Баг» (з англ. «Bug») - це збій програми, зависання, видання програмних та системних помилок, або дрібні неточності і текстові помилки у веб-додатку.

Етапи тестування:

1) Аналіз програмного продукту.
2) Опрацювання вимог до веб-додатку.
3) Планування стратегії тестування та процесу перевірки якості.
4) Виготовлення тестової документації.
5) Перевірка прототипу.
6) Основне тестування.

7) Стабілізація.

8) Експлуатація.

Тест-план (Test Plan) – документація, що описує повний комплекс робіт по тестуванню.

У тест-плані повинно бути зазначено: що потрібно тестувати; що тестуватиметься; як тестуватиметься; коли тестуватиметься; які критерії початку тестування; які критерії закінчення тестування.

В стандарті IEEE 829 перелічені пункти, з яких має (або може) складатись тест-план:

- ідентифікатор плану випробувань;
- введення;
- тестові завдання;
- характеристики для тестування;
- ознаки для не тестування;
- підхід;
- пункт критерій проходження/провалу;
- критерії тимчасової зупинки і вимоги відновлення;
- тест очікуваних результатів;
- завдання тестування;
- потреби навколишнього середовища;
- відповідальності;
- потреби у навчанні;
- розклад;
- ризики і непередбачені обставини;
- допуски. [9]

Тест дизайн (Test Design) – це етап процесу перевірки IT-продукту, на якому проектуються та створюються тестові сценарії (test cases), у відповідності до визначених критеріїв якості та цілей тестування.

1.3.2 Рівні тестування програмного забезпечення

В системі тестування програмного забезпечення виділяють наступні рівні.

Модульне (компонентне) тестування (Unit Testing) – перевіряє функціональність і виявляє недоліки в окремих, доступних частинах ПЗ.

Інтеграційне тестування (Integration Testing) – перевіряє взаємодія між модулями (компонентами) системи після проведення модульного (компонентного) тестування.

Системне тестування (System Testing) - перевіряє функціональні та не функціональні вимоги до ПЗ в цілому. Виявляються дефекти можливого неправильного застосування програмних ресурсів, непередбачені комбінації даних, несумісність інформації, неочікувані сценарії застосування, незручність використання і т.ін.

Операційне тестування (Release Testing) заходи із з'ясування, що ПЗ влаштовує користувача і виконує свої задачі, які були визначені в системі. Очевидно, що знаходження дефектів на стадії впровадження – критична і коштовна проблема. Тому на самому початку розробки веб-додатків важливо здійснення як верифікації, так і валідації.

Приймальне тестування (Acceptance Testing) – формальний процес перевірки, під час якого з'ясовується чи задовольняє ПЗ приймальним критеріям та виноситься рішення про прийняття веб-додатку.

1.4 Аналіз та визначення недоліків технологій і систем тестування веб-додатків

Ручне тестування додатків – це трудомісткий і тривалий процес. Без поєднання з автоматизованими тестами, такий вид перевірок може застосовуватись тільки на невеликих, короткотривалих проектах. Найбільший з недоліків ручного тестування - людський фактор. Він, звичайно, впливає на все, що відбувається в команді - і на роботу учасників, і на події, що відбуваються на стороні клієнта. Частина помилок продукту може бути пропущена, а деякі

результати перевірки можуть виявитися суб'єктивними. У разі QA, причиною того, що тестувальник буде недостатньо занурений в процес і пропустить потенційну помилку, може бути брак досвіду, сімейні проблеми або головний біль.

Один і той же сценарій два тестувальника можуть перевірити різними способами. Важливо, щоб кожен непередбачений або той, що відрізняється від очікуваного результат, був зафіксований у вигляді кейса. Це дозволить будь-якому тестувальнику перевірити його, зробивши той самий набір дій. Але і цього може бути мало - якщо кейс виявиться недостатньо докладним, а тестувальник – не розбереться в описі. Гарантувати стовідсоткове виключення людського фактору, звичайно, неможливо – але можна намагатися максимально знизити ймовірність проблем, які він викликає.

Це може негативно позначається на термінах і трудовитратах, адже до періодичних провідних базових кейсів і регресії додаються і кейси, створенні тестувальниками в процесі.

Погіршує ситуацію з перевіркою ймовірність того, що частина зустрінутих багів, ще не матиме суворого опису бо причини їх виникнення поки не зрозумілі. Тому, автоматизація проходження проблемних кейсів – у випадку переходу до програмних тестів, буде цілком виправдана як з точки зору часу і фінансів.

У будь-якому випадку - поява перших кейсів, які потребують регресивних тестів або реліз другої версії додатка і, відповідно цим подіям масштабування команди – той момент, коли автоматизація стає актуальною. На цьому етапі автоматизація вже почне економити час: розробник зможе сам, ще до передачі QA-відділу, запустити регрес-тести нової функції, щоб переконатися, що вона не руйнує існуючий функціонал, а тестувальникам не доведеться зайвий раз проходити по базовим кейсам. Ще одна перевага впровадження автоматизованих тестів на цьому етапі – їх можна запускати за певним розкладом: щоночі; в дні закінчення спринтів; при публікації нових збірок додатків.

Враховуючи вищезазначене, можна виділити такі основні недоліки систем тестування веб-додатків:

- людський фактор – частина помилок продукту може бути пропущена, а деякі результати перевірки можуть виявитися суб'єктивними;
- трудовитрати і тривалість – мануальне тестування не дозволяє протестувати веб-додатки швидко;
- відсутність можливості моделювання великого навантаження – при ручному тестуванні неможливо змоделювати велику кількість користувачів.

1.5 Постановка задачі дослідження

В даній роботі ми проведемо дослідження ефективності існуючих систем та методів тестування за наступними критеріями оцінки:

- час (швидкість) тестування;
- повнота тестування;
- повторюваність, неодноразове використання;
- навантаження;
- надійність.

1.6 Висновки до розділу 1

Під час проведення дослідження були розглянуті поняття програмного забезпечення та веб-додатків, їх види та класифікація, структура, технології створення. Опрацьовані моделі розробки програмного забезпечення.

Також були дослідженні сучасні засоби створення веб-додатків. Окремо розглянуті платформи, що базуються на серверних технологіях. Акцентовано увагу на питанні перевірки надійності та якості програмних продуктів. Також було розглянуте питання тестування, його види, етапи і цілі. Основні методи тестування програмних засобів, для підвищення якості процесу перевірки програмного забезпечення. В ході аналізу виникла потреба у проектуванні цілої системи автоматизованого тестування веб-додатків та її програмної реалізації.

					КР 122.6157м.ПЗ Р2			
Зм.	Лист	№ докум	Підпис	Дата				
Студент		Потужний І.Р.			Методи та засоби тестування веб-додатків	Літера	Лист	Листів
Керівник		Гайда А.Ю.					31	9
Консульт.		Дудченко О.М.				ХФ НУК ім. адм. Макарова		
Рецензент		Новиков Ю.М.						
Зав. Каф		Гучек П.Й.						

2 МЕТОДИ ТА ЗАСОБИ ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ

2.1 Системи управління конфігурацією програмними продуктами та визначення критеріїв оцінки методів та засобів тестування веб-додатків

Конфігураційне керування (від англ. Software Configuration Management, SCM) – сукупність заходів, які здійснюються розробниками під час створення програмного забезпечення, з метою збереження цілісності системи при змінах програмного продукту, обліку та формалізації процесу внесення змін, запобігання збоїв у роботі системи [14].

На теперішній час, для здійснення заходів із конфігураційного керування використовуються сервери безперервної інтеграції програмного продукту. Розглянемо найбільш поширені з них.

Jenkins (Дженкінс) – система з відкритим вихідним кодом, тобто продукт доступний для перегляду, вивчення та зміни, створений на базі Java. Дженкінс дозволяє автоматизувати частину процесу розробки програмного забезпечення, без участі людини. Дана система призначена для забезпечення процесу безперервної інтеграції програмного забезпечення.

BuildBot (Continuous Integration FrameworkBuildbot) – це «фреймворк» з відкритим вихідним кодом, який використовується для автоматизації процесів збирання, тестування та випуску програмного забезпечення. Він реалізований в основному на Python і дозволяє автоматизувати всі аспекти циклу розробки програмного продукту. З використанням BuildBot реалізується система, яка відповідає робочому процесу і розвивається разом з організацією. Цей «фреймворк» створений для масштабованості, і завдяки широкому спектру функцій може вирости до великих установок.

CruiseControl – це система безперервного обміну програмного забезпечення реалізована на платформі Java, з відкритим вихідним кодом, яка поширюється під BSD ліцензією -style. Вона включає в себе «плагіни» для повідомлень електронною поштою, Ant, різні інструменти управління версіями,

але не обмежується ними. З використанням веб-інтерфейсу можливий перегляд даних про поточну і попередню збірки. Це робить можливим тривалу інтеграцію будь-якого процесу розробки програмного забезпечення. CruiseControl – була однією з перших програм такого типу, спочатку створена співробітниками ThoughtWorks для забезпечення безперервної інтеграції в проект. Пізніше, CruiseControl була виділена в окремий додаток.

GoCD – сервер, який відмінно справляється з моделюванням складних робочих процесів завдяки використанню швидкого зворотного зв'язку з його конструкціями моделювання, паралельним виконанням і управлінням залежностями. Він має простий інтерфейс та допомагає усувати несправності конвеєра, відстежуючи кожну зміну від фіксації до розгортання в режимі реального часу.

Strider – це платформа безперервного розгортання та інтеграції з відкритим вихідним кодом. Вона реалізована на Node.JS/JavaScript і використовує MongoDB в якості резервного сховища. Вона видається під ліцензією BSD. Конфігурація відбувається у змінних середовищах. Більшість значень за замовчуванням повинні задовільно працювати на локальному хості, однак для розгортання, доступного через Інтернет може бути потрібна мануальна правка.

TeamCity – CI/CD-рішення загального призначення реалізоване на Java. Воно надає гнучкість у взаємодії з різними робочими процесами і методами розробки.

Travis CI – безкоштовна для «опенсорс»-проектів CI-система, яка легко налаштовується, стабільно працює і тому дуже популярна в проектах з відкритим вихідним кодом. Не треба плутати travis-ci.org і travis-ci.com. Перший - безкоштовний сервіс для «опенсорсних» проектів, другий - платна CI-система. Налаштування Travis CI в перший раз може здатися складним, але в подальшому буду значно простіше.

У таблиці 2.1 здійснено порівняння вищезазначених серверів безперервної інтеграції на основі обчислювальної платформи, побудови та інструментів інтеграції та підтримки різних середовищ інтеграції.

Таблиця 2.1 – Таблиця порівняння серверів інтеграції

Назва	Платформа	Побудова, Windows	Побудова, Java	Побудова, інші	Інтеграція, середовище розробки	Інтеграція, інші
Jenkins	Web container	MSBuild, NAnt	Ant, Maven 2, Kundo	Cmake, Gant, Gradle, Grails, Phing, Rake, Ruby, SCons, Python, shell script, command-line	Eclipse, IntelliJ IDEA, NetBeans	Bugzilla, Google Code, Jira, Bitbucket, Redmine, FindBugs, Checkstyle, PMD and Mantis, Trac, HP ALM
BuildBot	Python	Command line	Command line	Command-line	-	-
Cruise Control	Cross platform	NAnt, Rake, Xcode	Phing, Apache Ant, Maven	catch-all 'exec'	Eclipse	-
GoCD	Cross platform	Command line	Command line	Command-line	Hi	GitHub
Strider	Node.js	Hi	Hi	C, C++, Clojure, Erlang, Go, Groovy, Haskell, Java, Node.js, Perl, PHP, Python, Ruby, Scala	Hi	GitHub, Bitbucket, Heroku, GitHub Enterprise, Git
Team City	Web container	MSBuild, NAnt, Visual Studio, Duplicates finder for .NET	Ant, Maven 2-3, Gradle, IntelliJ IDEA .ipr based and Inspections and Duplicates finder	Rake, FxCop, command-line	Eclipse, Visual Studio, IntelliJ IDEA, RubyMine, PyCharm, PhpStorm, WebStorm	Jetbrains Youtrack, Jira, Bugzilla, FishEye, FindBugs, PMD, dotCover, NCover
Travis CI	Hosted	Hi	Ant, Maven, Gradle	C, C++, Clojure, Elixir, Erlang, Go, Groovy, Haskell, Java, Node.js, Perl, PHP, Python, Ruby, Rust, Smalltalk	Hi	GitHub, Heroku

Підтримка найбільш поширених SCM систем вищезазначеними серверами інтеграції на основі системи керування версіями, наведена у таблиці 2.2.

Таблиця 2.2 – Підтримка SCM систем серверами інтеграцій [17]

Назва	Darcs	Git	GNU Bazaar	Integrity	Mercurial	Perforce	PVCS	Subversion
Jenkins	Так	Так	Так	Так	Так	Так	Так	Так
BuildBot	Так	Так	Так	Ні	Так	Так	Ні	Так
Cruise Control	Так	Так	Ні	Так	Так	Так	Ні	Так
GoCD	Ні	Так	Ні	Ні	Так	Так	Ні	Так
Strider	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні
Team City	Ні	Так	Ні	Ні	Так	Так	Ні	Так
Travis CI	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні

2.2 Засоби модульного тестування веб-додатків

Модульні тести дозволяють розробникам і тестувальникам швидко знаходити логічні помилки у ПЗ. Сутність модульного тестування складається у з'ясуванні питання: чи працює код блоку (модуля) так, як було замислено або очікувалося? При роботі з додатками великого обсягу, модульне тестування дозволяє реально економити час, допомагає відстежувати проблеми без риску оновлення програмного коду. Розглянемо найбільш поширені засоби модульного тестування.

JUnit - це інфраструктура модульного тестування для мови програмування Java, з відкритим вихідним кодом, що використовується для написання та запуску тестів. Вона відіграє важливу роль при розробці ПЗ, дозволяє перевірити окремий блок (модуль) на коректність роботи. JUnit являє собою сімейство платформ модульного тестування, відомих під загальною назвою xUnit. Тести

JUnit можливо автоматизувати і, як результат, вони дають миттєвий зворотній зв'язок.

NUnit – відкрита платформа юніт-тестування для .NET, що дозволяє створювати автоматизовані тести. Бібліотека NUnit перенесена з JUnit. Чим вище якість програми, тим менше коштів витрачається на усунення недоліків проекту.

YUI Test – незалежний модуль для тестування блоків (модулів) програмного продукту, який є компонентом бібліотеки YUI (Yahoo!). Він є інтегрованим середовищем для всебічного і повного модульного тестування.

TestNG- це фреймворк, який вмістив у собі багато чого з JUnit та NUnit. У ньому впровадженні сучасні можливості, що роблять його ефективним і простим інструментом. TestNG дозволяє здійснювати багатопоточне тестування, робити анотації, застосовувати XML для конфігурування тестів та підтримується Eclipse, Hudson, Ant, IDEA, Netbean.

PHPUnit – це фреймворк модульного тестування, який застосовується під час розробки програмного забезпечення на PHP. Він є різновидом сімейства фреймворків XUnit на основі пакету SUnit.

Виконаємо порівняльний аналіз засобів модульного тестування за швидкодією. Обчислимо функцію факторіалу для вищезазначених інструментів блочного тестування.

Зобразимо порівняльний графік виконання вказаної функції в залежності від часу (рисунок 2.1).

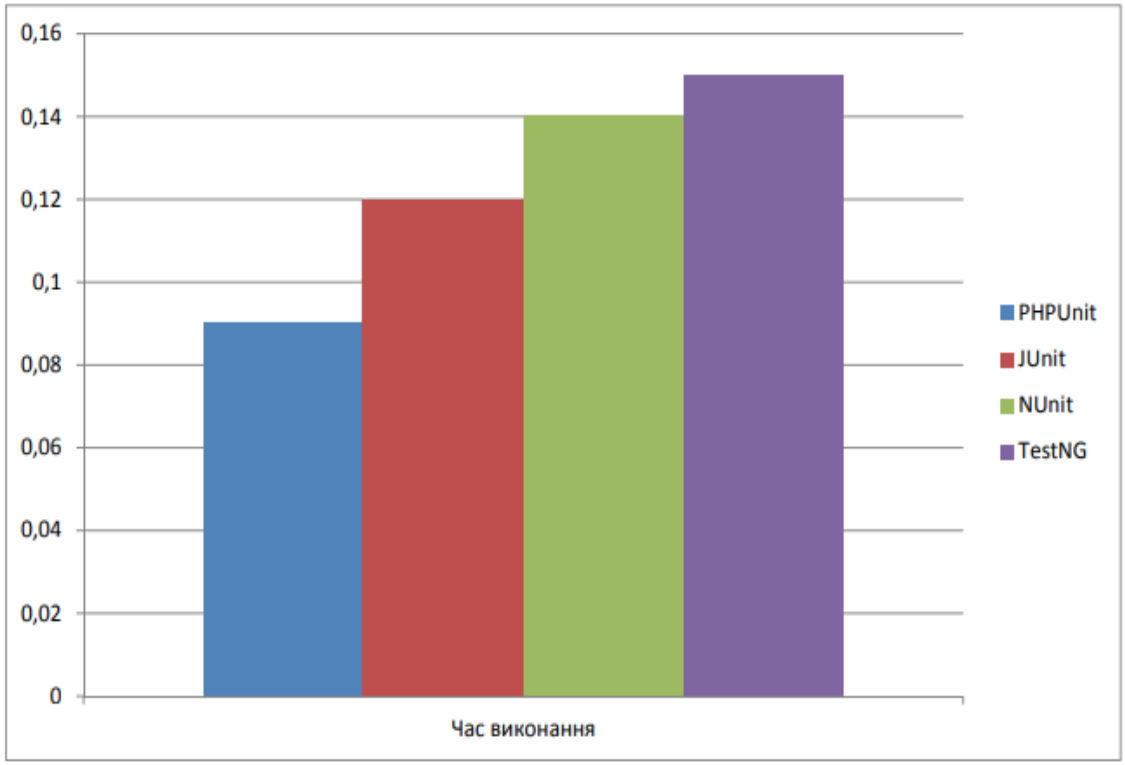

Рисунок 2.1 – Швидкодія виконання функції обчислення факторіалу інструментами модульного (блокового) тестування

Як з'ясувалося в результаті дослідження - PHPUnit є найшвидшим засобом. Тому, при написанні модульних тестів будемо приміняти PHPUnit.

2.3 Засоби тестування прикладного програмного інтерфейсу веб-додатку

Прикладний програмний інтерфейс (від англ. Application Programming Interface, API) – це набір підпрограм, протоколів і засобів створення ПЗ, що дає можливість будь-яким програмним компонентам взаємодіяти один з одним.

Система програмного забезпечення, що виконує API, вміщує у собі функції, що виконуються за допомогою іншої системи програмного забезпечення. Перевірка API цілком різниться від перевірки графічного інтерфейсу.

Це тестування не буде базуватися на зовнішньому вигляді додатків.

Зазвичай API використовується у випадках, коли:

- інтегрований в середовище програмування, наприклад CC++ або Java API;
- у спеціальному значенні, наприклад Google Maps API или Java API XML для веб-послуг. За допомогою Google Maps API можна використати послугу із зазначенням місцезнаходження на карті через інтерфейс, що надається Google;
- API операційної системи, це інтерфейс, під впливом якого додатки отримують доступ до послуг операційної системи. Наприклад, Windows API, має доступну для додатків процедуру для кожної служби операційної системи.

При використанні API при розробці веб-додатків, як правило, прикладний програмний інтерфейс визначається набором повідомлень запиту HTTP, також визначається структура повідомлень-відповідей, зазвичай у розширенні мови розмітки XML або в форматі об'єктного запису JavaScript (JSON).

Порівняльний аналіз тестування API і модульного тестування наданий у таблиці 2.3 [40].

Таблиця 2.3 – Таблиця-порівняння модульного тестування та тестування API

Модульне тестування	Тестування API
Зазвичай виконують розробники	Зазвичай виконують тестувальники
Тестуються окремі функції	Тестується вся функціональність системи
Розробник може звертатися до вихідного коду	Тестувальник не може звертатися до вихідного коду
Включене тестування графічного інтерфейсу	Тестуються лише API функції
Тестуються лише базові функції	Тестуються усі функціональні питання
Обмежене застосування	Широка сфера застосування
Зазвичай проходить до початку роботи	Проходить після створення побудови

Так як, модульне тестування та API однаково спрямовані на вихідний програмний код, то для них може використовуватися схожий функціонал тестування, наприклад:

- SOAPUI;
- PHPUnit+Curl;
- Runscope;
- dotTEST;
- CTESK;
- Postman with Newman;
- Postman with jetpacks;
- Cfix;
- Check;
- Eclipse SDK tool.

2.4 Висновки до розділу 2

Перевірка програмного забезпечення має бути не лише корисною, але й максимально ефективною та раціональною. На теперішній час практично у кожному ІТ-проекті намагаються досягнути максимальної якості. Для вирішення цієї задачі розробляються автоматизовані системи тестування.

Були досліджені інструменти автоматичного тестування веб-сайтів. А саме, програмне забезпечення безперервної ітерації, засоби модульного тестування веб-додатків та прикладні програмні інтерфейси. Порівняно їх ефективність, швидкість проходження перевірки та інші характеристики на прикладі роботи з конкретними даними в наявній системі.

Здійснивши аналіз отриманих даних, зроблено висновки про ефективність, переваги та недоліки методів, розглянуто перспективи і можливості їх застосування. Обрано конкретні інструменти і сервіси, які використовуються для автоматизованої системи тестування веб-додатків.

					КР 122.6157м.ПЗ Р3		
Зм.	Лист	№ докум	Підпис	Дата			
Студент		Потужний І.Р.			*Програмна реалізація системи автоматизації тестування веб-додатку*	Літера	Лист 40 / Листов 19
Керівник		Гайда А.Ю.					
Конс-т		Латанська Л.О.					
Рецинзент		Новиков Ю.М.				**ХФ НУК ім. адм. Макарова**	
Зав. Каф		Гучек П.Й.					

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ ВЕБ-ДОДАТКУ

3.1 НАЛАШТУВАННЯ СИСТЕМИ ПОСТІЙНОЇ ІНТЕГРАЦІЇ ПРОЕКТУ

Систему автоматизованого тестування реалізуємо на базі реально діючого сервісу порівняння цін, вибору товарів та пропозицій Інтернет-магазинів Izzi (Price Compare).

Порівняння цін на товари в Інтернеті зробило величезну споживчу революцію та призвело до різкого зниження їх цін. Використання Інтернет-ресурсу порівняння цін, дозволило споживачам та постачальникам суттєво спростити процес купівлі та продажу, зробити його контрольованим та об'єктивним. Навіть малі підприємства, які зазвичай мають труднощі з маркетингом, мають можливість отримати вигоду від широкого кола клієнтів та збільшити обсяг своїх продажів.

Сьогодні Інтернет-магазини продають величезну різноманітність товарів за різними цінами та різною якістю. Один має виріб у різноманітних кольорах, а інший у різних розмірах, тому важко пробитися між усіма цінами. Сервіс Izzi пропонує просте, сучасне та всебічне порівняння цін, щодня скануючи товари, запропоновані різними магазинами в самих різних галузях, відстеження цін, з метою знайти найактуальніший товар.

Веб-сайт Izzi був створений з метою порівняння цін у всьому світі від широкого кола постачальників. Не виходячи з дому і не витрачаючи час на пошуки та порівняння, для власників малого та великого бізнесу: Сайт допомагає постачальникам по всьому світу збільшити продажі та залучити нових клієнтів. Сприяння збуту в Інтернеті - це широке поле, яке включає різноманітні методи, які можна застосувати в галузі інтернет-маркетингу з метою збільшення потенційних клієнтів і, зрештою, продажу бізнесу через інтернет-мережу.

Обслуговування клієнтів є найважливішим елементом бізнесу в інтернеті.

Інтернет давно є вагомою частиною дня для кожного з нас. Він використовується вдома, в офісі та протягом дня через мобільний пристрій, який давно адаптований не лише для здійснення дзвінків. Для власників бізнесу та брендів це справжня революція, яка також вимагає рекламних посилань. Онлайн-просування доступне для будь-якого бізнесу, великого та малого, і не вимагає великих бюджетних вкладень, але правильних та заощадливих.

Якщо раніше великі та малі підприємства інвестували в друковані ЗМІ, радіо та телебачення, то сьогодні цього недостатньо, і це не завжди можливо. Власники великого бізнесу розуміють, що Інтернет може залучити більше клієнтів, спрямовувати громадську думку та генерувати продажі. Саме те, що корисно для великого бізнесу, є доречним і доступним для малого бізнесу, який може використовувати менший бюджет для значного зростання продажів.

Залучення цільової аудиторії в Інтернеті, за допомогою соціальних мереж та правильних налаштувань, дає можливість впливати на обсяг продажів. Інтернет-просування відрізняється від старого способу реклами, коли бізнес повинен був придбати рекламний пакет. Інтернет дозволяє гнучко керувати кампаніями, вивчати результати за короткий час та контролювати свій бюджет. Якщо ви вирішили продавати через Інтернет, наступним кроком буде побудова бізнес-моделі для вашого магазину. Це не повинна бути особливо складна модель, а насправді вона повинна бути навіть простою та короткою.

Веб-сервіс Izzi написаний на мові програмування PHP. Для об’єднання різних видів тестування, безперервного автоматичного тестування та безперервної розгортки системи, потрібно налаштувати сервер, який буде своєчасно інтегрувати тести. Для вирішення цієї задачі скористаємося платформою Jenkins. Це програмна система з відкритим вихідним кодом, яка дозволяє нам зробити процес безперервної інтеграції. Завдяки Jenkins отримаємо цілісну систему автоматизованого тестування нашого проекту.

Після успішної інсталяції, стартова сторінка в Jenkins виглядатиме як зображено на рисунку 3.1.

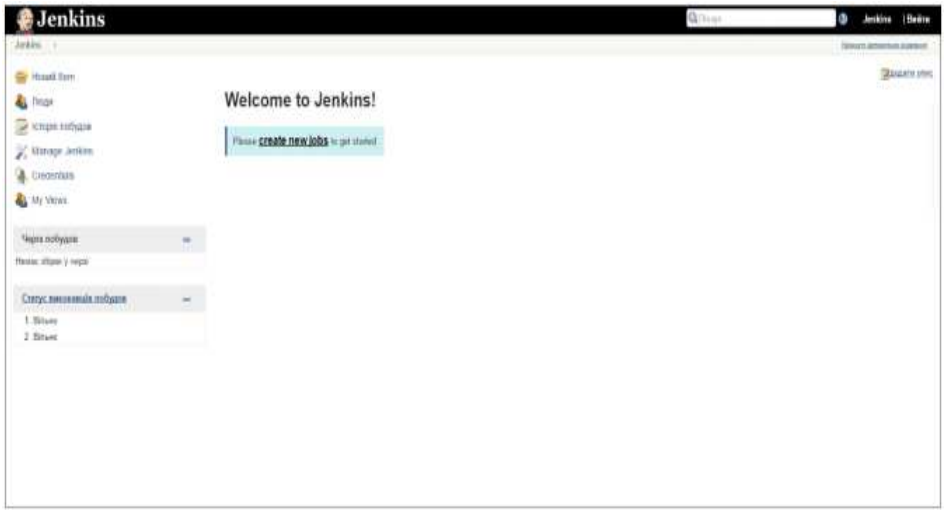

Рисунок 3.1 – Стартова сторінка налаштувань Jenkins після інсталяції

Наступним кроком буде створення проекту. Переходимо до меню та вибираємо пункт «Новий Item». Далі, потрібно ввести назву нашого проекту (izzi.co.il_test) та вибрати його тип, який в нашому випадку – «freestyle project» (рисунок 3.2). Призначення проекту izzi.co.il_test – для unit тестів.

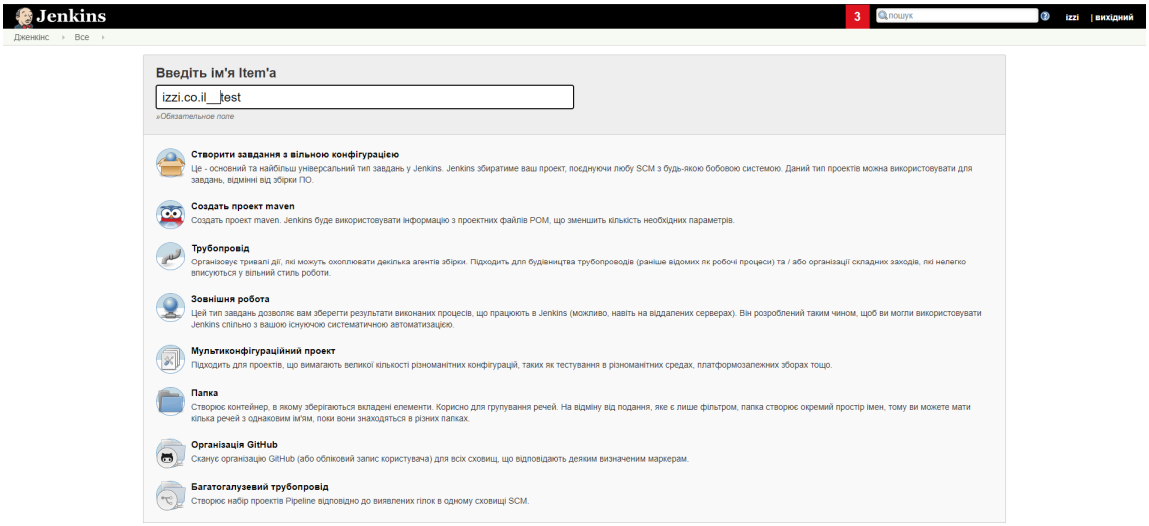

Рисунок 3.2 – Вікно створення проекту

Відкриваючи сторінку налаштування проекту, можна побачити таку інформацію:

- Project name: назва проекту;
- Description: опис задачі;
- Strategy: алгоритм ведення журналу;

- Parameterized: формулювання перемінливих проекту. Є такі типи: параметр тексту, параметр файлу, параметр рядка і т.д.;
- Where: лімітує простір реалізації проекту;
- Advance configuration: додаткова комплектація для керування засобом збірки проекту.

Вибрані для інсталяції плагіни, зумовлюють категорії і функції, які будуть використані у нашому проекті. Перелічимо деякі з них:

- Source code management: функція керування вихідним кодом – Git;
- Build triggers: метод запускання збірки – стороньо;
- Build: збірка – найголовніший елемент проекту. Будемо спиратися на команди DOS для Windows або Shell для Linux;

- Post-build actions: дії, які здійсняються по завершенню збірки. Зазвичай – це інформування на електронну адресу, активація різних збірок та друкування зведень про результати.

Проект розташований на сторінці:

http://185.132.133.198:9889/job/izzi.co.il_test/.

У Jenkins є змога запускати збірку як автоматично, так і власноруч. Вибравши автоматичну збірку, буде доступна установка визначення параметру Build Triggers, при налаштуванні конфігурацій проекту (рисунок 3.3). Припустимі такі випадки:

- у разі, якщо проект пов’язаний з другими проектами, то після його налаштування слід зробити висновок, чи є необхідність після цього проекту збирати інші;
- віддалений старт збірки проекту, якщо збірка починає виконання з другої системи або з другого вузла;
- періодична збірка проекту, шляхом складання календарю його збірки згідно з налаштуваннями;

- Jenkins періодично надсилає запити до системи управління версіями (запити до SCM), та у разі виявлення змін програмного коду, реалізується перезбірка проекту.

Рисунок 3.3 – Конфігурації старту збірки

Безперервне розгортання має за ціль автоматизацію тестування інкрементальних змін програмного забезпечення і оперативне розгортання оновлених версій на робочій платформі. При роботі в такому режимі, зміни, що вносяться програмістами, можуть потрапляти до клієнтів за лічені дні або навіть години.

Безперервне розгортання дозволяє розробникам оперативно вводити нові функції та усувати дефекти, і хоча воно не гарантує, що дефект буде знайдений миттєво, в разі його більш пізнього виявлення ціна зміни залишається колишньою. Якщо ж дефект не виявляють протягом довгого часу, то, найімовірніше, відповідною функцією мало користуються і її взагалі можна прибрати.

Щоб реалізувати задачу безперервної розгортки, необхідно створити своєчасний механізм впровадження наборів змін. Розгортання можна поділити на декілька етапів:

- зміна програмного коду;
- запуск збірки засобами керування вихідного програмного коду;
- виконання автоматизованої перевірки;
- інсталяція збірки.

Алгоритм системи безперервного розгортання полягає у відмінності від стандартного ходу розгортки. Так, програміст відзначає зміни на сервері управління вихідним програмним кодом, а потім другий сервер збірки реалізує збірку. Під час виконання безперервної розгортки з'являється ведуча машина Jenkins. Для стягнення вихідного програмного коду, ведуча машина Jenkins застосовує Git, після чого починає старт збірки. Середовища розробки, перевірки та створення використовуються в якості ведених машин Jenkins. Керує ними ведуча машина Jenkins, а вони реалізують проекти за календарем.

Алгоритм безперервного розгортання зображений на рисунку 3.4.

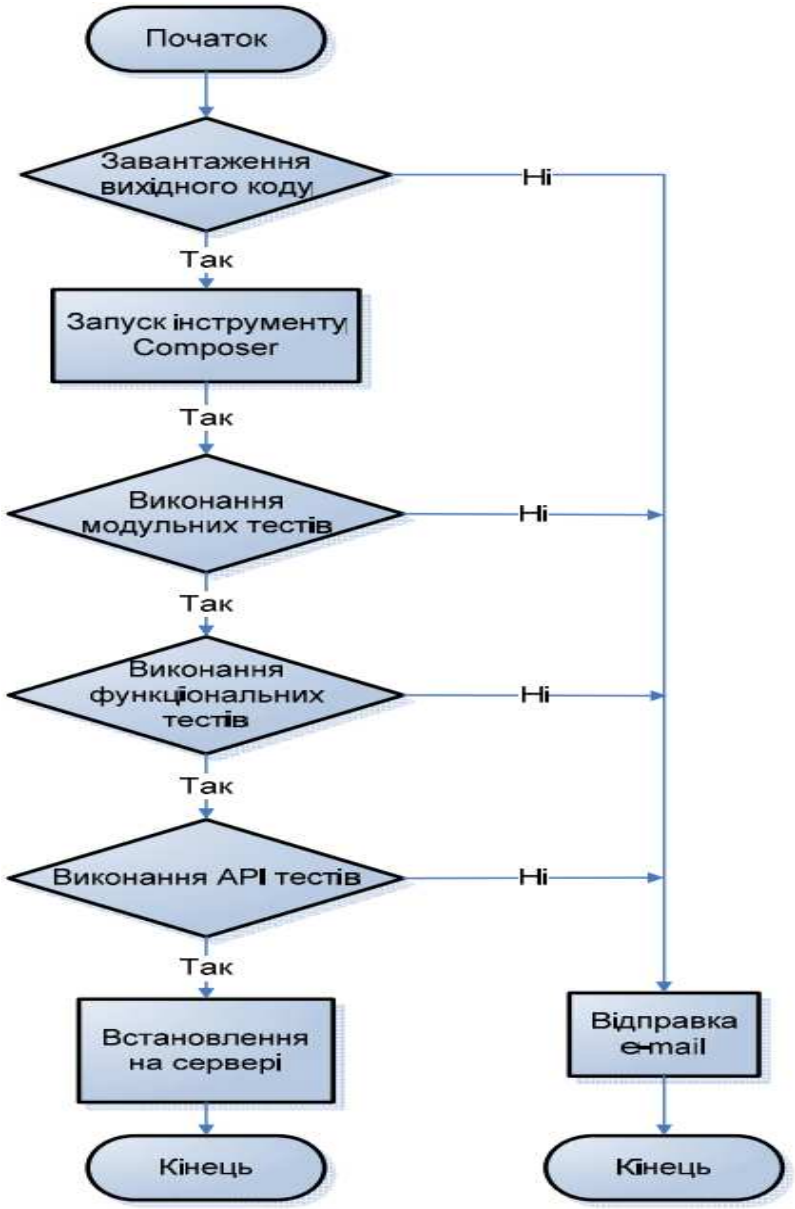

Рисунок 3.4 – Алгоритм побудування автоматизованого проекту

Розробка проекту тестування здійснюється наступним чином:

1) Виконаємо команду «git checkout», щоб отримати код усього проекту. Ми використовуємо конфігурації керування командним кодом Git.
2) Скористуємося інструментом «Composer» для спрощення підключення важливих для тестування бібліотек. Потрібно описати бібліотеки, які залежні від коду. Потім, використаємо команду «Composer install», для того, щоб автоматично знайти необхідні версії бібліотек для нашого проекту та завантажити і встановити у папку проекту.
3) Виконаємо модульні тести – «PHPUnit».
4) Виконаємо функціональні тести – «Selenium».
5) Протестуємо API-функції системи.
6) Створення zip-архіву від Jenkins при проходженні тестів та успішній побудові.
7) Збірка розгортається та встановлюється на сервері.

Якщо проходження тестів було неуспішним, то припиняється побудова та з'являється помітка невдалої збірки. Після чого, система сповіщає про невдалу побудову на поштову скриньку.

На рисунку 3.5 проілюстрована сторінка успішних побудов проекту в Jenkins.

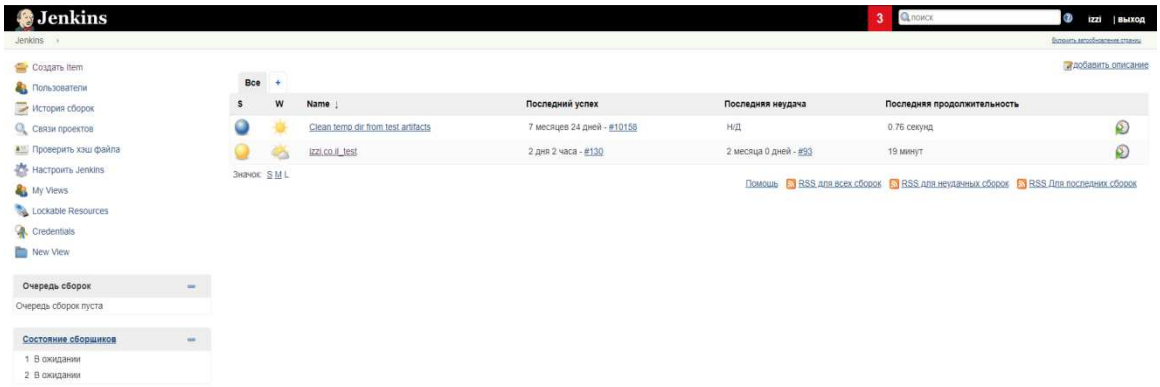

Рисунок 3.5 – Головна сторінка проекту izzi.co.il_test в Jenkins

3.2 Програмна реалізація функціональних тестів веб-додатку

Для початку роботи з Selenium слід приєднати потрібні бібліотеки. Після чого можна зайнятися написанням тестів.

Для повідомлення транслятору, у якому саме «пакеті» визначати класи, котрі знаходяться у потрібному файлі, скористуємося оператором «package». Пакети – це важливий механізм поділу класів, саме тому всі класи Java знаходяться в них. Завдяки пакетам задається набір відокремлених просторів назв, у яких збережені імена класів.

Вкажемо потрібний шлях:

package com.qa.izzi.ui;

Також додамо списки операторів «import».

WebDriver – популярний інструмент для управління реальним браузером, який можна використовувати як для автоматизації тестування веб-додатків, так і для виконання інших рутинних завдань, пов'язаних з роботою в Інтернеті. Управління відбувається як локально, так і віддалено і найближче імітує дії користувача. Крім того, «WebDriver» - проект з відкритим вихідним кодом, підтримує безліч мов програмування і має велике співтовариство користувачів.

Ми будемо використовувати інтернет-браузер Google Chrome. Тому саме Chrome Driver забезпечить нам роботу.

Наступним кроком підключимо потрібний клас:

```
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.*;
package TestCases; - місце розташування нашого тесту
import Utilities.ReadProperties;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
```

Проведемо функціональне тестування тест-кейсу авторизації в систему, так як без неї (або у разі негативної поведінки), нема можливості працювати з системою.

Веб-сторінка логіну клієнт-юзера проілюстрована на рисунку 3.6. , де пронумеровано головні елементи сторінки, які варто протестувати:

1 – поле вводу логіну (у нашому випадку е-майлу);

2 – поле вводу особистого паролю користувача;

3 – «кнопка підтвердження» для входу.

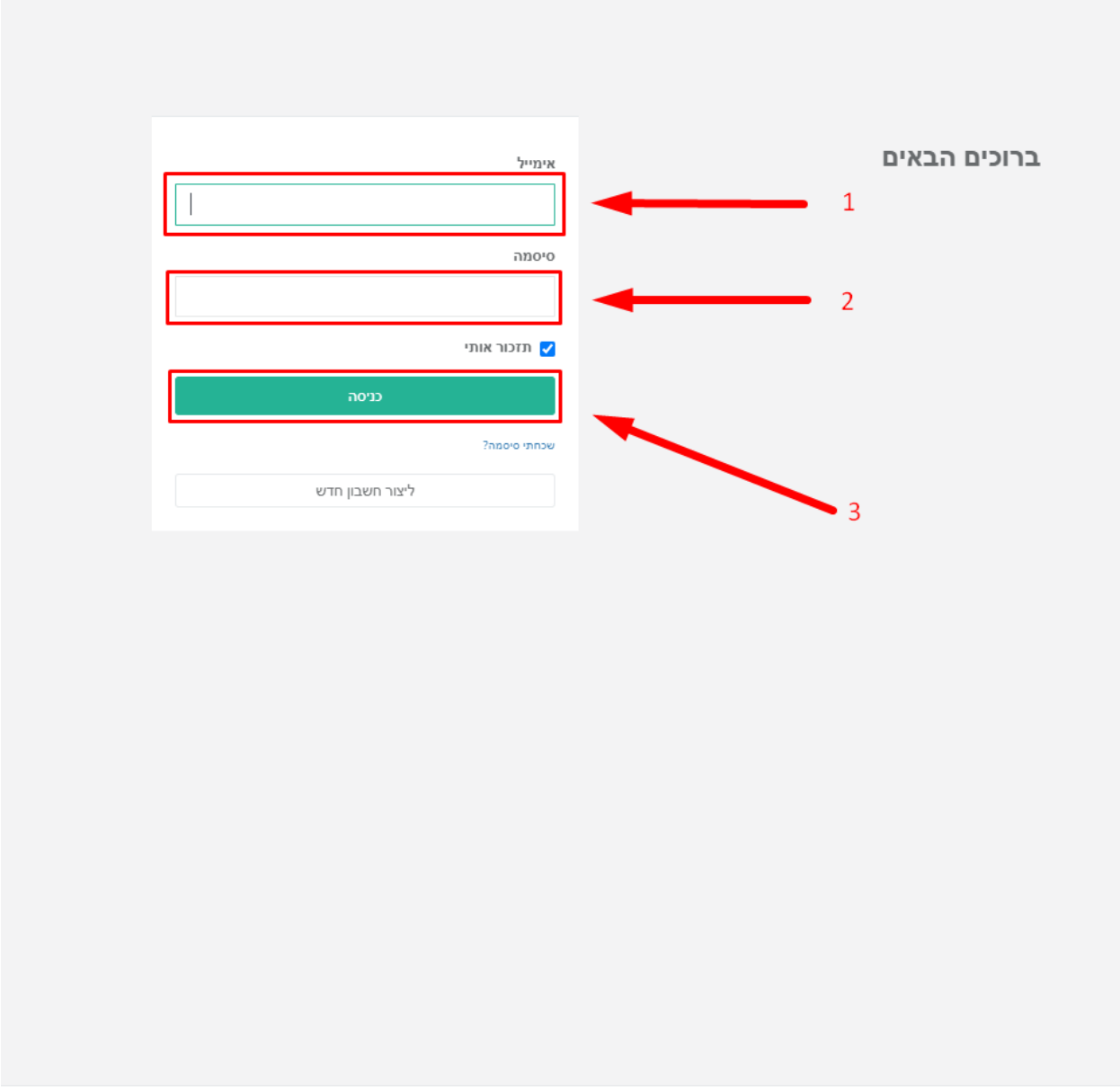

Рисунок 3.6 – Веб-сторінка логіну клієнт-юзера до власного кабінету Izzi

ADVLogin.java – тест-кейс, який відповідальний за тестування правильності роботи сторінки.

Для засвідчення у файлі тестового методу, проанотуємо його як – «@Test», та запустимо його в окремий потік для реалізації тестування.

Анотації виступають «дескрипторами», які включають в програму та користуються для збереження метаданих програмного коду, потрібних на різноманітних стадіях життєвого циклу ПО.

Структура написаного нами тест-кейсу на мові програмування "Java" має вигляд, який відображено на рисунку 3.7.

Рисунок 3.7 – Конструкція коду тест-кейсу ADVLogin.java

Анотація @Before зазначає методи, що викликаються до створення екземпляра тестового класу. Розміщує предустановки для тесту в таких випадках: клас розміщає декілька тестів та користуються різними конфігураціями; декілька тестів користуються одними і тими же даними, з метою економії часу на їх створення для кожного тесту.

Виконуватися даний метод буде перед тестовим методом @Test.

Тому, в блоці методу @Before створюємо шаблонний метод, у якому вказуємо шлях до драйверу chromedriver, та налаштування відкритого вікна браузера.

```
@Before
    public void setup() {
    System.setProperty("webdriver.chrome.driver","src/tes
t/chromedriver.exe");ChromeOptions options = new
ChromeOptions();
    options.addArguments("disable-infobars");
    driver = new ChromeDriver(options);
    driver.manage().window().maximize();
    }
```

Після цього, напишемо команди емалювання клінт-юзера: вхід на сторінку авторизації адвертайзера на проекті Izzi та безпосередньо авторизація .

В блоці @Test ми будемо писати самі команди емулювання користувача. Використовуючи функцію driver.get() ми перейдемо на потрібну нам сторінку.

Функція авторизації розпочинається із заповнення полів е-мейлу та паролю (рисунок 3.6). У відповідне поле вводимо (інструмент sendKeys () – вводить

потрібний текст у потрібне поле) електронну адресу, яка нам потрібна. Наступним кроком буде поле паролю. Після чого тиснемо на блок кнопки підтвердження використовуючи інструментом "click()", який емалює натискання клавіш клавіатури або комп'ютерної миші юзера на абиякий вказаний нами елемент:

Блок @Test виглядає наступним чином:

```
@Test
Public void test(){
driver.get("https://office.izzi.co.il/user/login");
driver.findElement(By.id("loginform_email")).sendKeys("e
mail");
driver.findElement(By.id("loginform-
password").sendKeys("password");
driver.findElement(By.xpath("//*[@id="login-
form"]/button").click();
```

Визначити точне місцезнаходження елементу на веб-сторінці та атрибути можна за методом " xpath ()". Як наслідок, програма відкриває нам веб-сторінку успішної авторизації, а саме особистий кабінет (рисунок 3.8).

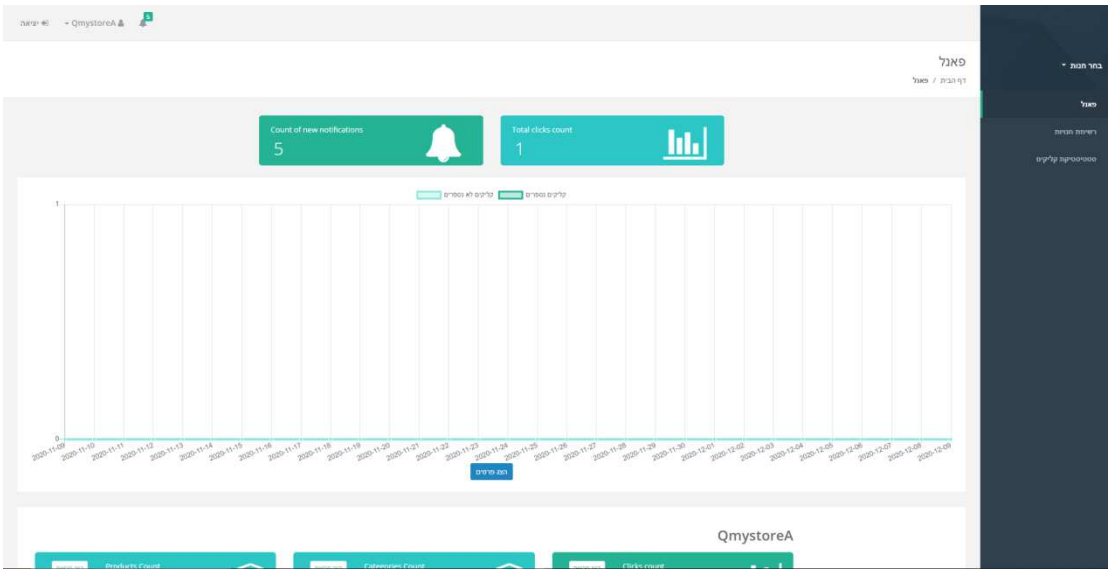

Рисунок 3.8 – Результат здійснення успішної авторизації

Реалізуємо перевірку правильності функціонування авторизації, використовуючи оператори вибору if else та функцію isDisplayed(), умовою якого буде з'ясування присутності вибраного елементу на сторінці.

```
WebElement logOutButton = driver.findElevent (By.id("user-dropdown"));
if (logOutButton.isDisplayed()==true){
System.out.println("Advertiser is logged in");}
else {
System.out.println("Advertiser is not logged in");}
```

Анотація @After свідчить про те, що здійснення методу буде проходить услід за тестовим методом @Test.

По закінченню виконання тестового методу буде виконуватися ще один шаблонний метод – "exit ()", задача якого, завершити роботу системи. У цього методу немає залежності від результату виконання тесту, успішного чи з виявленими помилками.

Тому, у блоці @After пишемо такі команди:

```
@After
public void exit() {
driver.quit();}
```

Після закінчення перевірки, програма закриває інтернет-браузер. На рисунку 3.9 зображений результат виконання тесту у консольному виводі.

```
Starting ChromeDriver 85.0.4183.87 (cd6713ebf92fa1cacc0f1a598df280093af0c5d7-refs/branch-heads/4183@{#1689}) on port 1335
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Oct 13, 2020 2:23:30 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Advertiser is logged in
```

Рисунок 3.9 – Позитивний результат тесту ADVLogin.java

3.3 Програмна реалізація модульного тестування веб-додатку

Будь-який unit тест будемо розміщати у окремого файлі. Так, для файлу User.php існує функція getProfilePercentage (), яка обчислює відсоток повноти анкети клієнт-юзера даними.

Ось програмний код, яким будемо користуватись:

```
public functionis turnAilePercent ($abolute = true) {
$percent = 0;
if ($this->scrin) $percent += 15;
if ($this->name1) $percent += 5;
if ($this->name2) $percent += 5;
if ($this->allname) $percent += 5;
if ($this->quatro) $percent += 5;
if (SettingInterest::brend()->findAllByAttributes(array('user_id' => $this->id)))
{ $percent += 15; }
if ($this->expressAvailableCheck ()) $percent += 25;
if ($this->mail_form) $percent += 25;
if($abolute) {
return $percent;
} else {
$returnPersent = 0;
$range = [25, 50, 75, 100];
foreach ($range as $key=>$val) {
if($key < 3 && $percent >= $range[$key] && $percent < $range[$key+1]) {
$returnPersent = $range[$key];
} elseif($percent == 100) {
$returnPersent = 100; } }
return $returnPersent; } }
```

Функція повертає абсолютне (параметр = true) або відносне (параметр = false) значення відповідно до параметру. В першому випадку параметр функції за

замовчуванням true, - в тесті інформація незмінна. В другому випадку, передаємо функції параметр false. Абсолютне значення розраховується складанням результатів тесту всіх атрибутів класу. Для відносного значення є наперед визначені: 0, 25, 50, 75, 100. З них повертається одне, до рівня якого досягло абсолютне значення.

Властивість «screen», ураховуючи її існування, зміна $percent повинна стати більшою на 15. Отже це зроблено завдяки функції перевірки assertEquals – умова відповідання значення 1 аргументу – 2:

```
$person->screen = 'http://izzi.coil.img';
$this->assertEquals(15, $person->getFilePercent());
$this->assertEquals(0, $person->getFilePercent
(false));
```

По-цьому, перевіримо властивість «name1», при попередній умові збільшення на п’ять, щоб отримати результат 20.

```
$person->name1 = 'name1';
$this->assertEqual(20, $person->getFilePercent());
$this->assertEqual(0, $person-
>getFilePercent(false));
```

Включаючи те що є «name2», сума стає більшою на 5. Тому результати стають -25

```
$person->name2 = 'name2';
$this->assertEqual(25, $person->getFilePercent());
$this->assertEqual(25, $person-
>getFilePercent(false));
```

При існуванні «FirstName», результат буде 30, а умовний – 25:

```
$person->FirstName = 'FirstName';
$this->assertEqual(30, $person->getFilePercent());
$this->assertEqual(25, $person-
>getFilePercent(false));
```

Також, з присутності «quatro», ця сума стане більшою на п’ять, отже отримаємо цілковитий результат – 35, а умовний так як и був - 25:

```
$person->quatro = 'this_quatro';
$this->assertEqual(35, $person->getFilePercent());
$this->assertEqual(25, $person->getFilePercent(false));
```

«email_ckick» - сума стає більшою на 25, в результаті чого цілковитий результат та відносний стануть - 100:

```
$person->email_cclick = true;
$this->assertEqual(100, $person->getFilePercent());
$this->assertEqual(100, $person->getFilePercent(false));
```

Звіт Проходження авто-тесту UserTest.php ілюстровано на рисунку 3.10.

```
PHPUnit 3.7.21 by Sebastian Bergmann.

Configuration read from C:\xampp\htdocs\wasabi\vidfall\www\protected\tests\phpunit.xml

.................................................

Time: 25.66 seconds, Memory: 10.75Mb

OK (52 tests, 206 assertions)
```

Рисунок 3.10 – Висновое позитивного проходження тесту «UserTest»

Можна подивитися статистику покрівлі усього програмного коду нашого проекту на рисунку 3.11

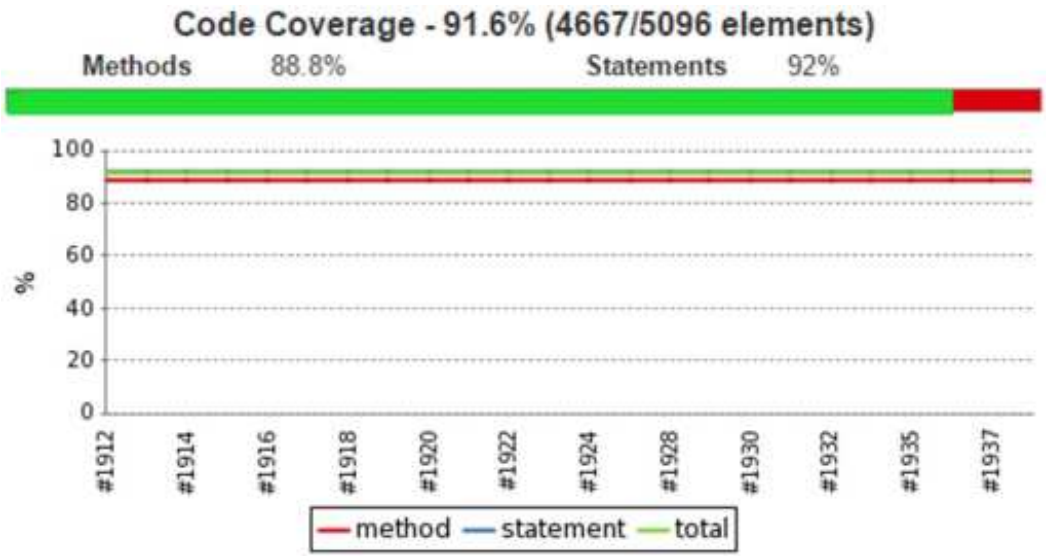

Рисунок 3.11 – Статистика покрівлі програмного коду Izzi автоматизованими тестами

Більш розгорнута інформація по кожному окремому файлу ілюстрована на рисунку 3.12, у табличці з файлами. Для кожного з них написаний модульний тест.

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		90.46%	1602 / 1771		90.39%	348 / 385		78.12%	50 / 64
Address.php		80.00%	20 / 25		80.00%	4 / 5		0.00%	0 / 1
Brand.php		100.00%	6 / 6		100.00%	4 / 4		100.00%	1 / 1
Config.php		100.00%	11 / 11		100.00%	4 / 4		100.00%	1 / 1
Content\Video.php		100.00%	12 / 12		100.00%	7 / 7		100.00%	1 / 1
Error.php		100.00%	2 / 2		100.00%	2 / 2		100.00%	1 / 1
Inventory.php		98.10%	103 / 105		91.67%	11 / 12		0.00%	0 / 1
Notification.php		96.15%	25 / 26		90.00%	9 / 10		0.00%	0 / 1
Order.php		96.74%	89 / 92		85.71%	12 / 14		0.00%	0 / 1
Product.php		99.21%	126 / 127		95.83%	23 / 24		0.00%	0 / 1
Question.php		100.00%	36 / 36		100.00%	11 / 11		100.00%	1 / 1
SpecialOffer.php		100.00%	5 / 5		100.00%	3 / 3		100.00%	1 / 1
Statistics.php		100.00%	5 / 5		100.00%	3 / 3		100.00%	1 / 1
Subscribe.php		90.91%	20 / 22		75.00%	3 / 4		0.00%	0 / 1
User.php		86.35%	430 / 498		85.07%	57 / 67		0.00%	0 / 1
View.php		100.00%	2 / 2		100.00%	2 / 2		100.00%	1 / 1
AppleWatchForm.php		100.00%	4 / 4		100.00%	2 / 2		100.00%	1 / 1
AskOrderInfoForm.php		100.00%	7 / 7		100.00%	2 / 2		100.00%	1 / 1
ChangePassForm.php		100.00%	26 / 26		100.00%	3 / 3		100.00%	1 / 1
ContactForm.php		100.00%	8 / 8		100.00%	2 / 2		100.00%	1 / 1
ContestEntryForm.php		100.00%	6 / 6		100.00%	1 / 1		100.00%	1 / 1
CouponForm.php		100.00%	15 / 15		100.00%	3 / 3		100.00%	1 / 1
ExpressCheckoutForm.php		77.78%	14 / 18		66.67%	2 / 3		0.00%	0 / 1
InviteSocialForm.php		86.21%	50 / 58		66.67%	4 / 6		0.00%	0 / 1
LoginForm.php		75.86%	22 / 29		66.67%	2 / 3		0.00%	0 / 1
RecoverPassForm.php		100.00%	12 / 12		100.00%	3 / 3		100.00%	1 / 1
RequestDemoForm.php		100.00%	11 / 11		100.00%	2 / 2		100.00%	1 / 1
RewardCodeForm.php		100.00%	15 / 15		100.00%	2 / 2		100.00%	1 / 1
SavePassForm.php		100.00%	4 / 4		100.00%	1 / 1		100.00%	1 / 1
SaveSettingsForm.php		100.00%	55 / 55		100.00%	2 / 2		100.00%	1 / 1
SaveUserForm.php		100.00%	40 / 40		100.00%	4 / 4		100.00%	1 / 1
SendInviteForm.php		100.00%	3 / 3		100.00%	1 / 1		100.00%	1 / 1
UsePointsForm.php		100.00%	2 / 2		100.00%	1 / 1		100.00%	1 / 1

Рисунок 3.12 – Список файлів, до яких є модульні тести

3.4 Висновки до розділу 3

В даному розділі було розроблено алгоритм та складено блок-схему роботи проекту системи тестування системи автоматизованого тестування на прикладі реально діючого сервісу порівняння цін, вибору товарів та пропозицій Інтернет-магазинів.

Для об’єднання різних видів тестування, безперервного автоматичного тестування та безперервної розгортки системи, було використано платформу Jenkins, яка дозволила реалізувати процес безперервної інтеграції. Завдяки Jenkins ми отримали цілісну систему автоматизованого тестування нашого проекту.

Для роботи системи було розроблено програмний продукт на основі декількох методик та мов програмування. Створено систему автоматизованої перевірки веб-додатку на основі реально існуючого проекту. Практично підтверджено надійність та ефективність системи автоматичного тестування веб-додатків.

					МР 122.6157м.ПЗ Р4		
Зм.	Лист	№ докум	Підпис	Дата			
Студент		Потужний І.Р.			Охорона праці та безпека у надзвичайних ситуаціях	Літера	Лист 59 / Листов 13
Керівник		Гайда А.Ю.					
Конс-т		Щедролосєв О.В					
Рецензент		Новиков Ю.М.				ХФ НУК ім. адм. Макарова	
Зав. Каф		Гучек П.Й.					

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Задача охорони праці

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально профілактичних мір та засобів, направлених на зберігання здоров'я та працездатності людини в процесі праці.

Правовою основою законодавства щодо охорони праці є Конституція України, Закони України: "Про охорону праці", "Про охорону здоров'я", "Про пожежну безпеку", "Про використання ядерної енергії та радіаційний захист", "Про забезпечення санітарного та епідеміологічного благополуччя населення", а також Кодекс законів про працю України.

Основним законодавчим документом про охорону праці є Закон України "Про охорону праці" від 14 жовтня 1992 року. Цей Закон визначає основні положення щодо реалізації конституційного права працівників на охорону їх життя і здоров'я у процесі трудової діяльності, на належні, безпечні і здорові умови праці, регулює за участю відповідних органів державної влади відносини між роботодавцем і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні.

Цілком безпечних і нешкідливих виробництв не існує. Задача охорони праці – звести до мінімуму ймовірність поразки або захворювання працюючих із забезпеченням комфорту при максимальній продуктивності праці. Реальні виробничі умови характеризуються наявністю небезпечних і шкідливих виробничих факторів.

Значення охорони праці вже давно вийшло за рамки тільки соціальної сфери. Вона значно впливає на економічні показники підприємств, що в умовах прискорення науково-технічного прогресу надзвичайно важливо. Погані умови

праці являються однією з важливих причин зниження працездатності та підвищення стомлюваності.

4.2 Аналіз потенційно небезпечних та шкідливих факторів при розробці підсистеми підтримки проектів високовольтних електричних мереж

Відповідно до ГОСТ 12.0.003-74 “Опасные и вредные производственные факторы. Классификация” небезпечні та шкідливі фактори за природою дії поділяються на такі групи: фізичні, хімічні, біологічні та психофізіологічні.

Безпека виробничих процесів забезпечується цілим комплексом проектних і організаційних рішень, що полягають у відповідному виборі технологічних процесів, робітників операцій і порядку обслуговування устаткування; виробничих помешкань або зовнішніх площадок; виробничого устаткування й умов його розміщення; засобів збереження і транспортування вихідних матеріалів, заготівель готової продукції і відходів виробництва; засобів захисту працюючих. Велике значення має організація фахового добору і навчання працюючих.

У виробничих умовах можуть виникнути такі ситуації, коли ушкоджується організм людини, що буває можливо в результаті несправності устаткування, недосконалості його конструкції, порушення норм і правил охорони праці.

4.2.1 Електробезпека.

Аналіз причин нещасних випадків показує, щоконтакт людини з проводами і струмоведучими частинами частіше відбувається випадково і не викликаються виробничою необхідністю. Крім того ураження струмом виникають при помилковій подачі електроенергії. При постійній роботі з електричними установками, що знаходяться під напругою варто пам’ятати про небезпеку ураження струмом. Основним заходом, що забезпечує безпеку обслуговування електроустаткування при його нормальній роботі, є ізоляція всіх струмоведучих

частин. Вона повинна бути розрахована так, щоб сила струму, що проходить через людину, що обслуговує електроустаткування, під час його експлуатації не перевищувала значень, установлених ГОСТ 12.1.019-79 “Электробезопасность. Общие требования и номенклатура видов защиты”.

Ще більшу небезпеку для людини являє зниження опору ізоляції електромереж.

Для захисту від ураження людей струмом при контакті з корпусами й іншими частинами електроустаткування, що звичайно не знаходиться під напругою, ці частини з’єднуються із землею. При цьому опір пристрою, що заземлює, не повинне бути більше 4 Ом.

4.2.2 Освітлення.

Важливим фактором для підвищення виробництва праці являється правильно запроектовані і виконанні освітлення робочого місця. Так як освітлення недостатньо, то застосовують загальне штучне освітлення. Штучне освітлення передбачається в усіх виробничих та побутових приміщеннях, де недостатньо природного світла, а також для освітлення приміщень в темний період доби. При організації штучного освітлення необхідно забезпечити сприятливі гігієнічні умови для зорової роботи.

4.2.3 Захист від шуму.

При роботі електричного обладнання виникає шум, який шкідливо діє на організм людини, і надає вплив на його психологічний стан.

Тривала дія шуму може привести до розлагодження та до захворювання органів слуху, серцево-судинної та нервової систем. Відповідно до ДБН В.1.1-31:2013 “Захист територій, будинків і споруд від шуму” нормовою шумовою характеристикою робочого місця при постійному шумі являються рівні звукового тиску в децибелах. Згідно ДБН В.1.1-31:2013 “Захист територій, будинків і споруд від шуму” встановленні допустимі норми звукового тиску та

шуму по октавним полюсам: 63, 125, 259, 500, 1000, 2000, 4000, 8000 Гц. Допустимі рівні шуму в багатьом залежні від характеру діяльності особового складу. Згідно ДБН В.1.1-31:2013 “Захист територій, будинків і споруд від шуму” допустимими рівнями шуму являються 60…75 дБ у відповідальних місцях, та 75…80 дБ в усіх інших приміщеннях.

Для зниження рівня шуму застосують різноманітні звукопоглинальні покриття, машині та механізми з можливо більш нижчим рівнем шуму.

Методи та засоби захисту від шуму:

- боротьба з шумом в джерелі його виникнення;

- зниження шуму звукопоглинанням та звукоізоляцією;

- зниження шуму акустичною обробкою приміщення.

4.2.4 Захист від вібрації.

Вібрація – це механічні коливання з частотою 0,1…1000 Гц, виникаючі в результаті роботі двигунів та інших машин. Загальна класифікація засобів та методів захисту від вібрації приведені в ГОСТ 12.1.012-90 “Вибрационная безопасность. Общие требования”. Вібрацію що діє на людину, нормують окремо для кожного встановленого напрямку згідно з ГОСТ 12.1.012-90 “Вибрационная безопасность. Общие требования”.

Існує три основних принципи боротьби з вібрацією:

- зменшення сил, викликаючи вібрацію (точна збірка механізмів, балансування і т.д.);

- усунення резонансів;

- штучні зусилля постійно виникаючого розсіювання енергії, тобто введення в систему демпфування.

4.2.5 Пожежна безпека.

Правовою основою діяльності в галузі пожежної безпеки є Конституція України, Закон України “Про пожежну безпеку”, прийнятий 17 грудня 1993

року, та інші закони України, укази та розпорядження Президента України, постанови та розпорядження Кабінету Міністрів України. Забезпечуючи пожежну безпеку слід також керуватись Правилами пожежної безпеки в Україні, стандартами, будівельними нормами, Правилами улаштування електроустановок (ПУЕ), нормами технологічного проектування та іншими нормативними актами, виходячи із сфери їх дії, які регламентують вимоги пожежної безпеки.

Під системою пожежного захисту і вибухозахисту розуміються комплекси організаційних заходів і технічних засобів, спрямованих на запобігання впливу на людей небезпечних чинників пожежі і вибуху, а також обмеження матеріального збитку.

До джерел запалювання у виробничих умовах відносять:

- вогонь і іскри, не викликані прийнятою технологією, наприклад при роботі двигуна;

- теплові прояви механічної енергії: розігрів тіл при терті, нагрівання газу при його стиску, іскри при зіткненні твердих тіл;

- теплові прояви електричної енергії при коротких замиканнях і перевантаженнях, несправності й ушкодження ізоляції;

- теплові виділення при деяких хімічних реакціях.

Пожежо- і вибухонебезпечність речовин і матеріалів за ГОСТ 12.1.044-89 “Пожаробезопасность веществ и материалов. Номенклатура показателей и методы их определения” (сюди не входять вибухові і радіоактивні речовини й матеріали) визначаються залежно від агрегатного стану речовини й умов застосування.

При несправностях в електричних мережах і електроустаткуванні електрична енергія перетворюється в теплову, в результаті температура окремих елементів може значно підвищуватися, з’являється іскріння, що спричиняє пожежу. Аналіз причин виникнення пожеж показує, що на короткі замикання доводиться 43,3%, на порушення правил використання нагрівальних приладів

33,3%, на перевантаження електродвигунів і мереж 12,3%, на утворення більших місцевих опорів 4,6%, на іскріння 3,3%, на інші причини 3,3%.

Небезпека вибухів і пожеж може виникнути і при нормальній експлуатації електроустаткування в результаті іскріння в пусковій апаратурі, підвищення температури окремих елементів.

Попередження вибухів і пожеж, при експлуатації електричних мереж і електроустаткування досягається їхнім конструктивним виконанням. Однак це приводить до значного подорожчання електроустаткування, тому при його виборі необхідно також враховувати стан повітряного середовища. Сучасні протипожежні системи автоматично можна розділити на три види:

- системи і пристрої пожежної профілактики, що забезпечують пожежо- і вибухобезпечність технологічних процесів;

- системи пристрою пожежної сигналізації;

- системи автоматичного гасіння пожеж.

В комплексі заходів, що використовуються в системі протипожежного захисту, важливе значення має вибір найбільш раціональних способів та засобів гасіння різних горючих речовин та матеріалів згідно зі СНиП 2.04.09-84 “Пожарная автоматика зданий и сооружений”.

Горіння припиняється:

- при охолодженні горючої речовини до температури нижчої, ніж температура її займання;

- при зниженні концентрації кисню в повітрі в зоні горіння;

- при припиненні надходження пари, газів горючої речовини в зону горіння.

- Припинення горіння досягається за допомогою вогнегасних засобів:

- води (у вигляді струменя або у розпиленому вигляді);

- інертних газів (вуглекислота);

- хімічних засобів (у вигляді піни або рідини);

- порошкоподібних сухих сумішей.

Вибір тих чи інших способів та засобів гасіння пожеж та вогнегасних речовин і їх носіїв визначається в кожному конкретному випадку залежно від стадії розвитку пожежі, масштабів загорянь.

Успіх швидкої локалізації та ліквідації пожежі на її початку залежить від наявних вогнегасних засобів, вміння користуватися ними всіма працівниками, а також від засобів пожежного зв'язку та сигналізації для виклику пожежної допомоги та введення в дію автоматичних та первинних вогнегасних засобів.

4.3 Безпека в надзвичайних ситуаціях

Надзвичайна ситуація (НС) — порушення нормальних умов життя і діяльності людей на об'єкті або території, спричинене аварією, катастрофою, стихійним лихом, епідемією, епізоотією, епіфітотією, великою пожежею, застосуванням засобів ураження, що призвели або можуть призвести до людських і матеріальних втрат.

Актуальність проблеми природно-техногенної безпеки населення і території обумовлена тенденціями зростання втрат людей і шкоди територіям в результаті небезпечних природних явищ і катастроф . Ризик надзвичайних ситуацій техногенного і природного характеру постійно зростає.

У Законі України "Про захист населення і території від надзвичайних ситуацій техногенного і природного характеру" враховані вимоги сформованих обставин і часу, визначені завдання, принципи і способи захисту населення в надзвичайних ситуаціях.

Основними завданнями захисту населення і території від надзвичайних ситуацій техногенного і природного характеру є:

- здійснення комплексу заходів щодо запобігання і реагування на надзвичайні ситуації техногенного і природного характеру;

- забезпечення готовності і контролю за станом готовності до дій і взаємодії органів управління в цій сфері, сил і засобів, призначених для запобігання

надзвичайних ситуацій техногенного і природного характеру і реагування на них.

Захист населення і території від надзвичайних ситуацій техногенного і природного характеру здійснюється за принципами:

- пріоритетності завдань, спрямованих на порятунок життя і збереження здоров'я людей та навколишнього середовища;

- безперечної переваги раціональної і превентивної безпеки;

- вільного доступу населення до інформації про захист населення і території від надзвичайних ситуацій техногенного і природного характеру;

- особистої відповідальності і турботу громадян про власну безпеку, неухильного дотримання ними правил поведінки і дій у надзвичайних ситуаціях техногенного і природного характеру;

- відповідальності в межах своїх повноважень, посадових осіб за дотриманням вимог даного Закону;

- обов'язковості завчасної реалізації заходів, спрямованих на попередження виникнення надзвичайних ситуацій техногенного і природного характеру і мінімізацію їх негативних психіко-соціальних наслідків;

- врахування економічних, природних та інших особливостей території і ступеня реальної небезпеки виникнення надзвичайної ситуації техногенного і природного характеру;

- максимально можливого, ефективного і комплексного використання наявних сил і засобів, призначених для запобігання надзвичайних ситуацій техногенного і природного характеру та реагування на них.

Політика в цій сфері фактично формується двома документами – "Концепцією захисту населення і територій в разі загрози та виникнення надзвичайних ситуацій", схваленою Указом Президента України від 26 березня 1999 р. № 234/99. Цей документ встановлює основні напрями, мету та завдання захисту населення і територій.

Основними завданнями захисту населення і територій під час надзвичайних ситуацій є:

- розроблення і реалізація нормативно-правових актів, додержання державних технічних норм та стандартів з питань забезпечення захисту населення і територій від наслідків надзвичайних ситуацій;

- забезпечення готовності органів управління, сил і засобів до дій, призначених для запобігання надзвичайним ситуаціям та реагування на них;

- розроблення та забезпечення заходів щодо запобігання виникненню надзвичайних ситуацій;

- збирання та аналітичне опрацювання інформації про надзвичайні ситуації;

- прогнозування та оцінка соціально-економічних наслідків надзвичайних ситуацій, визначення на основі прогнозу потреби в силах, матеріально-технічних і фінансових ресурсах;

- створення, раціональне збереження і використання резервів фінансових і матеріальних ресурсів, необхідних для запобігання надзвичайним ситуаціям та реагування на них;

- здійснення державної експертизи, нагляду і контролю в галузі захисту населення і територій від надзвичайних ситуацій;

- оповіщення населення про загрозу та виникнення надзвичайної ситуації і своєчасне та достовірне інформування його про наявну обстановку і вжиті заходи;

- організація захисту населення (персоналу) та надання безкоштовної медичної допомоги;

- проведення рятувальних та інших невідкладних робіт щодо ліквідації наслідків надзвичайних ситуацій та організація життєзабезпечення постраждалого населення;

- здійснення заходів щодо соціального захисту постраждалого населення;

- розроблення та забезпечення цільових і науково-технічних програм, спрямованих на запобігання надзвичайним ситуаціям та забезпечення сталого

функціонування підприємств, установ, організацій, незалежно від форм власності та підпорядкування, а також підвідомчих їм об'єктів виробничого і соціального призначення;

- реалізація визначених законодавством прав населення в галузі захисту від наслідків надзвичайних ситуацій, у тому числі осіб (чи їхніх сімей), які брали безпосередню участь в їх ліквідації;

- навчання та тренування населення способів захисту у разі виникнення надзвичайних ситуацій;

- міжнародне співробітництво у сфері захисту населення від надзвичайних ситуацій.

Захист населення і територій під час надзвичайних ситуацій забезпечується скоординованою роботою постійно діючих функціональних і територіальних підсистем єдиної державної системи (ЄДС).

Функціональні підсистеми створюються міністерствами та іншими центральними органами виконавчої влади для організації роботи, пов’язаної із запобіганням надзвичайним ситуаціям та захистом населення і територій від їх наслідків. У надзвичайних ситуаціях сили і засоби функціональних підсистем регіонального, місцевого та об’єктового рівня підпорядковуються в межах, що не суперечать законодавству, органам управління відповідних територіальних підсистем єдиної державної системи.

Перелік функціональних підсистем визначається Кабінетом Міністрів України.

Організація, склад сил і засобів, порядок діяльності функціональних підсистем захисту населення і територій визначаються положеннями про них, затвердженими керівниками відповідних міністерств, інших центральних органів виконавчої влади за погодженням з Міністерством України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи.

Територіальні підсистеми створюються в усіх областях, місті Києві для запобігання і ліквідації наслідків надзвичайних ситуацій.

Завдання, організація, склад сил і засобів, порядок функціонування територіальних підсистем захисту населення і територій визначаються положеннями про ці підсистеми, затвердженими Головою Ради міністрів Автономної Республіки Крим та головами відповідних державних адміністрацій за погодженням з МНС України.

Порядок дій у разі виникнення пожежі. При виникненні пожежі дії адміністрації підприємства, компанії, в першу чергу, мають бути спрямовані на забезпечення евакуації людей і матеріальних цінностей, а також гасіння пожежі.

Кожний працівник у разі пожежі зобов'язаний:

- негайно повідомити про це в пожежну службу;

- повідомити посадових осіб підприємства, компанії;

- приступити до евакуації людей і матеріальних цінностей, гасіння пожежі;

- за необхідності викликати інші аварійно-рятувальні служби.

Остаточні свої дії щодо гасіння пожежі керівник гасіння погоджує з керівником підприємства або особою, яка виконує його обов'язки.

Посадова особа підприємства, компанії, яка прибула до місця пожежі зобов'язана:

- перевірити, чи викликана пожежна охорона (продублювати повідомлення), довести подію до відома керівника підприємства, компанії;

- у разі загрози життю людей негайно організувати їх рятування (евакуацію);

- припинити всі роботи, крім робіт, пов'язаних із заходами щодо гасіння пожежі, і вивести з небезпечної зони всіх працівників, не пов'язаних з гасінням пожежі;

- відключити електроенергію, агрегати, подавання газу, вентиляцію в приміщення, що горять, і в суміжні приміщення, а також виконати інші заходи, що сприяють запобіганню поширенню полум'я;

- привести в дію систему оповіщення людей про пожежу, за необхідності привести в дію установки пожежогасіння;

- організувати зустріч підрозділів пожежної охорони, надати їм допомогу при виборі найкоротшого шляху для підходу до осередку пожежі та в установці на водні джерела;

- одночасно з гасінням пожежі організувати евакуацію і захист матеріальних цінностей;

- забезпечити дотримання техніки безпеки працівниками, які беруть участь у гасінні пожежі;

- повідомити керівника підрозділу пожежної охорони, який прибув до місця пожежі, про кількість людей, які потребують евакуації, і їх місце перебування, про кількість людей , які беруть участь у гасінні пожежі, про місце, розмір і характер пожежі, ужиті заходи для гасіння пожежі.

Після прибуття до місця пожежі пожежних підрозділів повинен бути забезпечений безперешкодний пропуск їх на територію підприємства.

Протягом усього часу гасіння пожежі посадова особа підприємства консультує керівника гасіння пожежі про конструктивні та інші особливості підприємства.

					КР122.6157м.ПЗ Р5			
Зм.	Лист	№ докум	Підпис	Дата				
Студент		Потужний І.Р.			Розрахунок економічного ефекту від впровадження автоматизованої системи з волатильною бізнес логікою	Литера	Лист	Листов
Керівник		Гайда А.Ю.					72	7
						ХФ НУК ім. адм. Макарова		
Рецензент		Новиков Ю.М.						
Зав. Каф		Гучек П.Й.						

5 РОЗРАХУНОК ЕКОНОМІЧНОГО ЕФЕКТУ ВІД ВПРОВАДЖЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ З ВОЛАТИЛЬНОЮ БІЗНЕС-ЛОГІКОЮ

5.1 Вступ до проблеми

Інформаційні системи широко використовуються багатьма організаціями України різної форми власності. Ці системи розповсюджуються все більше і більше, змінюючись відповідно до вимог замовників та модернізуючись з плином часу.

Створення інформаційної системи вимагає одноразових витрат на її розробку та придбання необхідних технічних засобів, а також поточних витрат, пов’язаних з функціонуванням підсистеми. Економічний ефект від експлуатації системи визначається з урахуванням витрат на її експлуатацію. Відношення економічного ефекту до витрат на створення системи характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються за діючими на момент розрахунку цінами, тарифами та ставками заробітної плати.

5.2 Розрахунок вартості створення автоматизованої системи з волатильною бізнес-логікою

Загальні витрати (на проектування, реалізацію та впровадження системи)

$$З = Сп\text{-}р + Стз + Спз + Свпр,$$

де Сп-р – витрати на проектування і реалізацію;

Стз – витрати на придбання або модернізацію обчислювальної техніки, периферійних пристроїв, засобів зв'язку, допоміжного устаткування та оргтехніки (з урахуванням витрат на транспортування, монтаж, налагодження і запуск), виробничо-господарського інвентарю;

Спз – витрати на придбання та інсталяцію програмного забезпечення (ПЗ), необхідного для розробки;

Свпр – витрати на впровадження системи (навчання персоналу тощо). Витрати на проектування та реалізацію розраховуються за формулою

$$Сп\text{-}р = Сз + Свід + См + Сн,$$

де Сз – заробітна плата розробників;

Свід – відрахування на соціальні заходи (відповідно до чинного законодавства Свід становить 47,5% від фонду оплати праці);

См - експлуатація та амортизація, См=Зекспл+Замор;

Сн – накладні витрати у розмірі 50-150% від витрат на оплату праці (охоплюють оплату праці управлінського персоналу з нарахуваннями; оплату службових відряджень; оплату консультаційно-інформаційних послуг; оплату ремонту і техобслуговування основних фондів, крім персональних комп'ютерів (ПК)).

1) Розробка підсистеми вимагає 6 місяців (експертна оцінка, яка базується на досвіді розробки систем такого класу). Заробітна платня розробників за місяць дорівнює 12000 грн (відповідно до сучасних розцінок та попиту на спеціалістів такого профілю). Кількість розробників – одна людина.

$$Сз = 1 * 12000 * 6 = 72000 \text{ грн.}$$

$$Свід = 72000 * 0{,}475 = 34200 \text{ грн.}$$

2) витрати на амортизацію ЕОМ, на якій виконується розробка:

$$Замор = (Цесм /Тсл) (Тразр /12),$$

де Цесм = 6000грн. – балансова вартість ЕОМ;

Тсл = 5 років – термін служби ЕОМ;

Тразр = 0,5 року – час, необхідний для розробки системи, виражений в місяцях, і тому ділиться на 12, щоб одержати число років.

$$Замор = (6000/5)\cdot 0{,}5 = 600 \text{ (грн.)};$$

3) витрати на експлуатацію ЕОМ, на якій виконується розробка, полягають в оплаті споживаної нею електричної енергії:

$$Зекспл = Рєвм*Тразр*Nрд*Nрч*Єст,$$

де Рєвм = 0,4 кВт/год – потужність ЕОМ;

Тразр = 6 міс. – тривалість розробки;

Nрд = 21день – число робочих днів у місяці;

Nрч = 8 годин – число годин у робочому дні;

Эст = 0,3073 грн – вартість 1 кВт електроенергії.

Зекспл = 0,4 6 21 8 0,3073 = 123,90(грн);

См=600+123,90=723,90 (грн.)

Сн=72000*0,3=21600 (грн.)

Сп-р=72000+34200+723,90+21600= 128 523,90 (грн.)

Витрати на програмне забезпечення не закладаються у даний кошторис, оскільки в роботі системи планується використовувати безкоштовне ПЗ з відкритими вихідними кодами.

Витрати на апаратне забезпечення складають 19981 грн (придбання серверу та комутаційного обладнання і матеріалів).

До витрат на впровадження системи можна віднести витрати вартість навчання кадрів, витрати на монтаж і настроювання мережі, вартість устаткування, вартість програмного забезпечення.

Таким чином, загальні витрати на проектування, реалізацію та впровадження системи складають

З = Сп-р + Стз + Спз + Свпр = 128 523,90 + 20781 =149304,90 (грн).

Витрати на створення та експлуатацію системи в перший рік експлуатації

Зсе = З + Зер,

де Зер – експлуатаційні витрати на технічне забезпечення за рік.

Зер = АОБ + Зм + Зе,

де Аоб – амортизаційні відрахування на устаткування;

Зм – витрати на основні і допоміжні матеріали;

Зе – витрати на електроенергію.

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік (за чинним законодавством)

Аоб = 20781 * 0,6 = 12468,6 грн.

В масштабі підприємства річні витрати на основні та допоміжні матеріали (гнучкі диски, папір тощо) визначаються в розмірі 3% вартості основного устаткування

$$ЗМ = 20781 * 0,03 = 623,43 \text{ грн.}$$

Річний обсяг робіт ПК у нормо-годинах

$$Фм = 253,3 * ТС,$$

де ТС – середнє місячне завантаження устаткування (близько 5 годин);

253,3 – середня кількість робочих днів у році.

$$Фм = 253,3 * 5 = 1266,5 \text{ годин.}$$

Витрати на електроенергію під час 1266,5 годин роботи устаткування в рік при потужності ЕОМ 0,4 кВт/год та ціні 0,3073 грн за 1 кВт електроенергії складуть

$$Зе = 1266,5 * 0,3073 * 0,4 = 155,68 \text{ грн.}$$

Експлуатаційні витрати на технічне забезпечення за рік

$$Зер = 12468,6 + 623,43 + 155,68= 13247,71 \text{ грн.}$$

Отже, у перший рік витрати на створення та експлуатацію системи складуть

$$Зсе = 149304,90 + 13247,71= 162552,61 \text{ грн.}$$

5.3 Розрахунок економічної ефективності

Економічна ефективність використання розподіленої інформаційної системи на базі технології хмарних обчислень полягає у наступному:

- зменшення часу на обслуговування системи за рахунок її відмовостійкості та автоматичного масштабування;
- можливість надання користувачам обчислювальних потужностей в оренду.

Це зменшення відбудеться за рахунок зменшення складності користування системою, збільшення надійності, стабільності та рівня безпеки. Визначимо, що це зменшення складе 50% від сучасного.

Однак, не піддається прямій грошовій оцінці зменшення кількості помилкових та необережних рішень, підвищення оперативності керування,

поліпшення організації роботи та своєчасне отримання необхідної інформації про пріоритетність варіантів рішень тощо.

Використання даної системи дозволяє вивільнити 0,2 робочого часу. Так як з системою працює 10 робітників, то система умовно вивільнить 2 робітників. Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження системи вивільнить 2 робітників.

Річна платня за рахунок вивільнення робочого часу складе

$$\Delta Cn = 12000*2*12 - 13247,71= 274752,29 \text{ грн.}$$

Річний економічний ефект розраховується за наступною формулою:

$$Эгод = \Delta Сп – Еп*k$$

де Еп – нормативний коефіцієнт економічної ефективності капітальних вкладень у галузь і для обчислювальної техніки приймається 0,5;

k – додаткові капітальні вкладення з урахуванням витрат на проектування, створення і функціонування системи.

$$Эгод = 274\ 752,29 – 0,5 * 162\ 552,61 = 193\ 475,98 \text{ (грн).}$$

Строк окупності підсистеми

$$T = k / \Delta Cn \text{ , тобто } T = 162\ 552,61 / 274\ 752,29 = 0,6 \text{ (року).}$$

Таким чином, витрати на створення розподіленої інформаційної системи окупляться протягом 0,6 року.

5.4 Підсумки

В результаті розрахунків витрати на створення системи обліку робочого часу (проектування, реалізація і впровадження) складуть 149304,90 грн; експлуатаційні витрати щорічно складатимуть 13247,71 грн. Загалом в перший рік витрати на модернізацію й експлуатацію складуть 162552,61 грн.

Вивільнюється 2 умовних фахівці.

Вивільнена річна зарплатня працівників складає 274752,29 грн.

Витрати на створення системи окупляться протягом 0,6 року.

За отриманими значеннями річного економічного ефекту та строку окупності можливо зробити висновок про економічну вигідність і обґрунтованість створення розподіленої інформаційної системи. Слід зазначити, що підвищити економічну ефективність цієї системи можна за рахунок запропонування стороннім користувачам обчислень як послуги. Таким чином можна буде скоротити строк, за який окупляться витрати на створення системи.

					КР.122.6157м.ПЗ Р6		
Зм.	Лист	№ докум	Підпис	Дата			
Студент		Потужний І.Р.			Охорона навколишнього середовища	Літера	Лист 79 / Листов 9
Керівник		Гайда А.Ю.					
Резензент		Новиков Ю.М.				ХФ НУК ім. адм. Макарова	
Зав. Каф		Гучек П.Й.					

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1 Вступ до проблеми

Охорона навколишнього середовища здійснюється різними, у тому числі й правовими, засобами. При цьому в правових формах захищаються переважно всі компоненти, які утворюють природне середовище.

Сучасними головними нормативно-правовими актами що регулюють основи організації охорони навколишнього природного середовища, є Закони України «Про охорону навколишнього природного середовища» від 25 червня 1991 р., «Про охорону атмосферного повітря» від 16 жовтня 1992 р., «Про природно-заповідний фонд України» від 16 червня 1992 р., «Про тваринний світ» від 3 березня 1993 р., «Про карантин рослин» від 30 червня 1993 р та інші. До того ж деякі відносини у сфері використання і охорони навколишнього природного середовища врегульовані кодексами (земельним, водним, лісовим, про надра), а також Законами України «Про плату за землю» від 3 липня 1992 р., «Про ветеринарну медицину» від 25 червня 1992 р. Важливе значення у вирішенні цього питання має затверджений Постановою Верховної Ради «Порядок обмеження, тимчасової заборони (зупинення) чи припинення діяльності підприємств, установ, організацій і об'єктів у разі порушення ними законодавства про охорону навколишнього природного середовища».

Різновидами права природокористування є: право землекористування, право водокористування, право лісокористування, право користуватися надрами, право користуватися тваринним світом, право користування природнозаповідним фондом. Право природокористування це процес раціонального використання людиною природних ресурсів для задоволення різних потреб та інтересів. Найважливішими принципами природокористування є його цільовий характер, плановість і тривалість, ліцензування, врахування надзвичайного значення у житті суспільства тощо. При цьому виділяють такі групи природокористування, як право загального і спеціального використання

землі, вод, лісів, надр, тваринного світу та інших природних ресурсів. Суб'єктами права загального користування природними ресурсами можуть бути, згідно з Законом України «Про охорону навколишнього природного середовища», усі громадяни для задоволення найрізноманітніших потреб і інтересів. Воно здійснюється громадянами безкоштовно і безліцензійно, тобто для цього не потрібен відповідний дозвіл уповноважених органів і осіб. Загальним є, наприклад, використання парків, скверів, водойм, лісів, збір дикорослих ягід, грибів, горіхів і т. ін. Право загального природокористування закріплене у ст. 13 Конституції України: «Кожний громадянин має право користуватися природними об'єктами права власності народу відповідно до закону». Похідним від загального природокористування є спеціальне використання природних ресурсів. На відміну від першого, це використання конкретних природних ресурсів, що здійснюється громадянами, підприємствами, установами і організаціями у випадках, коли відповідна, визначена у законодавстві частина природних ресурсів передається їм для використання. Як правило, така передача має вартість і визначена в часі. Надання природних ресурсів відбувається на основі спеціальних дозволів державних актів на право постійного користування.

Крім прав суб'єктів, як природокористувачів, сучасною юридичною наукою сформовані й інтенсивно розвиваються екологічні права і обов'язки. Так, у Конституції України записано, що «кожен має право на безпечне для життя і здоров'я довкілля та на відшкодування завданої порушенням цього права шкоди. Кожному гарантується право вільного доступу до інформації про стан довкілля, про якість харчових продуктів і предметів побуту, а також право на її поширення». Аналогічні формулювання є й у Законі України «Про охорону навколишнього природного середовища», бо це право одне з основних прав людини. Цьому праву відповідає обов'язок держави забезпечувати здійснення санітарно-гігієнічних заходів, спрямованих на поліпшення та оздоровлення навколишнього природного середовища. Усі екологічні права громадян

захищаються і відновлюються у судовому порядку. Поряд з правами Закон України «Про охорону навколишнього природного середовища» передбачає і певні обов'язки громадян. Так, незалежно від того, є громадяни природокористувачами, чи ні, вони зобов'язані берегти природу, раціонально використовувати її запаси, не завдавати шкоди. Крім того, Закон України «Про охорону навколишнього природного середовища» покладає на громадян і підприємства, установи й організації, як суб'єктів спеціального використання природних ресурсів, спеціальні обов'язки. Так, плата за спеціальне природокористування встановлюється на основі нормативів плати і лімітів використання природних ресурсів. Ці нормативи визначаються з урахуванням розповсюдження природних ресурсів, їх якості, можливості використання, місцезнаходження, можливості переробки і зберігання відходів. До того ж суб'єкти спеціального природокористування зобов'язані сплачувати певні кошти за забруднення навколишнього природного середовища, що встановлюються за викиди у атмосферу забруднюючих речовин; скидання забруднюючих речовин на поверхню води, у територіальні і морські води, а також під землю.

Контроль у сфері природовикористання і охорони навколишнього природного середовища здійснюється шляхом перевірки, нагляду, обстеження, інвентаризації та експертиз. Він може здійснюватись як уповноваженими державними органами, так і громадськими формуваннями. Державний контроль покладається на Ради народних депутатів, державні адміністрації та Міністерство охорони навколишнього природного середовища і його органи на місцях.

До основних пріоритетів охорони довкілля та раціонального використання природних ресурсів належать:

- гарантування екологічної безпеки ядерних об'єктів і радіаційного захисту населення та довкілля, зведення до мінімуму шкідливого впливу наслідків аварії на Чорнобильській АЕС;
- поліпшення екологічного стану басейнів рік України та якості питної води;

- стабілізація та поліпшення екологічного стану в містах і промислових центрах Донецько-Придніпровського регіону;
- будівництво нових та реконструкція діючих потужностей комунальних очисних каналізаційних споруд;
- запобігання забрудненню Чорного та Азовського морів і поліпшення їх екологічного стану;
- формування збалансованої системи природокористування та адекватна структурна перебудова виробничого потенціалу економіки, екологізація технологій у промисловості, енергетиці, будівництві, сільському господарстві, на транспорті;
- збереження біологічного та ландшафтного різноманіття, заповідна справа.

Для досягнення цього передбачається вирішення таких завдань:

1) зменшення до мінімуму рівня радіаційного забруднення;
2) захист повітряного басейну від забруднення, насамперед у великих містах і промислових центрах;
3) захист і збереження земельних ресурсів від забруднення, виснаження і нераціонального використання;
4) збереження і розширення територій з природним станом ландшафту, посилення природоохоронної діяльності на заповідних і рекреаційних територіях;
5) підвищення стійкості та екологічних функцій лісів;
6) знешкодження, утилізація та захоронення промислових та побутових відходів;
7) запобігання забрудненню морських і внутрішніх вод, зменшення та припинення скиду забруднених стічних вод у водні об'єкти, захист підземних вод від забруднення;
8) збереження та відродження малих річок, здійснення управління водними ресурсами на основі басейнового принципу;
9) завершення створення державної системи моніторингу навколишнього природного середовища;

10) створення системи прогнозування, запобігання та оперативних дій у разі надзвичайних ситуацій природного і природно-техногенного походження;
11) забезпечення екологічного супроводу процесу конверсії військовопромислового комплексу;
12) здійснення заходів щодо екологічного контролю за діяльністю Збройних Сил України;
13) розробка механізмів реалізації схем природокористування;
14) впровадження дійових економічних складових впливу на систему природокористування;
15) створення системи екологічної освіти, виховання та інформування.

6.2 Забруднення навколишнього середовища підприємством з розробки програмного забезпечення

Основними чинниками забруднення навколишнього середовища фірмою є лише використання автотранспорту. Бо підприємство окрім забруднення повітряного середовища не забруднює ні водне, ні ґрунтове.

В сучасних умовах автомобільний транспорт стає найбільш значним джерелом забруднення атмосферного повітря, особливо великих міст.

Відомо, що атмосферне повітря міст постійно забруднюється і за всіма параметрами докорінно відрізняється від повноцінного природного повітря. Міські поселення характеризуються найвищими рівнями антропогенних навантажень на навколишнє середовище, в результаті чого воно деформується, набуває якісно нових рис, аж до зміни мікрокліматичних факторів і фізикохімічних властивостей середовища, зокрема повітряного басейну. Метеорологічні спостереження свідчать, що температура повітря в межах міських територій у середньому на декілька градусів вища, ніж у приміських районах. Над містами, особливо великими, частіше випадають атмосферні опади, бувають густі тумани, а також смоги – густі тумани, змішані з димом, кіптявою та вихлопними газами. Прозорість атмосфери в містах менша, ніж за її межами і

в сільській місцевості. Тумани, а також запиленість повітря помітно зменшують проникнення до земної поверхні сонячних променів.

Однак за рахунок автомобільних викидів якість атмосферного повітря в містах погіршилась.

Прогресуючому забрудненню атмосфери в містах сприяє висока питома вага автомобілів, адже зростання кількості автотранспортних засобів супроводжується збільшенням об'ємів забруднюючих речовин із вихлопних труб. Останнім часом у міському повітрі виріс об'єм оксидів вуглецю, вуглеводнів, оксидів азоту, сажі. Але найбільшу небезпеку, окрім оксидів азоту, складають сірчані та свинцеві сполуки. Їх вміст у міському повітрі значною мірою виріс. Місто не пристосоване до такої кількості автотранспорту. Довжина пробігу без зупинок між світлофорами становить лише 400–600 м., внаслідок цього середня швидкість руху вдень у центрі міста і на великих автошляхах знижується до 12–20 км/год, а це збільшує витрати палива в 3–4 рази. Відповідно збільшуються й викиди забруднюючих речовин. Автотранспорт також призводить до специфічних форм забруднення повітря. При русі стираються шини і тисячі тонн гуми у вигляді пилу потрапляє в повітря. Автомобільний транспорт не тільки забруднює повітря продуктами згорання палива, він сприяє зростаючому надходженню свинцю в навколишнє середовище. В Україні поки ще використовують бензин із вмістом свинцю 0,36 г/л, тоді як в Англії, Німеччині та США – 0,013–0,15.

6.3 Розробка заходів щодо зменшення забруднення

Природоохоронною є будь-яка діяльність, спрямована на збереження якості навколишнього середовища на рівні, що забезпечує стійкість біосфери. До неї відноситься як великомасштабна, здійснювана на загальнодержавному рівні діяльність по збереженню еталонних зразків недоторканої природи і збереженню розмаїтості видів на Землі, організації наукових досліджень, підготовці фахівців-екологів і вихованню населення, так і діяльність окремих підприємств по

очищенню від шкідливих речовин стічних вод і відхідних газів, зниженню норм використання природних ресурсів і т.д. Така діяльність здійснюється в основному інженерними методами.

Існують два основних напрямки природоохоронної діяльності підприємств. Перший – очищення шкідливих викидів. Цей шлях «у чистому вигляді» малоефективний, тому що з його допомогою далеко не завжди вдається цілком припинити надходження шкідливих речовин у біосферу. До того ж скорочення рівня забруднення одного компонента навколишнього середовища веде до посилення рівня забруднення іншого.

І, наприклад, встановлення вологих фільтрів при газоочищенні дозволяє скоротити забруднення повітря, але веде до ще більшого забруднення води. Уловлені з відхідних газів і зливальних вод речовини часто отруюють значні земельні площі.

Використання очисних споруд, навіть найефективніших, різко скорочує рівень забруднення навколишнього середовища, однак не вирішує цієї проблеми цілком, оскільки в процесі функціонування цих установок теж виробляються відходи, хоча й у меншому обсязі, але, як правило, з підвищеною концентрацією шкідливих речовин. Нарешті, робота більшої частини очисних споруджень вимагає значних енергетичних витрат, що, у свою чергу, теж небезпечно для навколишнього середовища.

Крім того, забруднювачі, на знешкодження яких йдуть величезні засоби, являють собою речовини, на які уже витрачена праця і які за рідкісним винятком можна було б використовувати в народному господарстві.

Для досягнення високих еколого-економічних результатів необхідно процес очищення шкідливих викидів сполучити з процесом утилізації уловлених речовин, що уможливить об'єднання першого напрямку з другим.

Другий напрямок – усунення самих причин забруднення, що вимагає розробки маловідходних, а в перспективі і безвідхідних технологій виробництва,

які дозволяли б комплексно використовувати вихідну сировину й утилізувати максимум шкідливих для біосфери речовин.

Однак далеко не для усіх виробництв знайдені прийнятні технікоекономічні рішення по різкому скороченню кількості відходів, що утворюються, і їх утилізації, тому на даний час доводиться працювати в обох зазначених напрямках.

Піклуючись про удосконалювання інженерної охорони навколишньої природного середовища, треба пам'ятати, що ніякі очисні спорудження і безвідхідні технології не зможуть відновити стійкість біосфери, якщо будуть перевищені припустимі (граничні) значення скорочення природних, не перетворених людиною природних систем, у чому виявляється чинність закону незамінності біосфери.

Таким порогом може виявитися використання більш 1% енергетики біосфери і глибоке перетворення більш 10% природних територій (правила одного і десяти відсотків). Тому технічні досягнення не знімають необхідності рішення проблем зміни пріоритетів суспільного розвитку, стабілізації народонаселення, створення достатнього числа заповідних територій і інших, розглянутих раніше.

Основними напрямками робіт в галузі захисту атмосфери від забруднення викидами автотранспорту є:

- створення і розширення виробництва автомобілів з високоекономічними і малотоксичними двигунами, у тому числі подальша дизелізація автомобілів;
- розвиток робіт зі створення і впровадження ефективних систем нейтралізації відпрацьованих газів;
- зниження токсичності моторних палив;
- розвиток робіт з раціональної організації руху автотранспорту в містах, удосконаленню дорожнього будівництва з метою забезпечення невпинного руху на автомагістралях.

					КР 122.6157м.ПЗ		
Зм.	Лист	№ докум	Підпис	Дата			
Студент		Потужний І.Р.			Загальні висновки	Літера	Лист 88 / Листов 1
Керівник		Гайда А.Ю.					
Рецензент		Новиков Ю.М.				ХФ НУК ім. адм. Макарова	
Зав. Каф		Гучек П.Й.					

ЗАГАЛЬНІ ВИСНОВКИ

1) Було розглянуто види веб-додатків, їх призначення та класифікацію. А також інструменти та технології їх створення.

2) Під час дослідження було з’ясовані основні вимоги до веб-додатків. Також проаналізовані основні методи тестування програмних засобів, для підвищення їх якості.

3) Досліджено найпоширеніші засоби перевірки ПЗ для кожного методу, проаналізована їх продуктивність на практиці у реально існуючому проекті, з’ясована їх швидкодія та особливості. Проведено порівняння, виділено переваги та недоліки методів і зроблений висновок щодо їх ефективності.

4) Спрогнозовано перспективи використання технологій безперервної інтеграції, як інструменту реалізації комплексних проектів автоматичної перевірки веб-додатків.

5) Обрані діючі інструменти і сервіси для створення автоматичної системи тестування програмних продуктів.

6) Створено алгоритм та блок-схему функціонування проекту системи тестування веб-додатків, розроблено програмний код із застосуванням поширених технологій та мов програмування для реалізації системи тестування в реально існуючому проекті.

					КР 122.6157м.ПЗ		
Зм.	Лист	№ докум	Підпис	Дата			
Студент		Потужний І.Р.			Додаток А	Літера	Лист 90 / Листов 2
Керівник		Гайда А.Ю.					
Рецензент		Новиков Ю.М.				ХФ НУК ім. адм. Макарова	
Зав. Каф		Гучек П.Й.					

ДОДАТОК А

Код тесту функції реєстрації власного магазину

Тест-кейс: RegisterStore.java

```
package TestCases;
import Pages.FRONTRegisterStore;
import Utilities.ReadProperties;
import com.github.javafaker.Faker;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import
org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
public class RegisterStore {
private static WebDriver driver;
private static FRONTRegisterStore frontRegisterStore;
@BeforeClass
public void setup() {
System.setProperty("webdriver.chrome.driver",
"src/test/chromedriver.exe");
ChromeOptions options = new ChromeOptions();
options.addArguments("disable-infobars");
driver = new ChromeDriver(options);
driver.manage().window().maximize();
}
@Test
```

```
    public void test() {
    driver.get(ReadProperties.getTestProperty("config_izz
i","front_stage_register_store"));
    frontRegisterStore = new FRONTRegisterStore(driver);
    frontRegisterStore.ShopName("SomeShop”);
    frontRegisterStore.ShopURL("http://mystore.ua");
    frontRegisterStore.CompanyName("company");
    frontRegisterStore.YourName("name");
    frontRegisterStore.PhoneNumber("0674563496");
    Faker faker = new Faker();
    String email = faker.internet().emailAddress();
    String pass = faker.internet().password();
    System.out.println(email +"\n"+ pass);
    frontRegisterStore.YourEMail(email);
    frontRegisterStore.YourPassword(pass);
    frontRegisterStore.RegistrationConfirmButton();
    WebDriverWait regStore = new WebDriverWait(driver,
10);
    regStore.until(ExpectedConditions.visibilityOfElement
Located(By.id("user-dropdown")));
    System.out.println("Store register");
    }
    @AfterTest
    public void exit(){
    driver.quit();
    }
    }
```

					КР 122.6157м.ПЗ		
Зм.	Лист	№ докум	Підпис	Дата			
Студент		Потужний І.Р.			Додаток Б	Літера / Лист / Листов	
Керівник		Гайда А.Ю.				93 / 4	
						ХФ НУК ім. адм. Макарова	
Рецензент		Новиков Ю.М.					
Зав. Каф		Гучек П.Й.					

ДОДАТОК Б

Код тесту у якому порівнюємо категорії

Тест-кейс: CompareCategories.java

```
package TestCases;
import Utilities.ReadProperties;
import org.openqa.selenium.By;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.testng.Assert;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
import java.util.List;
public class CompareCategories {
public static WebDriver driver;
@BeforeClass
public static void setup() {
System.setProperty("webdriver.chrome.driver","src/tes
t/chromedriver.exe");
ChromeOptions options = new ChromeOptions();
options.addArguments("disable-infobars");
options.addArguments("start-maximized");
driver = new ChromeDriver(options);
driver.get(ReadProperties.getTestProperty("config_izz
i","front_stage_sitemap_category"));
}
```

```
    public static Boolean IsElementVisible(By
icssSelector)
    {
    try
    {
    boolean iv =
driver.findElement(icssSelector).isDisplayed();
    if (iv == true) { return true; } else { return false;
}
    }
    catch (NoSuchElementException e) { return false; } //
if elements not found
    }
    public boolean isElementPresent(By by) {
    try {
    driver.findElement(by);
    return true;
    }
    catch (org.openqa.selenium.NoSuchElementException e)
{
    return false;
    }
    }
    @Test
    public void TestCompareCategories(){
    List<WebElement> sitemap =
driver.findElements(By.tagName("loc"));
    System.out.println("Categories on sitemap =
"+sitemap.size());
```

```
    driver.get(ReadProperties.getTestProperty("config_izz
i","adm_stage_category"));
    driver.findElement(By.id("loginform-
email")).sendKeys(ReadProperties.getTestProperty("config_c
redential","login_1"));
    driver.findElement(By.id("loginform-
password")).sendKeys(ReadProperties.getTestProperty("confi
g_credential","password_1"));
    driver.findElement(By.name("login-button")).click();
    try {
    Thread.sleep(4000);
    } catch (InterruptedException e) {
    e.printStackTrace();
    }
    WebElement                logOutButton                =
driver.findElement(By.id("side-menu"));
    if (logOutButton.isDisplayed()==true){
    System.out.println("Admin is logged in");
    } else {
    System.out.println("Admin is not logged in");
    }
    driver.findElement(By.id("categoryinternalsearch-
productcountfilter")).click();
    String                    catInAdm                    =
driver.findElement(By.cssSelector("#w1 > div > div.float-
right > div > b:nth-child(2)")).getText();
    System.out.println("Categories   on   admin   page   =
"+catInAdm);
    String new_catInAdm = catInAdm.replaceAll("\\,", "");
    int string_from_adm = Integer.parseInt(new_catInAdm);
```

```
    System.out.println("Categories on admin page to int =
"+string_from_adm);
    if (sitemap.size() == string_from_adm) {
    System.out.println("Number of categories from
sitemap.xmp "+sitemap.size()+" !!!equals!!!
"+string_from_adm+" number of categories from admin
page");
    }
    else {
    System.out.println("Number of categories from
sitemap.xmp "+sitemap.size()+" !!!not equal!!!
"+string_from_adm+" number of categories from admin
page");
    Assert.assertEquals(string_from_adm,sitemap.size());
    }
    }
    @AfterTest
    public void afterTest() {
    driver.quit();
    }
    }
```

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) https://ru.wikipedia.org/wiki/Веб-приложение
2) https://pidru4niki.com/1970070547797/informatika/zasobi_stvorennya_web-saytiv
3) https://www.helloworld.ru/texts/comp/web/asp/asp2/index.htm
4) AJAX: Вікіпедія – енциклопедія. https://uk.wikipedia.org/wiki/AJAX
5) Савин. Р. Тестирование Дот Ком, Р. Савин. – Дело, 2007. – 312 с JavaScript: Вікіпедія –енциклопедія. –: https://uk.wikipedia.org/wiki/JavaScriptbook
6) Тестування програмного забезпечення: Вікіпедія – енциклопедія: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення
7) DHTML: Вікіпедія –енциклопедія: https://uk.wikipedia.org/wiki/DHTML
8) XMLHttpRequest: Вікіпедія – енциклопедія: https://uk.wikipedia.org/wiki/XMLHttpRequest
9) https://habr.com/ru/post/279535/ - Тестирование. Фундаментальная теория
10) International Organization for Standardization, 1994.
11) Керування конфігурацією. Вікіпедія – вільнаенциклопедія:https://uk.wikipedia.org/wiki/Керування_конфігурацією
12) Labaka – Структура веб-приложения. В. А. Солонин. – 2011.
13) Wiki. Сравнение систем управления конфигураций.: Д.Р. Туляков. – 2016. – http://www.dtulyakov.ru/ci-cd-cm.html
14) Comparison of continuous integration software. Вікіпедія – енциклопедія:https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software
15) DOU. Тестирование. Фундаментальная теорія: Gennadii Mishchevskii. – 2015: https://dou.ua/forums/topic/13389/
16) Хабрахабр. Что такое Selenium? А.В. Баранцев. – 2012: https://habrahabr.ru/post/152653/

17) ORDINATUS. 5 инструментов непрерывной интеграции. А.Р Рубенов. – 2016: http://ordinatus.ru/5-instrumentov-nepreryvnoj-integracii/
18)
19) WebForMyself. Тестирование кода с PHPUnit:О.В. Тотко. – 2014: https://webformyself.com/testirovanie-koda-s-phpunit/
20) Automated Testing of Desktop. Web. Mobile: Ranorex Company. – http://www.ranorex.com/
21) ISO 8402:1994 Quality management and quality assurance. – Selenium / WebDriver Selenium – свободный ресурс для автоматизация веб-приложений через браузер: http://selenium2.ru/
22) PVS-Studio. TestComplete. Р.В. Воронов. – 2014: http://www.viva64.com/ru/t/0089/
23) HP QuickTest Professional. Вікіпедія –енциклопедія.: https://ru.wikipedia.org/wiki/HP_QuickTest_Professional
24) Test Ministry Of Testing – co-creating smarter testers: http://www.ministryoftesting.com/resources/softwaretesting-tools/
25) List of Testing Toolsguru99 – professional courses: http://www.guru99.com/list-of-testing-tools.html
26) 10 REGRESSION/FUNCTIONAL WEB TESTING TOOLS TECH BLOG. – http://www.hurricanesoftwares.com/10-regressionfunctional- web-testing-tools/
27) DevColibri. Тестирование с помощью TestNG в Java: А.Р. Барчук. – 2013: http://devcolibri.com/15285 Best Test Automation Tools automated-360 blog. – http://automated- 360.com/automation-tools/5-best-test-automation-tools
28) Category:Software testing tools Вікіпедія –енциклопедія: https://en.wikipedia.org/wiki/Category:Software_testing_tools
29) What is Selenium? Selenium HQ: http://docs.seleniumhq.org/
30) HP Unified Functional Testing software Web-site of Hewlett-Packard Company.: http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA4-8360ENW.pdf

31) Test Studio Web-site of Telerik. –: http://www.telerik.com/teststudio

a. TestComplete Platform: Testing for Desktop, Mobile, & Web Applications Web-site of Smartbear: http://smartbear.com/product/testcomplete/overview/