ННК “Інститут прикладного системного аналізу”
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

“___”_____2016 р.

Дипломна робота

першого (бакалаврського) **рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: : Розробка віртуальної лабораторії функціонально-логічного моделювання

Виконав (-ла): студент (-ка) IV курсу, групи ДА-22
(шифр групи)

Якимець Роман Вікторович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент , к. т. н. Кирюша Богдан Анатолійович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічного розділу проф., д.е.н. Семенченко Н.В. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль ст. викладач Британов О.А. _____
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2016 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

«___»_____2016 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту
Якимцю Роману Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка віртуальної лабораторії функціонально-логічного моделювання.

керівник проекту (роботи) Кирюша Богдан Анатолійович , к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 травня 2016 р. № 50-ст

2. Строк подання студентом проекту (роботи) 08.06.2016

3. Вихідні дані до проекту (роботи) :

- Отримати практичне уявлення про реалізацію віртуальної лабораторій функціонально логічного програмування у вигляді веб-додатка .
- Проаналізувати існуючі рішення.
- Розробити програмний продукт спираючись на проведену аналітичну роботу.

4. Зміст розрахунково-пояснювальної записки (перелік за8вдань, які потрібно розробити)

1. Дослідити принципи побудови веб-додатків для САПР.
2. Розглянути та дослідити існуючі лабораторії функціонально логічного моделювання.
3. Описати архітектуру розробленого рішення.
4. Розробити програмну реалізацію.
5. Описати додаток та продемонструвати результат його роботи.
6. Провести економічну оцінку даної розробки за допомогою функціонально-вартісного аналізу.
7. Зробити висновки по роботі.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Схема роботи клієнт-серверної моделі– плакат.
2. Приклад роботи програми – плакат.
3. Схема роботи веб-додатка – плакат.
4. Презентація у форматі Power Point.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економіка	Семенченко Н. В., професор, доцент економічних наук		
Основна частина			

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2016	
2	Збір інформації	15.02.2016	
3	Аналіз існуючих рішень	28.02.2016	
4	Вивчення варіантів реалізації та вибір варіанту для розробки	10.03.2016	
5	Проектування архітектури	20.03.2016	
6	Розробка серверної частини	25.03.2016	
7	Розробка інтерфейсу користувача	25.04.2016	
8	Тестування програми	20.05.2016	
9	Оформлення дипломної роботи	31.05.2016	
10	Отримання допуску до захисту та подача роботи в ДЕК	08.06.2016	

Студент _____ Р.В.Якимець
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ Б.А.Кирюша
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

до бакалаврської дипломної роботи Якимця Романа Вікторовича
на тему : «Розробка віртуальної лабораторії функціонально-логічного моделювання»

Дипломна робота присвячена розробці веб-додатку , для функціонально-логічного моделювання прямо в браузері на мові опису апаратури Verilog базуючись на клієнт-серверній архітектурі. Була розглянута клієнт-серверна архітектура та описана побудова на ній даного додатку.

Були досліджені принципи побудови веб-додатків для САПР та веб-додатків в цілому. Також були проаналізовані уже існуючі рішення з подібним функціоналом та проведено їх порівняння.
Загальний обсяг 84 сторінки , 17 рисунків , 7 таблиць , 12 бібліографічних найменувань.
Ключові слова : Verilog , онлайн симулятор , функціонально-логічне моделювання , веб-додаток , САПР.

АННОТАЦИЯ

к бакалаврской дипломной работе Якимца Романа Викторовича
на тему: «Разработка виртуальной лаборатории функционально-логического моделирования»

Дипломная работа посвящена разработке веб-приложения, для функционально-логического моделирования прямо в браузере на языке описания аппаратуры Verilog основываясь на клиент-серверной архитектуре. Была рассмотрена клиент-серверная архитектура и описано построение на ней данного приложения.

Были исследованы принципы построения веб-приложений для САПР и веб-приложений в целом. Также были проанализированы уже существующие решения с подобным функционалом и проведено их сравнение.

Общий объем 84 страницы, 17 рисунков, 7 таблиц, 12 библиографических наименований.

Ключевые слова: Verilog, онлайн симулятор, функционально-логическое моделирование, веб-приложение , САПР.

ANNOTATION

a bachelor thesis work Yakymets Roman Viktorovich

on the theme: "Development of a virtual laboratory functional-logical modeling."

Degree work is dedicated to the development of web applications for functional logic simulation directly in the browser on the Verilog hardware description language based on client-server architecture. Client-server architecture and described the construction on it of the application was considered.

Principles of web applications for CAD web-based applications and web-based applications in general have been investigated. Also we analyzed existing solutions with similar functionality and compared there.

The total amount of work 84 pages, 17 figures, 7 table, 12 bibliographical references. Keywords: Verilog, online simulator, functional and logical modeling, web-based application , CAD.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ , СКОРОЧЕНЬ І ТЕРМІНІВ

САПР – система автоматизованого проектування

HTTP – HyperText Transfer Protocol — «протокол передачи гипертекста»

JSP – Java Server Pages

AJAX – Asynchronous JavaScript And XML

XML – Extensible Markup Language

DOM – Document Object Model

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

DHTML – Dynamic HTML

ВСТУП

Останнім часом спостерігається тенденція до створення веб-додатків на основі різноманітних розробок . Це дозволяє полегшити доступ до цих розробок та налагоджує командну роботу при роботі з ними. І правда , доволі зручно мати доступ до різноманітних речей встановивши лише веб-браузер. Розміщення даних на сервері дає змогу дистанційно з різних електронних пристроїв працювати над певним проектом.

Майже всі розробники рано чи пізно стикаються з необхідністю запустити або швидко перевірити якийсь код, але не всі з них знають, що для такого простого завдання зовсім необов'язково запускати важкі десктопні IDE або прикладні компілятори. Досить скористатися онлайн інструментами, які дозволяють все зробити набагато швидше. Розробка віртуальної лабораторії функціонально-логічного моделювання дає змогу , дистанційно , без встановлення додаткових модулів на власний електронний пристрій , проводити Verilog симуляції. Однією з основних перешкод імплементації подібних сервісів являється збільшення навантаження на систему при їх використанні проте при зменшенні функціоналу вони знаходять своє місце на ринку. Також в зв’язку з розвитком технологій і ростом обчислювальної потужності електронних пристроїв дана концепція розробки прямо в веб-браузері виглядає доволі перспективною.

Завданням даних віртуальних лабораторій являється надання змоги розробникам швидко з будь-якого прибору в якому установлений веб-браузер получити можливість провести симуляцію на мові опису апаратури Verilog або ж дистанційно працювати в команді над одним проектом .

Мета цієї роботи дослідити існуючі лабораторії функціонально-логічного моделювання та реалізувати власну. Також ставиться завдання проаналізувати отриману реалізацію.

1. ПРИНЦИПИ ПОБУДОВИ ВЕБ-ДОДАТКІВ ДЛЯ САПР.

1.1 Принципи побудови веб додатків.

На даний момент існують і успішно застосовуються різні види технологій побудови Web-додатків. Всі такі додатки мають спільну мету - реалізацію бізнес - логіки на стороні сервера і генерацію коду для клієнта. Також у всіх цих додатків однакова архітектура взаємодії сервера і клієнта та загальний протокол взаємодії - HTTP. Загальна логіка роботи кієнт-серверних програми представлена на рисунку 1.

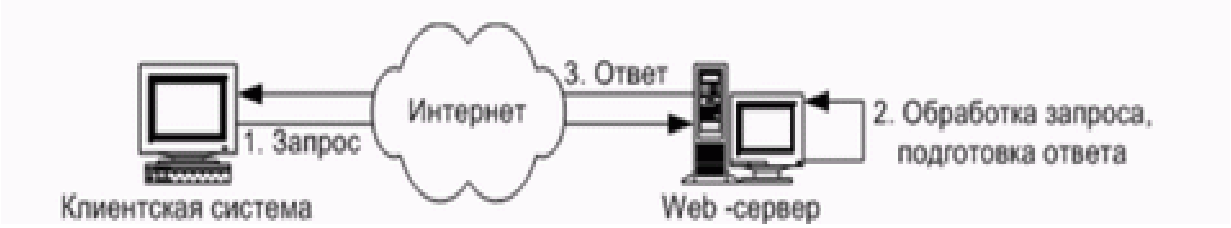

Рисунок 1– Архітектура роботи клієнт-серверних додатків [3]

Як видно з малюнка, робота серверних додатків відбувається в три основних етапи:

1. Запит. Клієнт, використовуючи web - браузер, ініціює запит до сервера.
2. Обробка запиту, підготовка відповіді. Після отримання запиту web - сервер проводить обробку запитуваного ресурсу. У разі, якщо запитується статичний ресурс, такий як HTML сторінка, малюнок, документ, ця інформація форматується для протоколу HTTP і передається клієнтові в якості відповідь. Якщо ж запитується динамічний ресурс, запит передається на обробку відповідного контейнеру web - додатків, де і відбувається подальша робота.
3. Після формування, дані передаються клієнту за допомогою протоколу HTTP в якості відповіді. Відповідь містить дані (зазвичай

HTML код, або двійкові дані), а також додаткові параметри, що передаються в заголовках HTTP відповіді.

Робота додатків серверної сторони завжди відбувається за описаним вище сценарієм. Очевидно, що такий підхід створює складності при створенні web-додатків, основною яких є відсутність стану у web - додатка (так зване stateless programming). Це означає, що додаток працює виключно в режимі запит-відповідь, не маючи даних про попередні кроки користувача або будь-якої іншої постійної інформації. Для вирішення цієї проблеми застосовується поняття користувальницької сесії, яка дозволяє зберігати дані на сервері протягом сеансу роботи користувача.

Однак , наявністю сесій , складності при створенні web - додатків повністю не усуваються. Чим більше можливостей надає платформа для додатків серверної сторони в подоланні цих труднощів, тим швидше і ефективніше може вестися розробка. Далі будуть розглянуті різні підходи до створення клієнт-серверних додатків, їхні переваги й недоліки, а також розглянуті конкретні платформи.

Вимоги до клієнт-серверних додатків.

При розгляді платформ для створення додатків необхідно виділити два основних існуючих підходу:

1. Безпосередня обробка запитів і формування відповідей.
2. Вбудовування програмного коду в шаблони HTML сторінок.

Перший підхід надає найбільші можливості по управлінню обробкою і підвищенню продуктивності. Він передбачає передачу всіх даних про запит безпосередньо виконуваного коду, який може як сформувати відповідь зі сторінкою для користувача, так і відкрити на передачу потік двійкових даних, наприклад для передачі зображення. Однак при такому підході всі дані для передачі формуються програмним шляхом, що уповільнює розробку простих

сторінок і ускладнює взаємодію між верстальником і програмістом. Прикладами цього підходу служать технології CGI, Java Servlets.

Другий підхід використовує шаблони сторінок користувача, оформлені особливим чином, що дозволяє вставляти в них ділянки програмного коду. Цей підхід особливо ефективний при створенні простих додатків, основна інформація в яких статична, а динамічна інформація може бути згенерована простими програмними конструкціями. При розробці складних програмних систем цей варіант ускладнює взаємодію між компонентами і ускладнює реалізацію складної архітектури. Також він менш ефективний за продуктивністю і призводить до обмеження можливостей реалізації складних сторінок. Прикладами цього підходу служать дуже популярні на даний момент технології PHP, ASP, JSP.

Крім різного підходу до генерації сторінок платформи розробки в різному ступені задовольняють сучасним вимогам, що висуваються при створенні складних Web систем. Найбільш важливі з цих вимог, наявність яких робить систему привабливою для використання, наведені нижче:

- Платформна незалежність.
- Мова реалізації.
- Продуктивність, масштабованість.
- Можливості розширення і інтеграції.
- Простота використання, наявність засобів розробки.
- Наявність необхідних програмних бібліотек.

Отже, ми визначили ряд вимог, необхідних для сучасної платформи розробки. Нижче розглядаються найбільш популярні на даний момент платформи, їх особливості, а також оцінка з точки зору наведених критеріїв.

Швидкий розвиток інформаційного Web - середовища призвів до того, що вимоги до Web-додатків суттєво змінилися. Зокрема спостерігається тенденція

до створення багатих Web-додатків, тобто додатків, інтерфейс яких надає можливості, що не відрізняються від можливостей звичайного додатку, який призначений для настільної системі. Але при роботі програм, що підтримують мережеву взаємодію, усунути затримку відповіді, пов'язану з передачею даних через мережу Інтернет, принципово неможливо. Пом'якшити негативний ефект від затримки даних дозволяє технологія Ajax. Але застосування цієї технології повністю змінило структуру та принципи роботи Web-додатків. В сучасних мережевих програмах все більше функцій виконується на клієнтському боці, тому обсяг коду клієнтської частини Web-додатку суттєво збільшується і робота над нею виконується групою розробників. В результаті виявилося, що мова JavaScript, яка застосовується для написання Ajax-додатків, має специфічне застосування і не відповідає вимогам до інструментальних засобів розробки та налагодження програм.

В цьому розділі запропоновано підхід до створення Ajax-додатків, згідно якому для написання коду клієнтської частини програми разом з JavaScript-сценаріями. Завдяки взаємодії між JavaScript та Java стає можливим розділити задачі, що стоять перед додатком. Застосовуючи мову Java для написання коду, що реалізує складні алгоритми, можна застосувати численні інструментальні засоби для розробки та налагодження програм. При цьому на долю JavaScript залишаються незначні по об'єму фрагменти коду, які динамічно змінюють вміст сторінки, що можуть бути написані та налагоджені без застосування спеціальних інструментальних засобів розробки та налагодження програм.

Запропонований підхід реалізований у вигляді набору базових засобів для написання Ajax-додатків з подальшим створенням віртуальної лабораторії функціонально-логічного моделювання.

1.1.1 Типи додатків

Розширення-(англ. extension) можуть бути використані для зміни поведінки наявних функцій або для додавання нових можливостей. Розширення особливо популярні у Firefox, оскільки розробники Mozilla створювали браузер, як досить мінімалістичну програму, що мало запобігти росту кількості помилок і запобігти громіздкості програми, зберігаючи при цьому високий степінь розширення, таким чином індивідуальні користувачі зможуть додати функції, яким вони віддають перевагу.

Розширення технологій

- CSS (Cascading Style Sheets)
- DOM (Document Object Model) - використовується для зміни XUL в реальному часі або зміни вже завантаженого HTML
- JavaScript - основна мова браузера Mozilla
- XPCOM (кросплатформлена модель компонентних об’єктів)
- XPConnect
- XPI
- XUL (XML-мова інтерфейсу користувача) - використовується для визначення інтерфейсу користувача, та взаємодії з ним.

Додавання можливостей

Розширення, зазвичай, використовуються, щоб додати нові можливості до програми. Приклади можливостей, які можуть бути додані за допомогою розширень: читачі RSS, менеджери закладок, пенали, клієнтські програми для окремих веб-сайтів, менеджери протоколу FTP, електронна пошта, жести мишки, перемикання проксі-серверів, засоби веб-розробки, тощо. Багато розширень Firefox виконують функції, які раніше раніше були частиною Mozilla Suite, наприклад, ChatZilla, клієнт IRC та календар.

Зміна зовнішнього вигляду веб-сторінок для користувача

Багато розширень можуть змінювати вміст веб-сторінки при її відтворенні на екрані. Наприклад, розширення Adblock може запобігти завантаженню рекламних зображень. Інше популярне розширення Greasemonkey, дозволяє користувачеві встановити скрипти, які змінюють цільові підмножини сторінок на ходу, у спосіб, що є програмним розширенням таблиць каскадних стилів.

1.1.2 Технологія Web 2.0

Саме поява та розвиток Web 2.0 дозволили створення динамічних веб-додатків і надали можливість створення віртуальної лабораторії функціонально-логічного програмування . Появу терміну Web 2.0 пов’язують зі статтею Тіма О’Реллі від 30 вересня 2005 року, в якій автор прив’язав появу великої кількості сайтів, об’єднаних деякими загальними принципами, із загальною тенденцією розвитку інтернет-спільноти, і назвав це явище Web 2.0, як противага «старому» Web 1.0.

Незважаючи на те, що значення цього терміну до цього часу викликає безліч суперечок, ті науковці, що визнають існування Web 2.0, виділяють декілька основних аспектів цього явища — Web-служби, Ajax, Mash-up, Теги і т.п.

Web-служби — програми, взаємодія з якими здійснюється через Web (протокол HTTP) а обмін даними відбувається в форматі XML, JSON та подібних. В результаті ПЗ може використовувати Web-служби замість самостійно реалізовувати потрібні функціональні можливості.

Ajax або Asynchronous JavaScript and XML — підхід до побудови Web-програм, при якому Web-сторіка асинхронно та без перезавантаження отримує потрібні користувачу дані з сервера. Дуже часто Ajax вважають синонімом Web 2.0, але це абсолютно не вірно — Web 2.0 не прив’язаний до будь-яких технологій і є скоріше тенденцією розвитку Інтернету.[5]

Mash-up — сервіс, що дозволяє використовувати інформацію з інших сервісів як джерело інформації, пропонуючи користувачу нові функціональні можливості для роботи. В результаті такий сервіс може стати новим джерелом інформації для інших mash-up сервісів. Виникає мережа залежних один від одного сервісів, інтегрованих один з одним.

Теги — ключові слова, що описують певний об’єкт, або відносять його до певної категорії. Це мітки, що надаються об’єкту, щоб визначити його місце серед інших об’єктів. Поява і швидке розповсюдження блогів, що активно використовують теги, також вписується концепцію Web 2.0

Багаті Web-програми — програми, що мають функціональність та можливості традиційних програм, але працюють в браузері і активно взаємодіють з сервером. Завдяки цьому створюється система, що дозволяє виконувати роботу, пов’язану з створенням та обробкою інформації більш ефективною.

Інтерфейс користувача таких програм більше нагадує інтерфейс класичних програм ніж web-програм тому ефективно використовувати такі програми можуть навіть ті користувачі, що мають мінімальні знання про Інтернет.

Технологія Web 2.0 включає в себе:

- синдикацію
- протоколи передачі даних
- браузери з плагінами та розширеннями
- клієнтське ПЗ
- Типовий Web 2.0 сайт використовує такі технології:
- Cascading Style Sheets — розділення вмісту та оформлення

1.2 Web – технології для побудови веб-додатків.

1.2.1 Базові технології Web

HTML — стандартна мова розмітки документів для Web, де всі Web-сторінки створюються за допомогою HTML (або XHTML). Мова HTML інтерпретується браузером у вигляді документу, зручному для людини.

HTML створювався в 1991-1992 роках як мова для обміну науковою та технічною документацією, яка зручна для людей, що не є спеціалістами з верстки. Вона успішно мінімізує проблеми зі складністю SGML шляхом визначення невеликої кількості структурних та семантичних елементів (які розмічаються тегами), які використовуються для створення простих, але гарно оформлених документів. Також, крім спрощення структури документу, у HTML міститься підтримка гіпертексту. Мультимедійні можливості були додані пізніше.

Текстові документи, які містять код на мові HTML, обробляються спеціальними програмами, які відображають документ у форматованому вигляді. Такі програми, що називаються браузерами, забезпечують зручний графічний інтерфейс для взаємодії користувача із сервером — запит Web-сторінок, їх відображення та відправлення введених користувачем даних на сервер.

Від початку HTML був спроектований і створений як засіб структурування та форматування документів, без їх прив’язки до засобів відображення. Але сучасні застосування HTML далекі від його початкових задач — додані мультимедійні можливості, з’явилися засоби для створення складних графічних оформлень, додана можливості підключення плагінів та розширень.

Для створення динамічних сторінок було розроблений цілий ряд технологій — JavaScript, Java Апплети, Adobe Flash, Microsoft Silverlight.

Реалізації деяких з них інтегровані в браузери (JavaScript), для роботи з іншими потрібно підключати спеціальні плагіни (доступні безкоштовно на Web-сайтах розробників або поставляються разом з операційними системами чи браузерами).

В середині 90х років розгорнулось боротьба між розробниками найбільш популярних (на той час) браузерів — Netscape Navigator та Microsoft Internet Explorer за ринок інтернет-браузерів. Основний спосіб боротьби — розробка та впровадження нових технологій, що були не сумісні з іншими браузерами. В результаті навіть на сьогоднішній день не вдалося досягти повної сумісності між усіма браузерами, хоча їх розробники та консорціум W3C, який займається стандартизацією Web-технологій, докладають максимум зусиль для цього.

З іншого боку, в результаті цієї боротьби, з’явився ряд технологій, що займають ключову роль в розвитку сучасного Web, серед них — JavaScript та Ajax. Зараз важко знайти сайт, побудований згідно принципів Web 2.0, який би не використовував Ajax або JavaScript.

1.2.2 Загальні відомості про Ajax

Ajax — група методів Web-розробки, що використовуються для створення Web-програм з багатими можливостями та мережевою взаємодією, що базується на «фоновому» обміні даними браузера з Web-сервером. В результаті сторінка не перезавантажується повністю і Web-програма стає швидкою та зручною.

Ajax це не самостійна технологія, а скоріше концепція використання декількох суміжних технологій. Ajax базується на двох основних принципах: використання технології взаємодії із сервером за допомогою JavaScript об’єкта XMLHttpRequest без перезавантаження усієї сторінки використання DHTML для динамічної зміни вмісту сторінки та реагування на дії користувача

Для передачі даних від сервера до клієнта використовуються формати

XML або JSON. Класична модель web-програм пов’язана не лише з використанням базових web-технологій, а і з специфічним способом роботи з web-програмою, при якому web-браузер є лише низькорівневим терміналом. Він не має інформації про те, який етап роботи виконується користувачем. Він лише отримує готову сторінку в форматі HTML і відображає її користувачу.

У web-програмах, побудованих за допомогою технології Ajax, частина функціональних можливостей переноситься з сервера на клієнт. На деякі дії користувача така web-програма може реагувати самостійно. Якщо наявних можливостей не вистачає для виконання ініційованих користувачем дій то відбувається взаємодія із сервером, при цьому користувач може виконувати інші дії. Оскільки HTML документ присутній на стороні клієнта протягом всього часу роботи з web-програмою, то він здатний зберігати всю інформацію про її стан.

Технологія динамічного завантаження вмісту існувала і раніше — за допомогою атрибуту src можна було завантажити зовнішній сценарій JavaScript, який змінить поточну сторінку. Але цей метод не є дуже вдалим через обмеження атрибуту src та додатковому навантаженні на сервер, бо він має виконати додаткові дії для генерації спеціального сценарію JavaScript, що містить інструкцію, як модифікувати поточну сторінку в нову.

Засоби, що використовуються в рамках технології Ajax не єдиний спосіб забезпечити асинхронний обмін даними з сервером. Наприклад Macromedia Flash (починаючи з 4ї версії) може завантажувати дані в форматі XML або CSV з серверу без перезавантаження сторінки. Але цю технологію не можна використовувати для створення багатих web-програм бо вона в основному використовується для роботи з мультимедійними даними і малопридатна для динамічної зміни вмісту сторінки.

Пізніше Microsoft створила об’єкт XMLHttpRequest в Internet Explorer 5, що і став основою Ajax.

Переваги Ajax:

- Створення web-програм, що мають інтерфейс та багаті можливості, подібні до звичайних програм — при цьому, завдяки активній взаємодії з сервером, web-програм мають значні переваги над звичайними програмами.
- Економія трафіку — замість завантаження усієї сторінки достатньо завантажити відносно невелику частину, що змінилася.
- Зменшення навантаження на сервер — серверу не потрібно кожного разу генерувати усю сторінки, а лише ту частину, що змінилася.
- Прискорення реакції інтерфейсу — оскільки завантажується лише частина сторінки то користувач бачить результат своїх дій швидше.

Недоліки Ajax:

- Відсутня інтеграція із стандартними інструментами браузера — не працює кнопка «Назад», сторінку, згенеровану за допомогою Ajax не можна додати в закладки.
- Проблема з індексуванням сайту пошуковими роботами — у них відсутня підтримка JavaScript.
- Використання JavaScript та DOM, що мають різну реалізацію в різних браузерах та навіть різних версіях браузерів.

1.2.3 HTML та XHTML

Використано для отримання веб-інтерфейсу що виводить аналізовані данні.

HTML — Мова розмітки гіпертекстових документів —стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML є похідною мовою від SGML, успадкувавши від неї визначення типу документу та ідеологію структурної розмітки тексту.

Попри те, що HTML — штучна комп'ютерна мова, вона не є мовою програмування.

HTML разом із каскадними таблицями стилів та вбудованими скриптами — це три основні технології побудови веб-сторінок.

HTML впроваджує засоби для:

- створення структурованого документу шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації із Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об'єктів до тексту.

XHTML (Extensible Hypertext Markup Language) — мова розмітки, що має таку саму виразну силу як і HTML але відповідає синтаксичним правилам XML.

В той час як HTML побудовано на основі правил SGML, XHTML побудовано на основі правил XML, суворішої підмножини правил SGML. Оскільки XHTML документи мають бути коректними XML документами, їх обробку можна здійснювати стандартними інструментами обробки XML документів на відміну від HTML, який вимагає порівняно складніших, важчих і повільніших синтаксичних аналізаторів. XHTML можна розглядати як, багато в чому, перетин HTML і XML, оскільки цей стандарт є переформуванням HTML засобами XML. XHTML 1.0 став рекомендацією консорціуму W3C 26 січня 2000. XHTML 1.1 став рекомендацією W3C 31 травня 2001.[6]

XHTML 1.0 є «реформуванням трьох типів документів стандарту HTML 4 засобами XML 1.0».[1] World Wide Web Consortium (W3C) також продовжує підтримку Рекомендації HTML 4.01 та активну роботу над специфікаціями

стандартів HTML5 і XHTML5. В поточному документі Рекомендацій XHTML 1.0, який було опубліковано та переглянуто до серпня 2002 року, W3C зазначив, що, "Сімейство XHTML є наступним кроком в еволюції Інтернету. Шляхом переходу на XHTML сьогодні, розробники контенту можуть увійти в світ XML з усіма супутніми перевагами, залишаючись впевненими в зворотній та майбутній сумісності їхнього контенту.

Проте в 2004 році, незалежно від W3C було створено Робочу групу з технологій застосування гіпертексту у Вебі (WHATWG), для роботи по вдосконаленню звичайного HTML не заснованого на XHTML. Більшість великих виробників браузерів не бажали реалізовувати функції з нових проектів стандартів W3C XHTML оскільки вважали, що вони не відповідають сучасним потребам розвитку Інтернету, а W3C захопився формалізмом XML і не реагує на реальні вимоги виробників. Apple, Mozilla та Opera сформували робочу групу WHATWG, яка почала працювати над стандартом HTML5, який допускав, але не вимагав застосування XML. У 2007 році, Робоча група W3C HTML проголосувала за офіційне визнання HTML5 і роботу над ним як наступне покоління стандарту HTML.[3] У 2009 році консорціум W3C дозволив добігти до кінця терміну дії Статуту Робочої групи XHTML 2, визнавши, що HTML 5 буде єдиним наступним поколінням стандарту HTML, як з XML, так і не-XML серіалізацію.

XHTML був розроблений з метою зробити HTML більш розширюваним і підвищити сумісність з іншими форматами даних. HTML 4 побудований на основі та є застосуванням стандартної узагальненої мови розмітки (SGML), однак специфікація SGML складна, і як веб-браузери, так і Рекомендація HTML 4 не були повністю сумісними з нею. Стандарт XML, затверджений в 1998 році, пропонував простіший формат даних, ближче за духом до HTML 4.[7] Існували сподівання, що за допомогою переходу на формат XML, HTML стане сумісним із загальними інструментами XML; а проксі сервери зможуть перетворювати документи, у разі необхідності, для пристроїв з обмеженими можливостями,

таких як мобільні телефони. Завдяки використанню просторів імен, XHTML документи могли б включати фрагменти інших, основаних на XML, мов, таких як Scalable Vector Graphics і MathML. Нарешті, відновлення роботи дала б можливість розділити HTML на компоненти для повторного використання (XHTML Модулі) і очистити неохайні частини мови.

1.2.4 CSS

Каскадні таблиці стилів (англ. Cascading Style Sheets або скорочено CSS) — спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів. Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі — сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо. [6]

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їхньої візуальної презентації.

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент,

наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.)

Один і той самий HTML або XML документ може бути відображений порізному залежно від використаного CSS. Стилі для відображення сторінки можуть бути:

Стилі автора (інформація надана автором сторінки):

- зовнішні таблиці стилів (англ. stylesheet), найчастіше окремий файл або файли .css;
- внутрішні таблиці стилів, включені як частина документу або блоку;
- стилі для окремого елементу.

Стилі користувача

- локальний .css-файл, вказаний користувачем для використання на сторінках і вказаний в налаштуваннях браузера (наприклад Opera)

Стилі переглядача (браузера)

- стандартний стиль переглядача, наприклад стандартні стилі для елементів, визначені браузером, використовуються коли немає інформації про стиль елемента або вона неповна.

Стандарт CSS визначає порядок та діапазон застосування стилів, те, в якій послідовності і для яких елементів застосовуються стилі. Таким чином,

використовується принцип каскадності, коли для елементів вказується лише та інформація про стилі, що змінилася або не визначена загальнішими стилями.

Переваги

- Інформація про стиль для усього сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок;
- Різна інформація про стилі для різних типів користувачів: наприклад великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв;
- Сторінки зменшуються в об'ємі та стають більш структурованими, оскільки інформація про стилі відділена від тексту та має певні правила застосування і сторінка побудована з урахуванням їх;
- Прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі. Досягається за рахунок того, що сучасні браузери здатні кешувати (запам'ятовувати) інформацію про стилі і використовувати для всіх сторінок, а не завантажувати для кожної.

1.2.5 JavaScript та JQuery

JavaScript – прототипна-орієнтована сценарна мова програмування. Є реалізацією мови ECMAScript (стандарт ECMA-262).

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипна парадигма програмування, функції як об'єкти першого класу.

На JavaScript вплинули багато мов, при розробці була мета зробити мову схожим на Java, але при цьому легким для використання непрограмістів.

Мовою JavaScript не володіє будь-яка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних у веб-розробці.

Назва «JavaScript» є зареєстрованим товарним знаком компанії Oracle Corporation. Назва "ECMAScript" не є торговим знаком будь-яких компаній.

JavaScript є об'єктно-орієнтованою мовою, але використовне в мові прототипування обумовлює відмінності в роботі з об'єктами в порівнянні з традиційними клас-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, - функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання - що додає мові додаткову гнучкість.

Незважаючи на схожий з Сі синтаксис, JavaScript у порівнянні з мовою Сі має корінні відмінності:

- об'єкти, з можливістю інтроспекції;
- функції як об'єкти першого класу;
- автоматичне приведення типів;
- автоматична Збірка сміття;
- анонімні функції.

У мові відсутні такі корисні речі , як:

- модульна система: JavaScript не надає можливості управляти залежностями і ізоляцією областей видимості;
- стандартна бібліотека: зокрема, відсутній інтерфейс програмування додатків по роботі з файловою системою, управлінню потоками
- введення-виведення, базових типів для бінарних даних;
- стандартні інтерфейси до веб-серверів і баз даних;

- система управління пакетами, яка б відстежувала залежності і автоматично встановлювала їх

jQuery — популярна JavaScript-бібліотека з відкритим сирцевим кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом (John Resig). Згідно з дослідженнями організації W3Techs, jQuery використовується понад половиною від мільйона найвідвідуваніших сайтів.[2] jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

jQuery є вільним програмним забезпеченням під ліцензією MIT (до вересня 2012 було подвійне ліцензування під MIT та GNU General Public License другої версії).

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних веб-сторінок.

Основне завдання jQuery — це надавати розробнику легкий та гнучкий інструментарій кросбраузерної адресації DOM об'єктів за допомогою CSS та XPath селекторів. Також даний фреймворк надає інтерфейси для Ajaxзастосунків, обробників подій і простої анімації.

Принцип роботи jQuery полягає в використанні класу (функції), який при звертанні до нього повертає сам себе. Таким чином, це дозволяє будувати послідовний ланцюг методів.

1.2.6 Аналіз механізмів вазаємодії у Web 2.0

В попередньому розділі була описана проблема розробки та налагодження web-програм, пов’язана з специфікою мови JavaScript. За весь час її існування було представлено декілька способів вирішення проблем несумісності:

Базовий набір засобів JavaScript — проблеми несумісності вирішуються розробниками базового набору засобів, але для розробки сценарію JavaScript потрібно використовувати певний рівень абстракції а не оригінальну мову.

Приклади — jQuery, Prototype, MooTools.

Недоліком цього способу є обмеження базових наборів засобів та їх не універсальність, хоча для деяких задач це може бути вдалим рішенням.

Також, через використання абстракції та врахування недоліків різних двигунців, у JavaScript сценаріїв, написаних за допомогою базових наборів засобів, збільшується час виконання та навантаження на комп’ютер (у порівнянні з використанням «чистого» JavaScript).

Базовий набір засобів JavaScript з використанням інших мов програмування — подібне до попереднього, але є і відмінності — для отримання сумісного JavaScript-коду необхідно використовувати іншу, більш досконалу, мову програмування (найчастіше це Java) і за допомогою спеціального базового набору засобів генерувати JavaScript-код.

Недоліки подібні до попереднього методу, хоча деякі компанії успішно його використовують. Приклади — Google Web Toolkit та сервіси, побудовані за його допомогою — Google Mail, Google Maps та інші.

Розробка подібного базового набору є доволі складною справою (потрібно знати недоліки та особливості обох мов а також особливості та недоліки різних двигунців), а наявні базові набори не є універсальними (а досить часто навіть специфічними) тому цей метод не є оптимальним.

Використовувати інші Web-технології (а JavaScript використовувати лише для зв’язку цих технологій з HTML). Такими технологіями можуть бути Adobe Flash, Microsoft Silverlight, Java Апплети та інші.

1.2.7 Класичний механізм взаємодії у Web

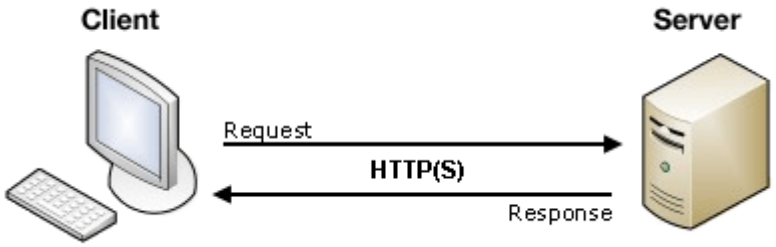

Рисунок 2– Класичний механізм взаємодії у Web.

HTTP — протокол передачі даних, що використовується в комп'ютерних мережах, належить до протоколів моделі OSI 7-го програмного рівня.

Основним призначенням протоколу HTTP є передача веб-сторінок (текстових файлів з розміткою HTML), хоча за допомогою його можна передавати й інші файли, як пов'язані з веб-сторінками (зображення і додатки), так і не пов’язані з ними (у цьому HTTP конкурує з складнішим FTP).

HTTP припускає, що клієнтська програма — веб-браузер — здатна відображати гіпертекстові веб-сторінки і файли інших типів в зручній для користувача формі. Для правильного відображення HTTP дозволяє клієнтові дізнатися мову і кодування веб-сторінки і/або запитати версію сторінки в потрібних мові/кодуванні, використовуючи позначення із стандарту MIME.

Класичний механізм взаємодії у Web відбувається так: браузер генерує HTTP запит і відправляє його на сервер. Сервер оброблює запит і відправляє відповідь клієнту у вигляді готової HTML сторінки, яку браузер показує користувачу. Для кожного обміну даними між сервером та клієнтом потрібен окремий запит (перезавантаження сторінки).

Є два основні види запитів до сервера — GET та POST.

З початку GET був єдиним способом передачі даних від клієнта до сервера. Дані від клієнта до сервера передаються у вигляді параметрів адреси Згідно стандарту HTTP запити типу GET вважаються «безпечними» —

багаторазове повторення одного і того ж запиту призводить до одного і того ж результату (при умові, що сам ресурс не змінився за час між запитами). Це дозволяє кешувати відповіді на HTTP запити з типом GET.

За допомогою GET не можна передавати великі об’єми даних та файли (в браузерах, проксі-серверах та web-серверах є ліміти на довжину адреси, наприклад в браузерів Microsoft Internet Explorer це 1Кб).

Використання GET є небезпечним для відправлення поролів та іншої конфіденційної інформації — вона буде присутня в адресі у відкритому вигляд.

В 1996 з’явилася специфікація HTTP 1.0, що містила новий механізм запиту до сервера — POST. Дані від клієнта до сервера передаються в тілі запиту і, при необхідності, можуть бути зашифрованими. На відміну від запиту з типом GET, запити з типом POST вважаються «небезпечними» — багатократне повторення одних і тих же запитів з типом POST може давати різні результати.

Також за допомогою POST запиту можлива передача файлів від клієнта до сервера.

Існують також інші методи доступів, але вони мають специфічне застосування:

- OPTIONS — повертає методи HTTP, які підтримуються сервером. Використовується для визначення можливостей Web-сервера.
- HEAD — аналогічний методу GET, єдина різниця — у відповіді сервера відсутнє тіло. Використовується для отримання мета-даних, що задаються в заголовку відповіді, без відправлення всього вмісту.
- PUT — завантажує вказаний ресурс на сервер.
- DELETE — видаляє вказаний ресурс.
- TRACE — повертає отриману відповідь так, що клієнт може побачити, що проміжні сервери додали чи модифікували в запиті.
- CONNECT — використовується разом з proxy-сервером, які можуть динамічно переключатися в тунельний режим SSL.

Переваги класичного механізму доступу до Web — підтримка будь-яким HTTP клієнтом (браузером, роботом пошукової системи і т.п.).

Недоліки цього механізму — навіть при незначній зміні сторінки потрібно повністю завантажувати всю сторінку, що негативно впливає як на швидкості та комфорт при роботі з Web-програмою, так і на збільшення трафіку між сервером та клієнтом.

1.2.8 Взаємодія у Web за допомогою Ajax

В цьому механізмі доступу з'єднуючою ланкою між сервером та сторінкою є JavaScript-об'єкт XMLHttpRequest. В різних двигунцях та їх версіях він реалізований по різному тому потрібно використовувати спеціальну функцію, яка враховує всі можливі варіанти.

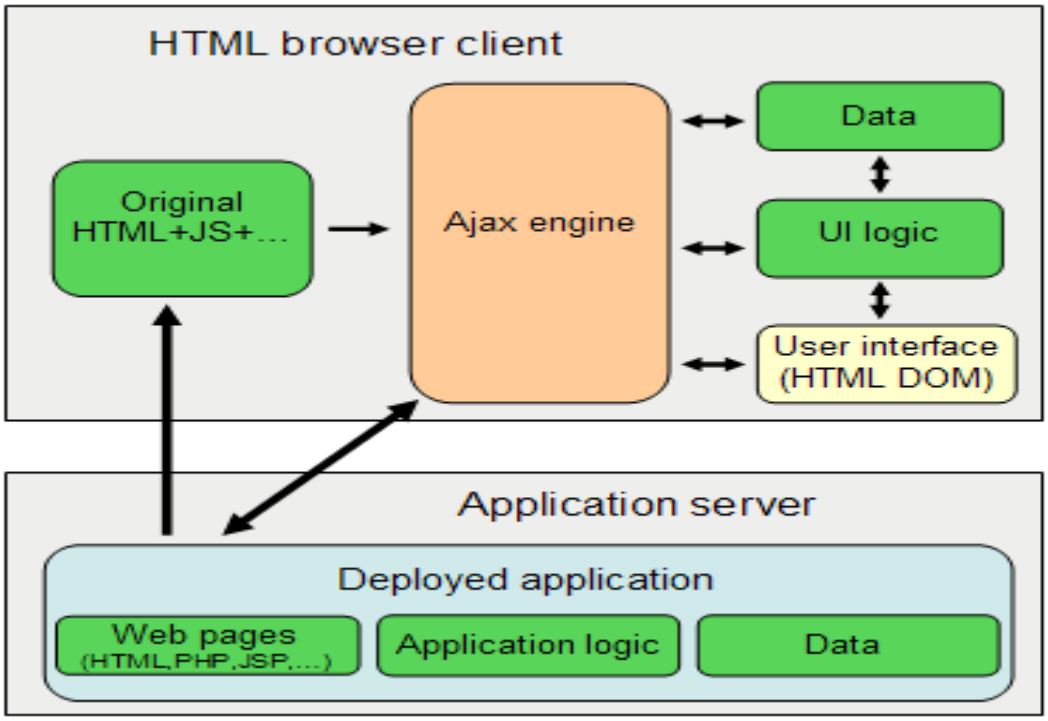

Рисунок 3– Взаємодії у Web за допомогою AJAX. [8]

При певних діях користувача (наприклад при активізації кнопки в складі користувацького інтерфейсу) браузер генерує запит і за допомогою JavaScript-об’єкта XMLHttpRequest відправляє його на сервер. При цьому метод доступу може бути GET або POST. Користувацький інтерфейс під час відправлення запиту і отримання відповіді не блокується і користувач може продовжувати виконувати певні дії, результатом яких можуть бути нові запити до сервера — Ajax підтримує декілька одночасних взаємодій сторінки з сервером. Користувацький інтерфейс виглядає і реагує на дії користувача як звичайна програма, що полегшує роботу з ним.

Сервер оброблює запит і відправляє браузеру відповідь у форматі XML, JSON або подібних. При цьому не відбувається генерації усієї сторінки (як у класичному механізмі доступу), тому час обробки запиту скорочується. Це дозволяє зменшити навантаження на сервер або збільшити кількість клієнтів, що можуть працювати одночасно.

Браузер, за допомогою JavaScript, обробляє отриману відповідь і модифікує сторінку без перезавантаження за допомогою DHTML.

Переваги цього механізму доступу — сторінка модифікується без повного перезавантаження, збільшується швидкість роботи з Web-програмою, зменшується трафік між сервером та клієнтом, метод роботи користувача з web-програмою є зручним.

Недоліки методу:

- важкість у розробці та налагодженні через використання мови сценаріїв JavaScript, що має специфічне застосування тому вона мало пристосована до розробки багатих web-програм.
- зміст сторінок, згенерованих за допомогою Ajax, не індексується пошуковими системами і сторінку не можна зберегти за допомогою браузера збережеться лише початкова сторінка та сценарії JavaScript.
- на сторінку, згенеровану за допомогою Ajax, не можна поставити

пряме посилання — при модифікації сторінка не змінює адреси.

Для подолання вказаних недоліків потрібно:

Обмежети використання мови сценаріїв JavaScript і використати технологію Java Апплетів. Повністю відмовитись від використання JavaScript не можна (це єдиний спосіб динамічної зміни сторінки, що, хоч і з недоліками, але функціонує в більшості сучасних браузерів) але якщо перенести більшу частину функціональних можливостей з сценарію JavaScript до Java Апплету і використовувати JavaScript лише для зв’язку HTML сторінки з Апплетом то складність розробки та налагодження такої системи буде на порядок нижча.

Створювати окремі статичні сторінки, що матимуть той самий вміст, що і динамічні сторінки, але їх зможуть прочитати та обробити пошукові системи а також переглянути ті користувачі, що використовують застарілі браузери або браузери із відключеними або заблокованими додатковими можливостями (JavaScript, Java, Flash і т.п.). Також ці статичні сторінки користувач може зберегти на свій комп’ютер для перегляду оффлайн або редагування за допомогою HTML-редакторів.

Створити спеціальний елемент користувацького інтерфейсу — «посилання на цю сторінку», що міститиме спеціально сформоване посилання, перейшовши за яким відкривається сторінка з таким самим вмістом, як і згенерована динамічно. Це потребує модифікації серверної частини (в більшості випадків ця модифікація є незначною), але подолання цього недоліку є дуже важливим для комфортної роботи з Web-сторіками, що побудовані динамічно.

Другий та третій пункти вже доволі широко використовується на сайтах, побудованих за допомогою концепції Web 2.0.

Спосіб, вказаний в першому пункті ще мало вивчений, тому майже не зустрічається на сайтах. Реалізація цього способу призведе до створення базового набору засобів, за допомогою якого розробники web-програм зможуть більш ефективно та з меншими витратами часу створювати web-програми, що матимуть багаті можливості та звичний для користувачів інтерфейс.

1.2.9 Альтернативна взаємодія у Web за допомогою Java Апплетів

Цей метод схожий на попередній, лише замість JavaScript-об’єкта XMLHttpRequest об’єкта використовується Java Апплет.

Ця заміна на перший погляд може здатися незначною — JavaScript (за допомогою об’єкту XMLHttpRequest) та Java Апплет мають схожі можливості для створення запитів, передачі їх на сервер та обробки отриманої відповіді. Але Java Апплет має набагато ширші можливості для обробки отриманої інформації.

Також перевагою Апплетів над сценаріями JavaScript є їх значно вища швидкодія та значно менші проблеми з розробкою та налагодженням — для Java існують досить потужні інтегровані середовища розробки та налагодження. Вони містять можливості, які відсутні в програмах для розробки сценарії JavaScript (які в більшості є простими текстовими редакторами з підсвіткою синтаксису та мінімальними функціональними можливостями) :

- вбудований налагоджувач;
- інструменти для рефакторингу (повного або часткового перетворення внутрішньої структури програми при збереженні її зовнішньої поведінки);
- проектування UML діаграм (графічний опис для об’єктного моделювання в сфері розробки програмного забезпечення);
- система керування версіями (програмне забезпечення для полегшення роботи з інформацією, що часто змінюється, основне застосування — слідкування за розробкою програм)
- колекція шаблонів коду та бібліотек, що дозволяють позбавитися рутинних операція при розробці Апплетів.

1.3 Аналіз існуючих віртуальних лабораторій функціонально логічного моделювання .

1.3.1 EDA Playground

EDA Playground це безкоштовний веб-додаток, що дозволяє користувачам редагувати, моделювати (переглядати waveforms), синтезувати і зберігати їх HDL код. Його мета полягає в тому, щоб прискорити вивчення проектування та розробки TestBench з легким спільним використанням коду і простим доступом до симуляторів і бібліотек. EDA Playground спеціально розроблений для невеликих дослідницьких зразків і прикладів (не призначений для використання повномасштабного FPGA або ASIC дизайну).

Модель використання проста - введіть свій код, виберіть свій улюблений симулятор або інструмент синтезу, і виберіть команду “Run” . Якщо моделювання відвантажило хвилі, все що вам потрібно зробити, це натиснути на прапорець і хвилі будуть відкриватися в новому вікні після запуску. Будь-який код або вихідні форми хвиль можуть бути збережені у вигляді статичного HTML-посилання, так що будь-хто може відкрити код, повторно запустити його, і отримати той же результат.

В даний час працює EDA Playground є проектом з відкритим вихідним кодом і дозволяє вільно моделювати та синтезувати EDA інструменти. Він підтримує кілька HDLs, таких як SystemVerilog, VHDL, MyHDL і Migen. Інженери можуть працювати з кількома бібліотеками і методологіями, такими як UVM, OVL, SVUnit і cocotb. UVM в даний час одна з найбільш популярних методик перевірки. Для синтезу, EDA Playground використовує фреймворк з відкритим кодом Yosys і VTR потоколи. Може використовуватися з наступними симуляторами :

- Icarus Verilog
- GPL Cver
- VeriWell

- Questa

Більшість користувачів використовують EDA Playground для швидкого прототипування або ж коли просто намагаються щось попробувати. Деякі з них використовувати його, щоб заберегти приклади коду, задаючи і відповідаючи на питання на інтернет-форумах. Деякі інженери , використовують EDA Playground під час технічних інтерв'ю для тестування HDL коду і перевірки навичок кандидатів. Один з користувачів , Ніл Джонсон, створив практичні підручники для бібліотеки SVUnit на EDA Playground.

EDA Playground працює з університетами, щоб створити практичну платформу веб-посібників і лабораторій, які будуть використовуватися в реальних університетських курсах цифрового дизайну.

Інтерфейс користувача виглядає наступним чином :

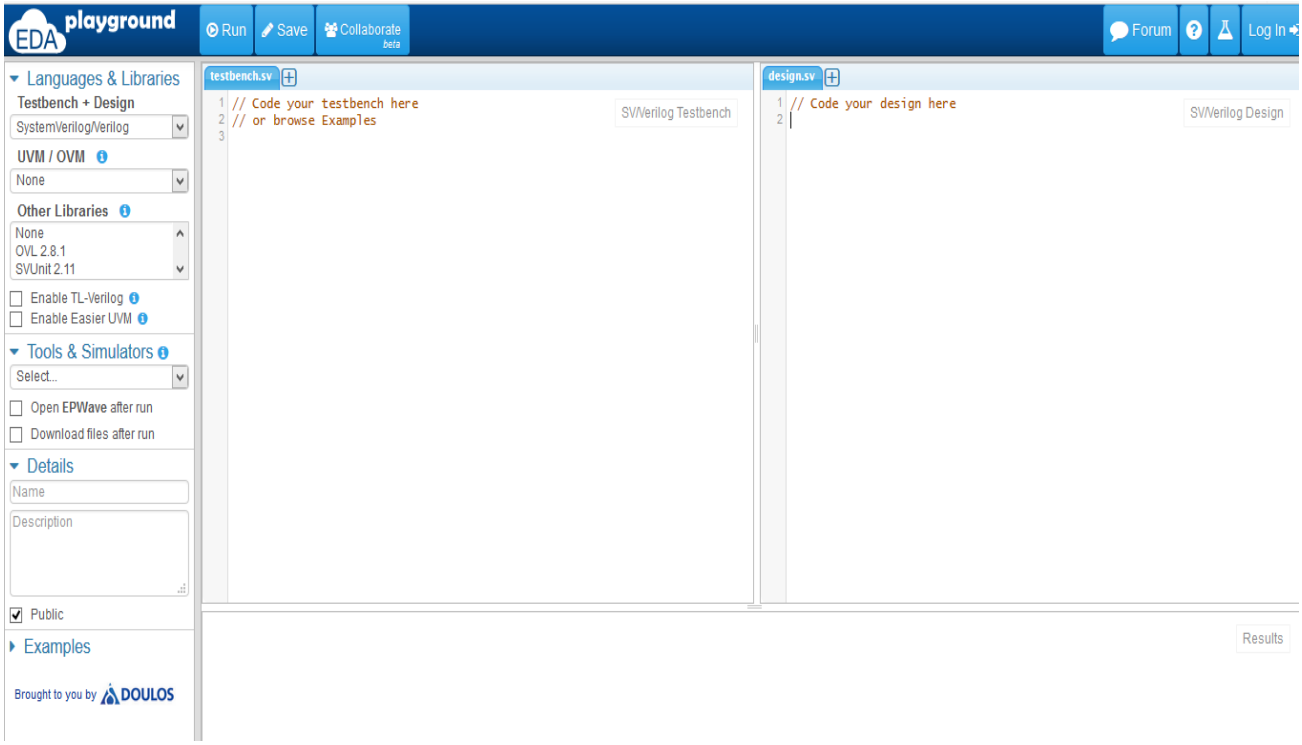

Рисунок 4 – Інтерфейс робочого простору в EDA Playground.

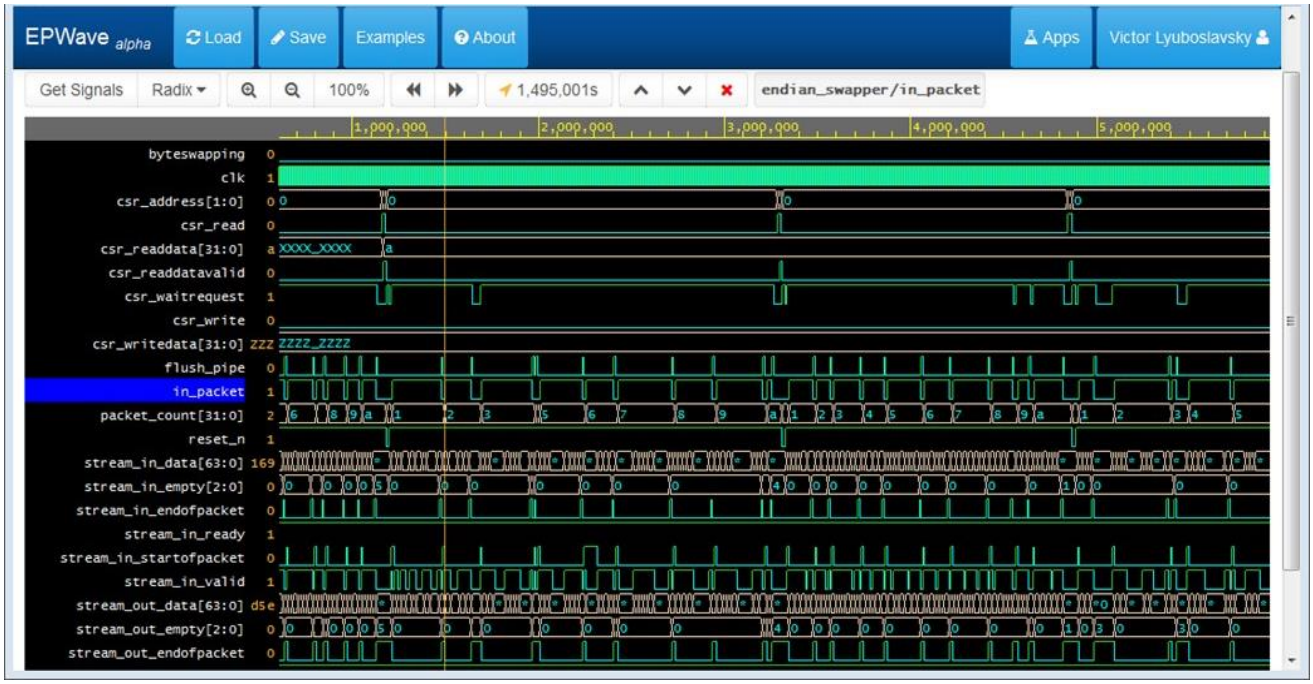

Рисунок 5 – Інтерфейс вихідних “waveforms” в EDA Playground.

1.3.2 iVerilog.com Verilog Online simulator.

Цей сайт дозволяє запускати Verilog моделювання з середовища веб-браузера. Якщо ви використовуєте Windows, Linux або Mac комп'ютер, смартфон або планшет - ви завжди можете бути в змозі запустити моделювання Verilog. Цей сайт використовує Icarus Verilog для імітації двигуна.

Мета цього сайту, зробити моделювання Verilog більш доступним і широко поширеним. За допомогою нього , ви можете запускати симуляції з IPAD, iPhone, IPod Touch, або будь-якого з Android-телефонів на доступних платформах. Інструмент на основі веб також дозволяє запускати симуляції без клопоту установки програмного забезпечення (яке іноді вимагає компіляції програмного забезпечення з вихідного коду). У тих випадках, коли використовується комп'ютер загального користування, наприклад, в бібліотеці

або комп'ютерної лабораторії школи чи університету, інструмент на веб-основі може бути єдиним варіантом для запуску Verilog.

Для роботи з даним сервісом введіть Verilog код в головному текстовому полі. При натисканні кнопки "Виконати код", ваш текст буде працювати і на виході буде відображатися в нижній частині сторінки. В даний час підтримується тільки вивід тексту.

Інтерфейс користувача виглядає наступним чином :

Рисунок 6 – Інтерфейс робочого простору в iVerilog.com.

Дана розробка виглядає не допрацьованою проте також справляється з поставленою задачею. Як бачимо в даній реалізації відсутня підсвітка синтаксису , що не сприяє полегшенню розробки .

1.3.3 Codingground

Цей сервіс надає можливість розробки на 95-ти різноманітних мовах , зокрема , дозволяє запускати Verilog моделювання з середовища веб-браузера. Він інтегрований в систему дистанційного навчання tutorialspoint.

Одна з найкращих реалізацій які були досліджені . Даний сервіс містить файловий менеджер для роботи з файлами , які знаходяться в “хмарі” , текстове поле з підсвіткою синтаксису та надає можливість компіляції та запуску файлів. Також надається доступ до роботи з терміналом. Імплементована інтеграція з такими хмарними сховищами даних як Dropbox , GitHub , Google Drive , OneDrive. Гнучкий інтерфейс і відсутність необхідності реєстрації характеризують цей сервіс лише з кращої сторони.

Проте у даного сервіса є великий мінус у вигляді проблем з безпекою. Заради зручності довелося нею пожертвувати. Лише прямий доступ до терміналу у злочинних руках уже може спричинити серйозні проблеми .

Інтерфейс користувача виглядає наступним чином :

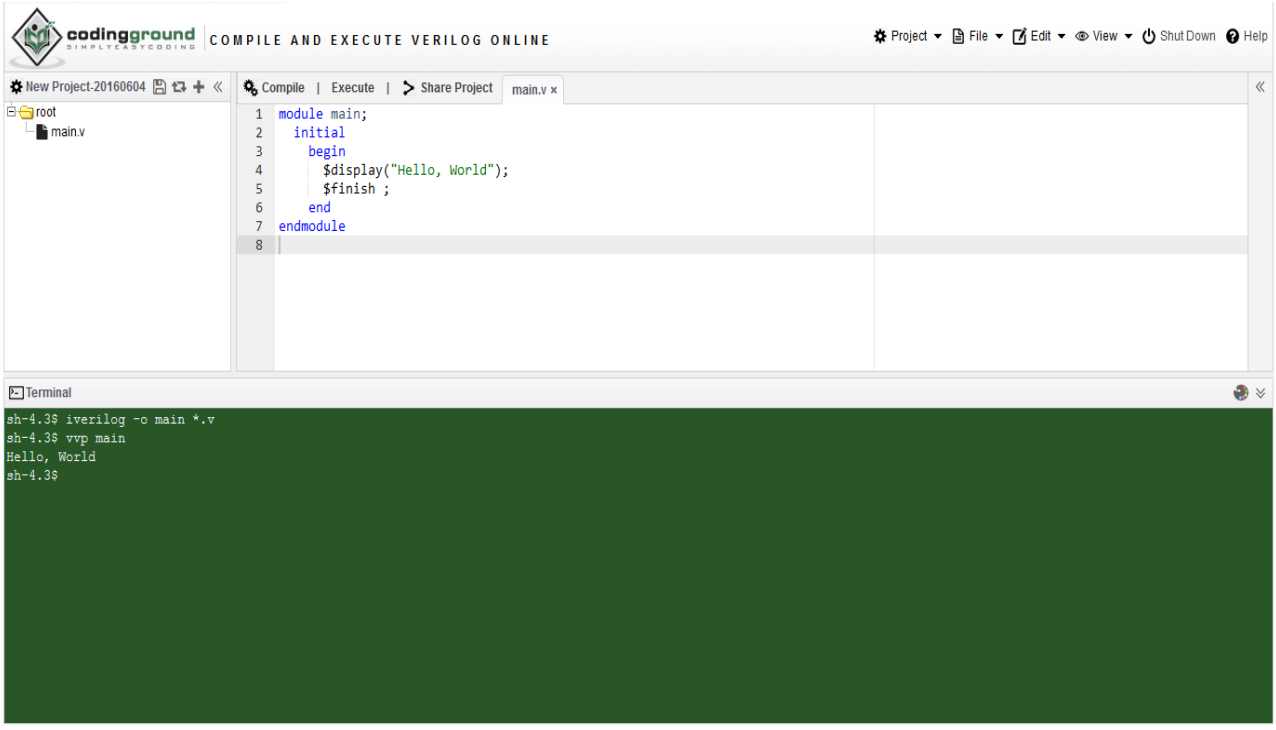

Рисунок 7 – Інтерфейс робочого простору в Codingground.

1.3.4 Порівняння розглянутих рішень.

Хоча і всі сервіси спроектовані і розроблені для практично однакових цілей їх функціонал та інтерфейс відрізняються .Оскільки відсутні дані про архітектуру даних сервісів можна провести їх порівняння на основі таких характеристик як зручність інтерфейсу , підтримка основного функціоналу, підтримка додаткового функціоналу , простота використання , можливість збереження проектів та окремих файлів , можливість інтеграції з іншими платформами ,безкоштовне користування і безпека.

Таблиця 1 – Порівняння сервісів .

Назва	EDA Playground	iVerilog.com	Codingground
Зручність інтерфейсу	+/-	-	+
Основний функціонал	+	+	+
Додатковий функціонал	+	-	+
Простота використання	-	+/-	+
Можливість збереженння проектів	+	-	+
Інтеграція з іншими платформами	+	-	+
Безкоштовне користування	+	+	+
Безпека	+	+	-

Згідно з таблиці 1 можна замітити що при оцінюванні за заданим критеріями найкраще виглядає сервіс Codingground , навіть не зважаючи на проблеми з безпекою. Тому при розробці власної реалізації буде зроблений акцент на подібній на подібний варіант , но також будуть враховані всі недоліки.

1.4 Висновки

В даному розділі були досліджені різні методи та засоби створення веб-додатків для САПР та веб-додатків в цілому . Також був проведений аналіз існуючих віртуальних лабораторій функціонально-логічного моделювання та зроблені висновки для подальшої розробки.

2. ОПИС ПРОГРАМНОГО ПРОДУКТУ. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ.

2.1 Постановка задачі .

Задачею даної роботи є реалізація власної лабораторії функціонально-логічного моделювання, гуртуючись на перевагах та недоліках уже готових рішень. Розробка повинна задовольняти наступні основні вимоги :

- надавати зручне середовище для моделювання;
- надавати доступ до файлів на сервері з можливістю їх редагування;
- надавати можливість зберігати , компілювати та запускати файли за допомогою Verilog компіляторів.

Проаналізувавши уже готові рішення та способи їх застосування , можна зробити висновок що актуальною буде розробка з невеликою кількістю функціоналу , зручним інтерфейсом та максимально простою складністю користування . Це спричинено бажанням забезпечити легкий та зручний доступ до симулятора для користувача , оскільки такі системи переважно використовуються для розробки або перевірки не великих систем або ж для навчання.

Отже , очікується отримати односторінковий веб-додаток з можливістю редагувати , та запускати на моделювання Verilog код.

2.2 Загальний опис та аналіз архітектури.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

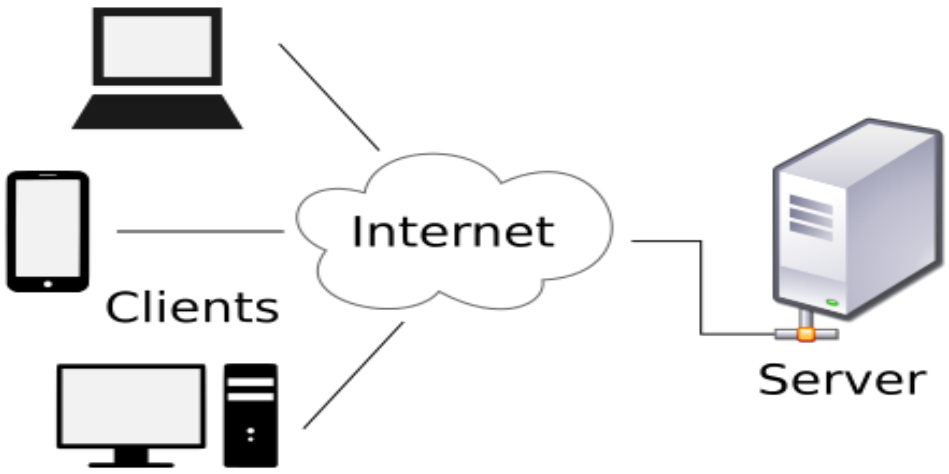

Рисунок 7 – Загальний архітектурний шаблон.[5]

В даній розробці архітектура виглядає таким чином :

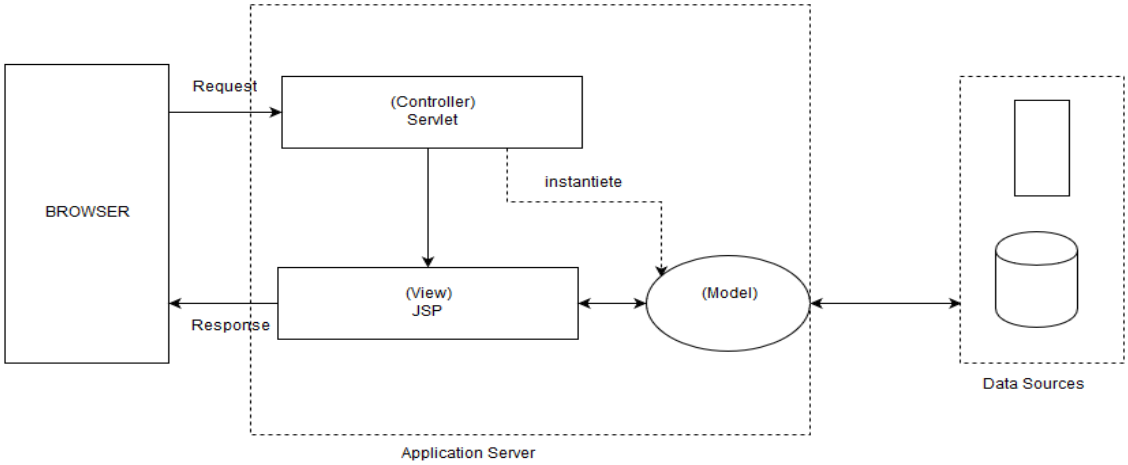

Рисунок 8 – Архітектура додатку.

Склад розробленого рішення.

Дана розробка складається з одного .jsp файла для генерації клієнтського HTML коду ,одного Javascript файла , 5 сервлетів у якості контролерів також підключено 3 додаткові сторонні модулі : dhtmlx , CodeMirror і iVerilog .

Алгоритм згідно якого відбувається клієнт-серверна взаємодія наступний:

- Спочатку текст програми для моделяції пишеться в текстовому полі у веб браузері , який задопомогою JavaScript бібліотеки CodeMirror підтримує підсвідку Verilog синтаксису.
- Потім при спробі запуску коду на виконання поточний текст програми прив’язується до текстового поля і у вигляді змінної передається для AJAX запиту на сервер.
- Виконується AJAX Post запит на сервер .
- Java Controller отримує код для запуску та викликає iVerilog компілятор.
- Після компіляції Java Controller запукає вихідний файл на виконання та отримує вихідні дані.
- Отримані вихідні дані відправляються у вигляді результату запиту назад на клієнт.

- Отримані дані обробляються та виводяться у відповідному модулі на клієнті.

2.3 Інсталяція та розгортання веб-додатку.

Вимоги до клієнтської машини мінімальні , потрібна лише можливість працювати у веб-браузері , пристрої вводу та виведення інформації.

Вимоги ж до сервера дещо більші . На сервері повинен бути установленим Icarus Verilog , для компіляції вихідного коду та JRE (Java Runtime Environment) для роботи з серверною частиною додатка , яка написана на Java.

Для розгортання віртуальної лабораторії функціонально-логічного проектування на сервері необхідно завантажити verilogOnline.war файл , який містить в собі упакований веб додаток.

Для початку , перед інсталяцією потрібно переконатися що встановлені та налаштовані всі необхідні для інсталяції залежності. Якщо ні ,то потрібно слідувати наступному алгоритму:

1. Встановіть середу виконання Java ™ (JRE) або Java Development Kit (JDK). Для установки можна вибрати один з наборів розробників IBM®.
2. Задайте змінну середовища з шляхом до JRE або JDK:
   - У разі установки JRE створіть змінну середовища JRE_HOME і вкажіть в ній шлях до установчого каталогу JRE, наприклад:
     Windows: C: \ Program Files \ Java70 \ jre
     linux: / usr / local / java70 / jre
   - У разі установки JDK створіть змінну середовища JAVA_HOME і вкажіть в ній шлях до установчого каталогу JDK, наприклад:
     Windows: C: \ Program Files \ Java70
     linux: / usr / local / java70 /

3. Завантажте і установіть Icarus Verilog .
4. Задайте змінну середовища з шляхом до Icarus Verilog.
5. Завантажте програму сервера Apache Tomcat.
6. Розпакуйте архів Apache Tomcat.
7. Перемістіть розпаковану папку в постійне розташування. Це розташування вказується як CATALINA_HOME, наприклад: C: \ apache-tomcat-версія.
8. Створіть змінну середовища CATALINA_HOME вкажіть в ній каталог CATALINA_HOME наприклад: C: \ apache-tomcat-версія
9. Додайте адміністратора сервера в файл CATALINA_HOME / conf / tomcat-users.xml і збережіть зміни, наприклад:
   <Role rolename = "manager-gui" />
   <User username = "admin" password = "admin" roles = "manager-gui" />
10. Перейдіть в каталог CATALINA_HOME / bin і запустіть сервер Apache Tomcat за допомогою файлу startup. Файл різниться в залежності від операційної системи.
    - Windows: startup.bat
    - linux: startup.sh

Apache Tomcat запущений після видачі в командному рядку повідомлення Сервер запущений через число мс.

Коли ж всі необхідні для інсталяції залежності встановлені та налаштовані і verilogOnline.war завантажений на сервер потрібно виконати наступну процедуру:

1. У браузері введіть наступний URL, щоб відкрити програму Tomcat Manager: http: // localhost: 8080 / manager / html
2. Введіть ім'я та пароль користувача.
3. У розділі Розгортання> Файл WAR для розгортання виберіть Вибрати файл.
4. Виберіть файл verilogOnline.war.

5. Виберіть Розгорнути. Файл verilogOnline.war розгортається, запускається і відображається в розділі Додатки. Крім того, файл verilogOnline.war копіюється в каталог CATALINA_HOME / webapps.
6. Переконайтеся, що віртуальна лабораторія доступна. Для цього додайте рядок довідка / index.jsp в кінець URL сервера додатків, Наприклад: http: // localhost: 8080 / verilogOnline / index.jsp.

2.4 Опис використнаних технологій та бібліотек.

2.4.1 CodeMirror

CodeMirror представляє собою універсальний текстовий редактор реалізований в JavaScript в браузері. Це спеціалізована бібліотека для редагування коду і поставляється з декількома режимами мови і аддонів, які реалізують більш розширені функціональні можливості редагування.

Хороше API і система CSS тематизації доступні для налаштування CodeMirror, щоб покращити додатки, розширюючи їх новими функціональними можливостями.

CodeMirror є проектом з відкритим вихідним кодом під ліцензією MIT. Це редактор, який використовується в розробницьких інструментах для Firefox і Chrome , Light Table, Adobe Кронштейни , Bitbucket і багатьох інших проектів.

Розробка і супровід відбувається на GitHub (альтернативний репозиторій ГИТ). Обговорення навколо проекту робиться на форумі. Існує також CodeMirror-анонс список , який використовується тільки для великих оголошень (наприклад, нові версії).

Підтримка браузерів

Настільні версії наступних браузерів, в стандартному режимі (HTML5 <! DOCTYPE HTML> рекомендується) підтримуються:

- Firefox версії 4 і вище
- Chrome будь-якої версії
- Safari версії 5.2 і вище
- версія Internet Explorer 8 і вище
- Opera версії 9 і вище
- Підтримка сучасних мобільних браузерів є експериментальним. Останні версії браузера Chrome і ОС IOS на Android повинні працювати добре.

CodeMirror є компонентом редактора коду, який може бути вбудований в веб-сторінки. CodeMirror працює з режимами конкретної мови. Режими програми, за допомогою JavaScript забезпечує підсвітку тексту, написаного на даній мові. Дистрибутив поставляється з цілим рядом режимів.

В даній роботі , ця бібліотека була використана для підсвітки Verilog синтаксису.

2.4.2 Icarus Verilog

Icarus Verilog є інструментом моделювання і синтез Verilog. Він діє як компілятор, застосовується для компіляції вихідного коду, написаного на Verilog (IEEE-1364) в якийсь кінцевий формат. Для пакетного моделювання, компілятор може генерувати проміжну форму, яка називається VVP збірки. Ця проміжна форма виконується за допомогою команди `` VVP ". Для синтезу, компілятор генерує нетлісти в потрібному форматі.[2]

Власне компілятор призначений для розбору і опису дизайну, написаного в стандарті IEEE IEEE Std 1364-2005.

Icarus Verilog находиться в стадії розробки, а так як стандартна мова не стоїть на місці або, він, ймовірно, буде також завжди розвиватися. Основною

метою є використання в Linux, хоча він добре працює на багатьох подібних операційних систем. Різні люди внесли свій вклад для проекту та стабільних релізів. Ці релізи портовано добровольцями. Icarus Verilog був перенесений на інші операційної системи, як інструмент командного рядка, і є інсталятори для користувачів без компіляторів. Ви можете зібрати його повністю з вільними інструментами, хоча є скомпільовані двійкові файли стабільних релізів.

В даній роботі цей інструмент застосовується для компіляції та отримання результату виконання Verilog коду.

2.4.3 DHTML

Динамічний HTML (Dynamic HTML, DHTML) не є якимось особливим мовою розмітки сторінок. Це всього лише термін, застосовуваний для позначень HTML-сторінок з динамічно змінним вмістом.

Реалізація DHTML стоїть на трьох "китах": безпосередньо HTML, каскадні таблиці стилів і мовою сценаріїв. Ці три компоненти DHTML пов'язані між собою об'єктною моделлю документа (DOM, Document Object Model), яка є по суті інтерфейсом прикладного програмування (API). DOM пов'язує воєдино три перерахованих компонента, надаючи простому документу HTML нову якість - можливість динамічного зміни свого вмісту без перевантаження сторінки.

Об'єктна модель документа робить все елементи сторінки програмованими об'єктами. З її допомогою через мови сценаріїв можна отримати доступ і управляти всім, що є в документі. Кожен елемент HTML доступний як індивідуальний об'єкт, а це означає, що можна змінювати значення будь-якого параметра будь-якого тега HTML-сторінки, і, як наслідок, документ дійсно стає динамічним. Будь-яка дія користувача (клацання кнопкою миші, переміщення миші у вікні браузера або натискання клавіші клавіатури)

об'єктною моделлю документа трактується як подія, яка може бути перехоплено і оброблено процедурою сценарію.

В даній роботі був використаний для створення динамічних слоїв , управління ними та їх вмістом.

2.4.4 JSP

JSP (JavaServer Pages) - технологія, що дозволяє веб-розробникам створювати вміст, який має як статичні, так і динамічні компоненти. Сторінка JSP містить текст двох типів: статичні вихідні дані, які можуть бути оформлені в одному з текстових форматів HTML, SVG, WML, або XML, і JSP- елементи, які конструюють динамічний вміст. Крім цього можуть використовуватися бібліотеки JSP-тегів, а також EL (Expression Language), для впровадження Java-коду в статичне вміст JSP-сторінок.

Код JSP-сторінки транслюється в Java-код сервлету за допомогою компілятора JSP-сторінок Jasper, і потім компілюється в байт-код віртуальної машини java (JVM). Контейнери сервлетів, здатні виконувати JSP-сторінки, написані на платформо незалежній мові Java. JSP-сторінки завантажуються на сервері і управляються зі структури спеціального Java server packet, який називається Java EE Web Application. Зазвичай сторінки упаковані в файлові архіви .war і .ear.

JSP є платформонезалежна , яку переносять і легко розширюється технологією для розробки веб-додатків.

В даній роботі була використана для генерації HTML коду для клієнта.

2.4.5 XML

В 1998 році міжнародною організацією W3C мова XML. XML (eXtensible Markup Language) - це розширювана мова розмітки, призначена для опису в текстовій формі структурованих даних. Цей текстовий (text-based) формат,

багато в чому схожий з HTML, розроблений спеціально для зберігання і передачі даних.

XML дозволяє описувати і передавати такі структуровані дані, як:

- окремі документи
- метадані, що описують вміст якого-небудь вузла Internet
- об'єкти, що містять дані і методи роботи з ними (наприклад, елементи управління ActiveX або об'єкти Java)
- окремі записи (наприклад, результати виконання запитів до баз даних)
- всілякі Web-посилання на інформаційні та людські ресурси Internet (адреси електронної пошти, гіпертекстові посилання й ін.)

Дані, описані на мові XML, називаються XML-документами. Мова XML легко читаємо і досить простий для розуміння. Якщо Ви були знайомі з HTML, то навчитися складати XML-документи не складе для Вас ніяких труднощів.

Оригінальний текст XML-документа складається з набору XML-елементів, кожен з яких містить початковий і кінцевий теги. Кожна пара тегів представляє частину даних. Тобто, як і HTML, мова XML для опису даних використовує теги. Але, на відміну від HTML, XML дозволяє використовувати необмежений набір пар тегів, кожна з яких представляє не те, як ув'язнені в неї дані повинні виглядати, а то, що вони означають.

Будь-який елемент XML-документа може мати атрибути, що уточнюють його характеристики. Атрибут - це пара ім'я = "значення", яка задається при визначенні елемента в початковому тегу.

Принцип розширюваності мови XML полягає в можливості використання необмеженої кількості пар тегів, які визначаються творцем XML-документа.

Принцип незалежності визначення внутрішньої структури документа від способів подання цієї інформації полягає в відділенні даних від процесу їх

обробки і відображення. Таким чином, отримані дані можна використовувати відповідно до потреб клієнта, тобто вибирати потрібне оформлення, застосовувати необхідні методи обробки.

Управляти відображенням елементів у вікні програми-клієнта (наприклад, у вікні браузера) можна за допомогою спеціальних інструкцій - стильових таблиць XSL (eXstensible Stylesheet Language). Ці таблиці XSL дозволяють визначати оформлення елемента в залежності від його місця розташування всередині документа, тобто до двох елементів з однаковою назвою можуть застосовуватися різні правила форматування. Крім того, мовою, які лежать в основі XSL, є XML, а це означає, що таблиці XSL більш універсальні, а для контролю коректності складання таких стильових таблиць можна використовувати DTD-описи або схеми даних, розглянуті нижче.

Формат XML, в порівнянні з HTML, має невеликий набір простих правил розбору, який дозволяє розбирати XML-документи, не вдаючись до будь-яких зовнішніх описів використовуваних XML-елементів. У загальному випадку XML-документи повинні відповідати таким вимогам:

- Кожен відкриває тег, що визначає деяку частину даних в документі, обов'язково повинен супроводжуватися закриває, тобто, на відміну від HTML, не можна опускати закривають теги.
- Вкладеність тегів в XML строго контролюється, тому необхідно стежити за порядком проходження відкривають і закривають тегів.
- У XML враховується регістр символів.
- Вся інформація, що розташовується між початковим і кінцевим тегами, розглядається в XML як дані, і тому враховуються всі символи форматування (тобто прогалини, переклади рядків, табуляції не ігнорує, як в HTML).

У XML існує набір зарезервованих символів, які повинні бути задані в XML-документі тільки спеціальним чином.

Багато фахівців розглядають XML як нову технологію інтеграції програмних компонент. Основними перевагами використання XML є:

- Інтеграція даних з різних джерел. XML можна використовувати для об'єднання різнорідних структурованих даних на середньому рівні трирівневих Web-систем, баз даних.
- Локальна обробка даних. Отримані дані в форматі XML можна розбирати, обробляти і відображати безпосередньо на клієнті без додаткових звернень до сервера.
- Перегляд і маніпулювання даними в різних розрізах. Отримані дані можуть оброблятися і проглядатися клієнтом різними способами в залежності від потреб кінцевого користувача.
- Можливість часткового оновлення даних. За допомогою XML можна оновлювати тільки ту частину структурованих даних, яка була змінена, а не всю структуру цілком.

Всі ці переваги роблять XML незамінним інструментом для розробки гнучких засобів пошуку інформації в базах даних, потужних трирівневих Web-додатків, а також додатків, що підтримують транзакції. Іншими словами, за допомогою XML можна формувати запити до баз даних різних структур, що дозволяє здійснювати пошук інформації в численних несумісних один з одним базах даних. Використання XML на середньому рівні трирівневих Web-додатків дозволяє здійснювати ефективний обмін даними між клієнтами і серверами систем електронної комерції.

Крім того, мова XML може використовуватися як засіб для опису граматики інших мов і контролю правильності складання документів.є

В даній роботі використовується для генерування файлового дерева та організації роботи веб-додатка.

2.4.6 Ajax

Ajax — група методів Web-розробки, що використовуються для створення Web-програм з багатими можливостями та мережевою взаємодією, що базується на «фоновому» обміні даними браузера з Web-сервером. В результаті сторінка не перезавантажується повністю і Web-програма стає швидкою та зручною.

Ajax це не самостійна технологія, а скоріше концепція використання декількох суміжних технологій. Ajax базується на двох основних принципах: використання технології взаємодії із сервером за допомогою JavaScript об’єкта XMLHttpRequest без перезавантаження усієї сторінки використання DHTML для динамічної зміни вмісту сторінки та реагування на дії користувача

Для передачі даних від сервера до клієнта використовуються формати XML або JSON. Класична модель web-програм пов’язана не лише з використанням базових web-технологій, а і з специфічним способом роботи з web-програмою, при якому web-браузер є лише низькорівневим терміналом. Він не має інформації про те, який етап роботи виконується користувачем. Він лише отримує готову сторінку в форматі HTML і відображає її користувачу.

У web-програмах, побудованих за допомогою технології Ajax, частина функціональних можливостей переноситься з сервера на клієнт. На деякі дії користувача така web-програма може реагувати самостійно. Якщо наявних можливостей не вистачає для виконання ініційованих користувачем дій то відбувається взаємодія із сервером, при цьому користувач може виконувати інші дії. Оскільки HTML документ присутній на стороні клієнта протягом всього часу роботи з web-програмою, то він здатний зберігати всю інформацію про її стан.

Технологія динамічного завантаження вмісту існувала і раніше — за допомогою атрибуту src можна було завантажити зовнішній сценарій JavaScript, який змінить поточну сторінку. Але цей метод не є дуже вдалим

через обмеження атрибуту src та додатковому навантаженні на сервер, бо він має виконати додаткові дії для генерації спеціального сценарію JavaScript, що містить інструкцію, як модифікувати поточну сторінку в нову.

Засоби, що використовуються в рамках технології Ajax не єдиний спосіб забезпечити асинхронний обмін даними з сервером. Наприклад Macromedia Flash (починаючи з 4ї версії) може завантажувати дані в форматі XML або CSV з серверу без перезавантаження сторінки. Але цю технологію не можна використовувати для створення багатих web-програм бо вона в основному використовується для роботи з мультимедійними даними і малопридатна для динамічної зміни вмісту сторінки.[5]

Пізніше Microsoft створила об’єкт XMLHttpRequest в Internet Explorer 5, що і став основою Ajax.

В даній роботі використовується для «фонового» обміну даними браузера з Web-сервером щоб отримувати результат компіляції і виконання Verilog коду без перезавантаження сторінки.

2.4.7 Web -сервер Apache

Найпоширеніший Web-сервер в світі - це Apache. За даними компанії Netcraft, загальна кількість Web-вузлів, що працюють під його управлінням, з 1996 року і ось уже навіть через 10 років, являється найбільш популярним веб-сервером у світі.. Для порівняння: на частку серверів Microsoft доводиться 17.22%, Netscape - 7%. Будучи безкоштовної відкритої програмою, призначеної для безкоштовних же Unix-систем (FreeBSD, Linux і ін.), Apache по функціональних можливостях і надійності не поступається комерційним серверам, а широкі можливості конфігурації дозволяють налаштувати його для роботи практично з будь-якої конкретної системою.

Apache Tomcat — контейнер сервлетів, розроблений Apache Software Foundation. Повністю написаний мовою програмування Java та реалізує специфікацію сервлетів і Java Server Pages від Sun Microsystems, що є стандартами для розробки веб-застосунків на Java.

Робочий прототип віртуальної лабораторії функціонально-логічного моделювання був розміщений на веб-сервер Apache.

2.5 Опис та результат роботи додатку.

Запустивши додаток в браузері , ми отримуємо сторінку як на рисунку 9

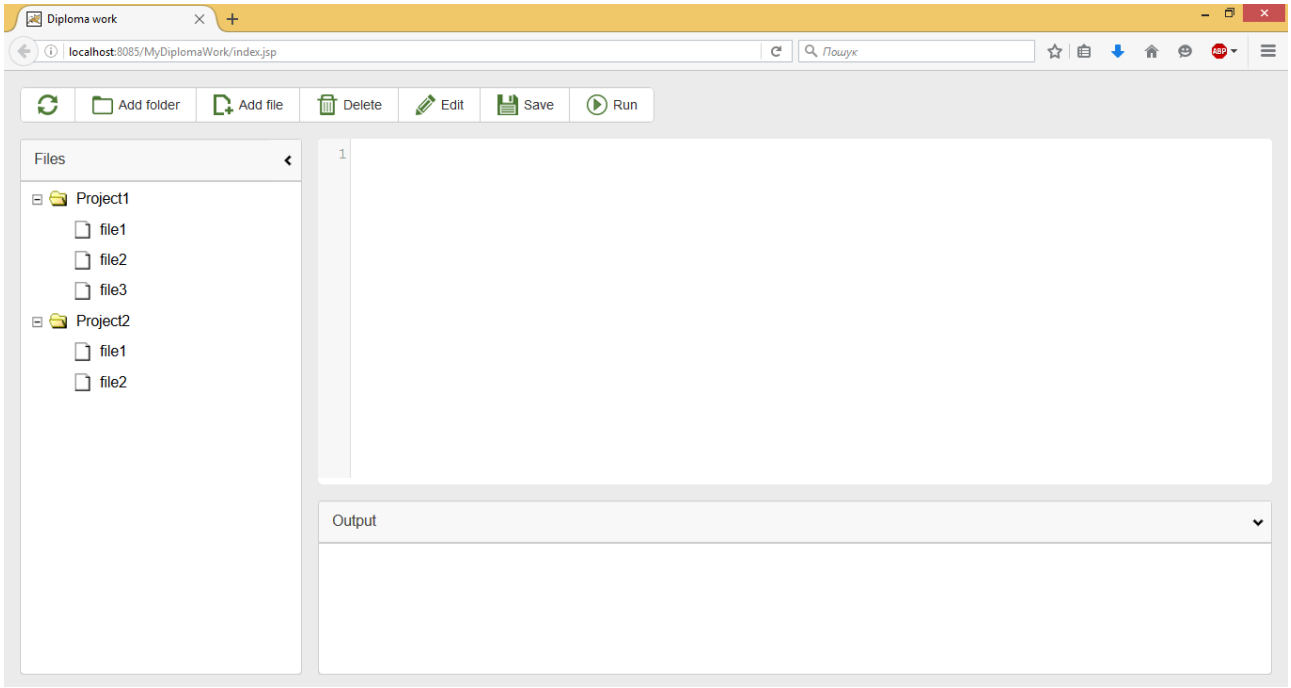

Рисунок 9 – Початкова сторінка додатку.

Як ми можемо побачити на рисунку 9 , даний додаток складається з трьох блоків : ліворуч знаходиться файловий менеджер для навігації по файлах, по центрі знаходиться текстовий редактор з підсвіткою Verilog синтаксису і нижній частині вікна знаходиться блок виводу інформації. Також в верхній частині вікна є навігаційна панель для роботи з додатком.

Далі попробуємо написати код простої програми для моделювання. Наприклад стандартного “Hello World” на Verilog (рисунок 10).

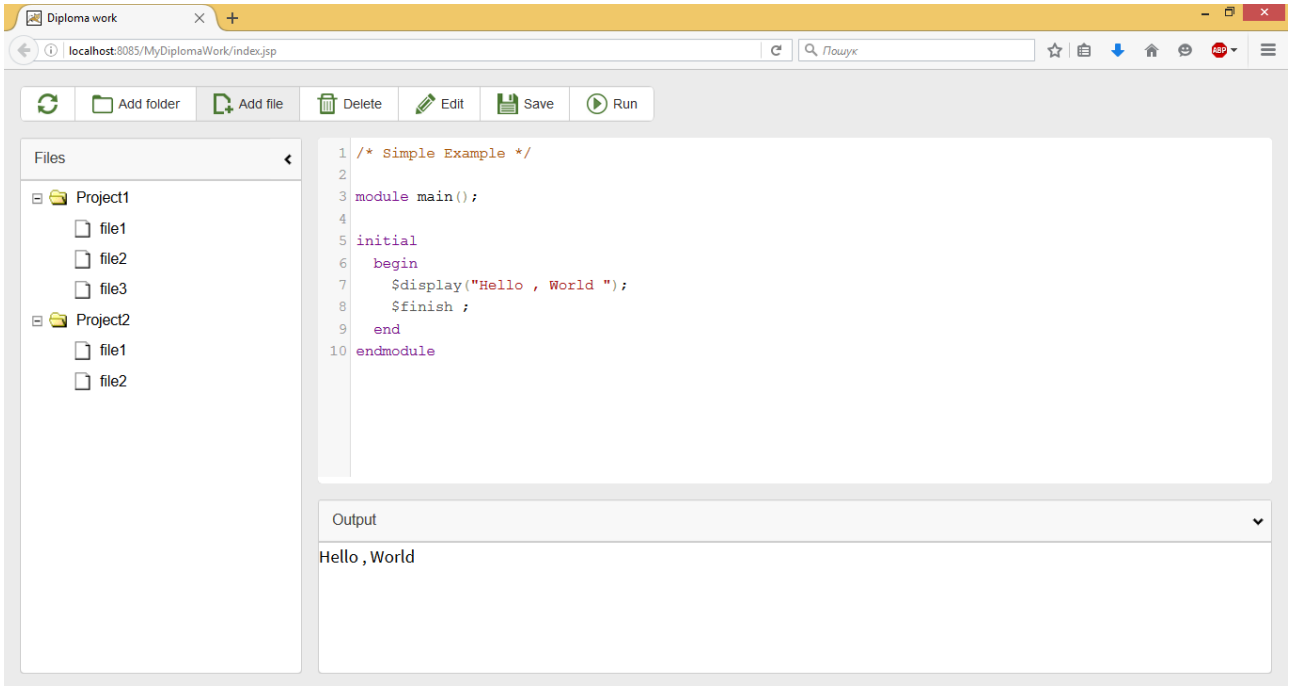

Рисунок 10 – Приклад роботи.

Для зручної розробки деякі блоки можна приховати і звільнити більше місця для текстового редактора , так як на рисунку 11.

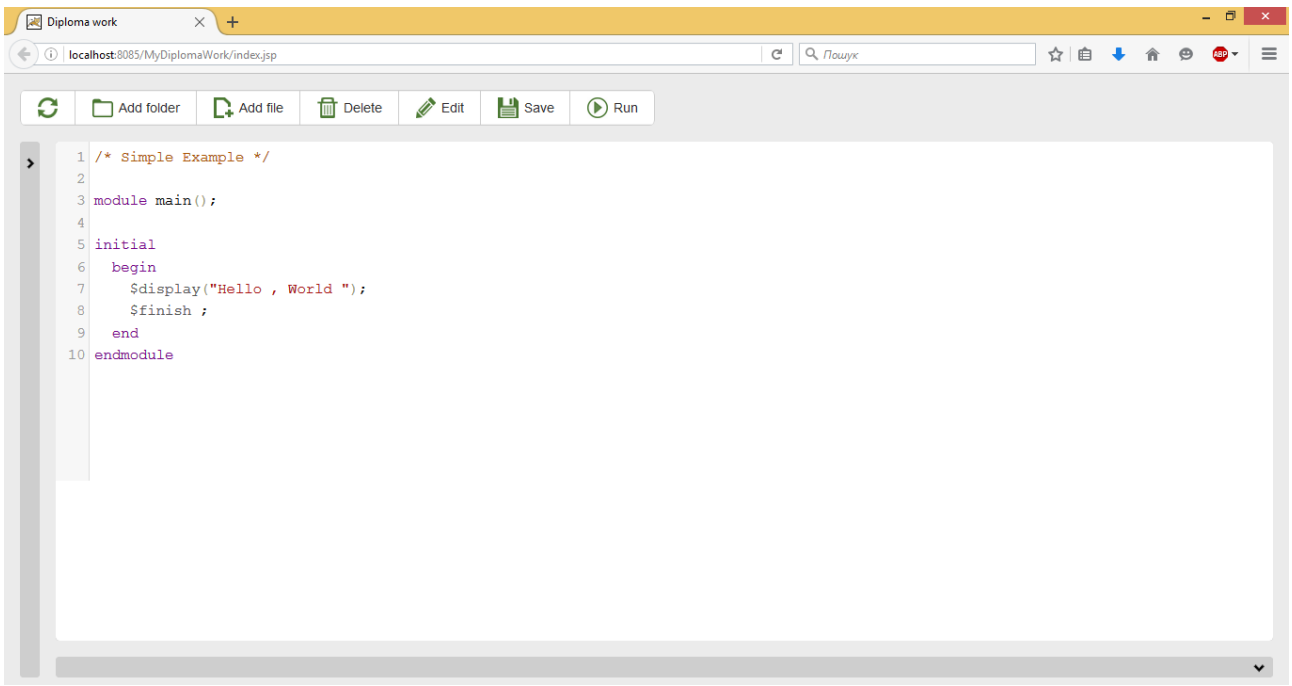

Рисунок 11 – Додаток з розгорнутим текстовим редактором.

При збереженні файла ми отримаємо відповідь про те що файл збережений , як на рисунку 12.

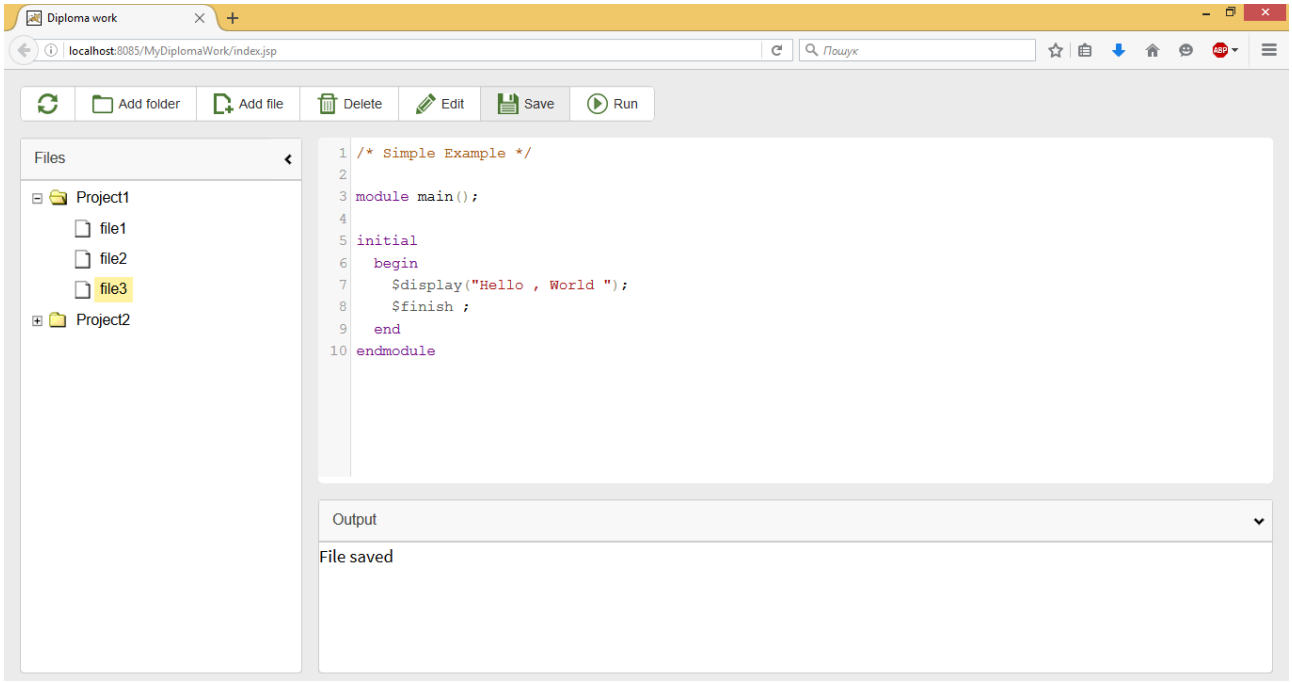

Рисунок 12 – Збереження файла.

Дана реалізація виглядає простою в використанні і не перевантажена додатковими модулями , які заважають розробці. Також можливість доступу до сервера та open source компілятора дає можливість добавляти власні додаткові модулі та розширювати функціонал.

2.6 Розширення функціоналу.

Отримана архітектура являється досить гнучкою і надає можливість розширення функціоналу в майбутньому. Наприклад динамічний модуль “Output” може приймати не лише текстову інформацію , а й може забезпечувати вивід “waveforms” намальованих на Javascript по даних з сервера. Також можлива додаткова загрузка vpi модулів для виклику функцій, написаних на мові C, з коду Verilog під час симуляції проекту.

Для збереження та поширення власних проектів можна добавити взаємодію з хмарними сховищами такими як Dropbox , GitHub , Google Drive , OneDrive та ін. Також можлива інтеграція бази готових простих прикладів для навчання . При необхідності можна добавити інші симулятори на сервер , та можливість вибору між ними клієнту. У випадку розростання сервісу , можна добавити форум для обміну інформації між користувачами .

В майбутньому можна добавити плагін для доступу до апаратних засобів користувача.

2.7 Висновки.

В даному розділі був проведений опис програмного продукту та засобів його реалізації. Реалізація задовольняє вимоги до основного функціоналі та надає гнучке і легке в використанні середовище для моделювання.

3. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для віртуальної лабораторії функціонально-логічного моделювання. Інтерфейс користувача був розроблений за допомогою мови розмітки HTML та мови програмування Javascript у середовищі розробки Netbeans. Серверна реалізація була розроблена на мові програмування Java.

Програмний продукт призначено для функціонально-логічного моделювання у власному веб-браузері.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

3.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

3.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – вибір оптимальної СКБД;

F_3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Java;

б) мова програмування Python;

Функція F_2:

а) MSQL Server;

б) Oracle.

Функція F_3:

а) інтерфейс користувача, створений за технологією HTML ,CSS, JS;

б) інтерфейс користувача, створений за технологією Java applet.

3.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

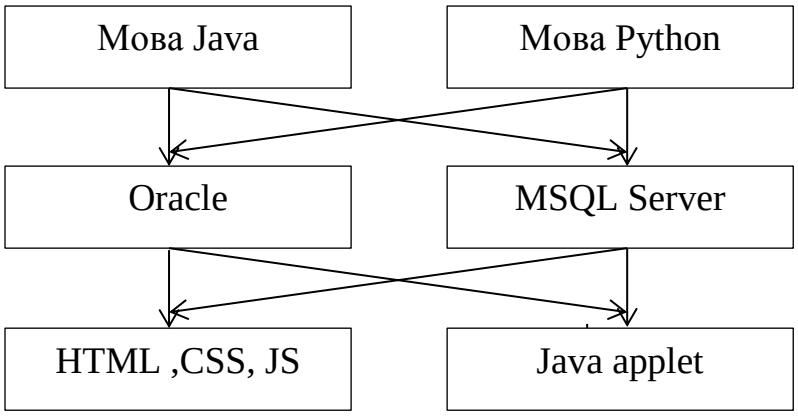

Рисунок 13 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 2 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Кросплатформений	Низька швидкодія
	Б	Займає менше часу при написанні коду	Не кросплатформений
F2	А	Подовжений термін користувацької підтримки	Більш висока вартість корпоративної ліцензії. Необхідність додаткової інсталяції

Продовження таблиці 2.

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F2	Б	Більш дешева вартість корпоративної ліцензії	Необхідність додаткової інсталяції, низький рівень користувацької підтримки
F3	*А*	Простота створення.	Необхідність додаткової інсталяції.
	Б	Простота створення , можливість розробки клієнтської і серверної на тій-же мові програмування.	Низька швидкодія

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки розрахунки проводяться з великими об’ємами вхідних даних, то час виконання програмного коду є дуже необхідним, тому варіант б) має бути відкинутий.

Функція *F2:*

Вибір СКБД не відіграє велику роль у даному програмному продукту, тому вважаємо варіанти а) та б) гідними розгляду.

Функція *F3:*

Оскільки, програмний продукт реалізується на мові Java, використовуємо варіант А як єдиний можливий.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3а
2. F1а – F2б – F3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

3.2 Обґрунтування системи параметрів ПП

3.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – швидкодія мови програмування;

– *X2* – об’єм пам’яті для збереження даних;

– *X3* – час обробки даних;

– *X4* – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

3.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 3 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Час обробки запитів користувача	X3	мс	1000	420	60
Потенційний об’єм програмного коду	X4	кількість строк коду	2000	1500	1000

За даними таблиці 3 будуються графічні характеристики параметрів – рис. 14 – рис. 17.

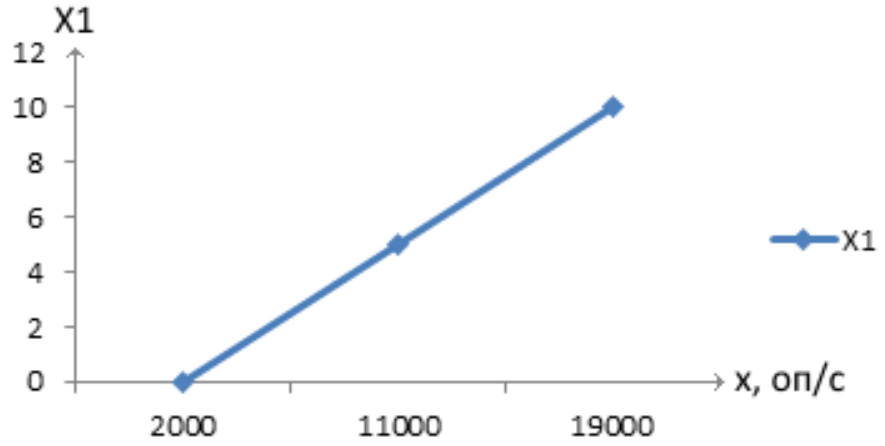

Рисунок 14 – X1, швидкодія мови програмування

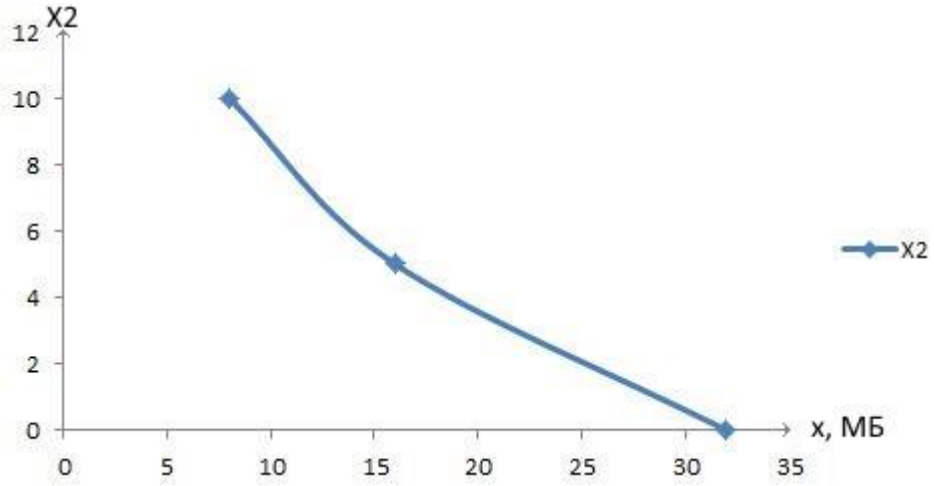

Рисунок 15 – X2, об’єм пам’яті для збереження даних

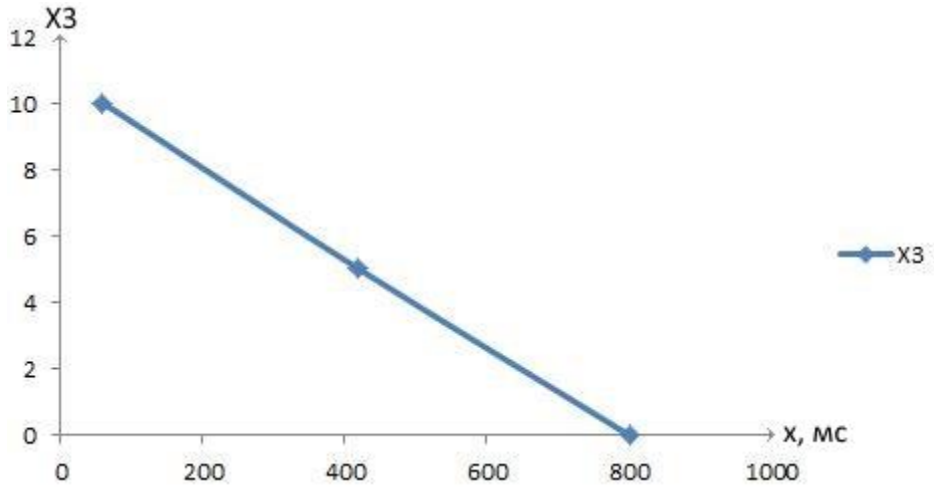

Рисунок 16 – X3, час виконання запитів користувача

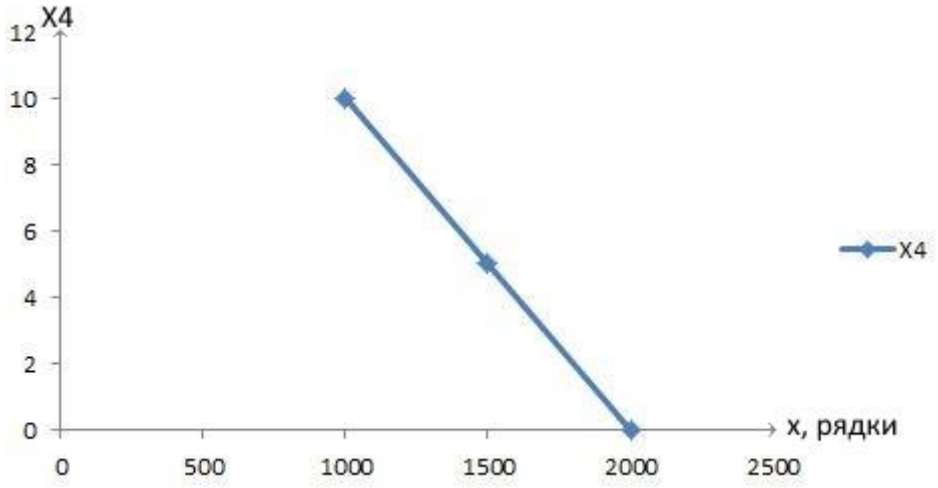

Рисунок 17 – X4, потенційний об’єм програмного коду

3.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки запитів користувача	Мс	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,75	217,56
	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105, \quad (1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}25. \quad (2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \quad (3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420{,}75. \quad (4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 420{,}75}{7^2(5^3-5)} = 1{,}03 > W_k = 0{,}67 \quad (5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 5.

Таблиця 5 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \qquad (6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}. \qquad (7)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j. \quad (8)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 6 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

3.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 1000 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 7):

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j}, \quad (9)$$

де n – кількість параметрів;$K_{\text{в}i}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

Таблиця 7 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3,6	0,215	0,774
F2(X2)	А	16	3,4	0,283	0,962
F3(X3,X4)	А	800	2,4	0,348	0,835
	Б	80	1	0,154	0,154

За даними з таблиці 7 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \quad (10)$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 0,774 + 0,962 + 0,835 = 2,57

K_{K2} = 0,774 + 0,962 + 0,154 = 1,89

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

3.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (11)$$

де T_P – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт;$K_{СК}$ – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

T_1 = 90·1.7·0.8 = 122.4 людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

T_2 = 27 · 0.9 · 0.8 = 19.44 людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

T_I = (122.4 + 19.44 + 4.8 + 19.44) · 8 = 1328,64 людино-годин;

T_{II} = (122.4 + 19.44 + 6.91 + 19.44) · 8 = 1345.52 людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 10 000 грн., один фінансовий аналітик з окладом 10 500 грн. Визначимо зарплату за годину за формулою:

$$СЧ = \frac{М}{T_m \cdot t} \text{ грн.}, \qquad (12)$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$СЧ = \frac{10\,000 + 10\,000 + 10\,500}{3 \cdot 21 \cdot 8} = 60{,}51 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$СЗП = С_ч \cdot Т_i \cdot КД, \qquad (13)$$

де $С_ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 60,51· 1328.64 · 1.2 = 96484,57 грн.

II. $С_{ЗП}$ = 60,51· 1345.52 · 1.2 = 97700,89 грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

I. $С_{ВІД} = С_{ЗП}$ · 0,22 = 96484,57 *0,22 = 21226,6 грн.

II. $С_{ВІД} = С_{ЗП}$ · 0,22 = 97700,89 *0,22 = 21494,19 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 10 000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 10\,000 \cdot 0{,}2 = 24000$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 24000 \cdot (1 + 0.2) = 28800$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД}$= $С_{ЗП}$ · 0,22 = 28800 * 0,22 = 6336 грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10 000 грн.

$С_А = К_{ТМ}· К_А·Ц_{ПР}$ = 1.15 · 0.25 · 10 000 = 2875 грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ}·Ц_{ПР} · К_Р$ = 1.15 · 10 000 · 0.05 = 575 грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ}$ =($Д_К – Д_В – Д_С – Д_Р$) · $t_З· К_В$ = (365 – 104 – 8 – 16) · 8 · 0.9 = 1706.4 годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ}· N_С· К_З· Ц_{ЕН}$ =1706,4 · 0,156 · 0,9733 · 2,0218 = 404,18 грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР}$·0.67 = 10 000· 0,67 =6700 грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 28800 + 6336 + 2875 + 575 + 404,18 + 6700 = 45690,18$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М-Г} = С_{ЕКС} / Т_{ЕФ} = 45690,18 / 1706,4 = 26,77$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} \cdot Т \qquad (14)$$

I. $С_М = 26,77 * 1328,64 = 35567,69$ грн.;

II. $С_М = 26,77 * 1345.52 = 36019,83$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67 \qquad (15)$$

I. $С_Н = 96484,57 * 0,67 = 64644,66$ грн.;

II. $С_Н = 97700,89 * 0,67 = 65459,59$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н \qquad (16)$$

I. $С_{ПП} = 96484,57 + 21226,6 + 35567,69 + 64644,66 = 217\,923,52$ грн.;

II. $С_{ПП} = 97700,89 + 21494,19 + 36019,83 + 65459,59 = 220\,674,5$ грн.;

3.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \quad (17)$$

$К_{ТЕР1} = 2,57 / 217\,923,52 = 1,17 \cdot 10^{-5}$;

$К_{ТЕР2} = 1,89 / 220\,674,5 = 0,85 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 1,17 \cdot 10^{-5}$.

3.6 Висновки до розділу 3.

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У

нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 1{,}17 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Java;

– Oracle DB;

– інтерфейс користувача, створений за допомогою HTML ,CSS, JS.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

ВИСНОВКИ

В даній роботі було розглянуто основні принципи та засоби створення веб-додатків в цілому та веб-додатків для САПР зокрема. Були досліджені існуючі лабораторії функціонально-логічного моделювання , та проведено їх порівняння.

Була розроблена власна лабораторія функціонально-логічного моделювання у вигляді клієнт-серверного веб-додатка . Для клієнтської частини були обрані DHTML – для динамічної зміни вмісту сторінок , Javascript для реалізації різних функцій та його бібліотеки для формування текстового редактора. Для серверної частини були обрані JAVA , як основна серверна мова , та Verilog симулятор Icarus Verilog , для компіляції та запуску Verilog коду на сервері. Для взаємодії клієнта та сервера був застосований AJAX підхід.

Отримана реалізація була протестована на різних вхідних описах апаратури на Verilog і видавала ті ж результати що й аналоги. Тому можна зробити висновки що дана реалізація працює справно.

В подальшому слід зосередити свою увагу на збільшенні та покращенні функціоналу даного веб-додатка. Отримана програмна реалізація може бути застосована для функціонально-логічного моделювання на мові Verilog. Також дана розробка може бути імплементована в певний начальний сервіс для полегшення входження студентів в моделювання та для демонстрації роботи певних модулів.

ПРЕЛІК ПОСИЛАНЬ

1. Блинов, И.Н. Java. Промышленное программирование : практ. пособие / И.Н. Блинов, В.С. Романчик. –Минск : УниверсалПресс, 2007. –704 с.
2. Офіційний сайт Icarus Verilog – Режим доступа : http://iverilog.icarus.com/ – Дата доступу: 07.06.2016.
3. Анопрієнко О. Я. Методика проектування Web-додатків з використанням платформи Java EE / Анопрієнко О. Я. // Інформатика і компютерні технології – 2010 – № VI – С. 160 - 165 .
4. Хорстманн, Кей С., Корнелл Гари . Java2. Основы, 7-е изд./ Хорстманн, Кей С., Корнелл Гари: Пер. с англ. – М.: И.Д. «Вильямс», 2006. – 896 с.
5. Budi Kurniawan . Java for the Web with Servlets, JSP, and EJB: A Developer's Guide to J2EE Solutions / Budi Kurniawan : New Riders Publishing , April 12 2002 – 976 p.
6. Э. Фримен, Э. Фримен. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. — П.: «Питер», 2010. — 656 с.
7. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2010. — 656 с.
8. Дейв Крейн, Бер Бибо, Джордон Сонневельд. Ajax in Practice. / Дейв Крейн, Бер Бибо, Джордон Сонневельд.— М.: Вильямс, 2007.
9. Методичні вказівки до виконання організаційно-економічного розділу дипломних проектів / Уклад. В.Є. Богданюк, К.В. Березовський, В.П. Пашін та ін. - К.: НТУУ "КПІ", 1999. - 66 с.
10. Пашин В.П. Функционально-стоимостный анализ конструкторско-технологических решений. - К.: РДЭНТП «Знание» УССР, 1989. - 22 с.
11. Пашин В.П. Управление качеством изделий на основе функционально-стоимостного анализа. - К.: «Технология и организация производства», 1989. - №1. с. 17-19.

12. Укрупненные нормы времени на разработку программных средств вычислительной техники. - М.: Экономика, 1988. - 62 с.