МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА ПРИКЛАДНОЇ ІНФОРМАТИКИ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

_____ __Гамаюн В.П.

(підпис) (ПІБ)

"_____"_____2021р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "БАКАЛАВР"

Тема: Розробка веб інтерфейсу для роботи з мережевими каталогами_____

Виконавець: _____ Пасько Денис Олександрович

(підпис) **(ПІБ)**

Керівник: _____ Гамаюн Володимир Петрович

(підпис) **(ПІБ)**

Нормоконтролер: _____ Боровик Володимир Миколайович

(підпис) **(ПІБ)**

Київ 2021

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. ДОСЛІДЖЕННЯ ТЕХНОЛОГІЇ ПОБУДОВИ ВЕБ-ДОДАТКІВ

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.

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.

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:

Безпосередня обробка запитів і формування відповідей.

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 творенням віртуальної лабораторії функціонально-логічного моделювання.

1.2Типи 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 длявизначення 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.

1.3 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 таможливості 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;
- Web – технології для побудови веб-додатків.

1.4 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 створенняскладних 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.

1.5 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. Класичний механізм взаємодії у Web показани на рисунку 1.1.

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 специфічнезастосування:

- 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.

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:

1. Обмежети використання мови сценаріїв JavaScript і використати технологію Java Апплетів.

2. 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.).

3. 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.

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.

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.

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.

1.6 Аналіз існуючих віртуальних каталогів

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 запуститийого, і отримати той же результат.

В даний час працює EDA Playground є проектом з відкритим вихідним кодомі дозволяє вільно моделювати та синтезувати EDA інструменти. Він підтримує кілька HDLs, таких як SystemVerilog, VHDL, MyHDL і Migen. Інженери можутьпрацювати 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 як показано на рисунку 1.3. Інтерфейс вихідних “waveforms” в EDA Playground показано на рисунку 1.4.

iVerilog.com Verilog Online simulator 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 моделюванняVerilog. 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 виглядає як показано на рисунку 1.5.

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

Codingground 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 як показано на рисунку 1.6:

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 харкактеристик 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. Порівняння сервісів показано в таблиці 1.1.

Таблиця 1.1

Порівняння сервісів .

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

Згідно таблиці 1.1 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.

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 ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ.

2.1 Постановка задачі

Задачею даної роботи є реалізація власного веб додатку мережевих каталогів, 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.

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:

1. 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.

2. Для кожної функції визначаються повні річні витрати й кількість робочих часів.

3. Для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

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

5. Постановка задачі техніко-економічного аналізу.

6. 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.

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.

Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи на рисунку 2.1. На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій, таблиця 2.1.

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

Таблиця 2.1.

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
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.

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.

Обґрунтування системи параметрів ПП

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.

Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у таблиці 2.2.

Таблиця 2.2.

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

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 передбачаєтакі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, якізвертаються 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 можутьне мати жодного уявлення про існування інших клієнтів. Загальний архітектурний шаблон показаний на рисунку 2.6.

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 наклієнті.

2.3. Опис використнаних технологій та бібліотек

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
- Internet Explorer версії 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.

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.

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.

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.

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.

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 4-ї 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.

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.

Висновки

В розділі 2 деталізована постановка завдання, виокремлені функціональні вимоги. Спроектована загальна архітектура виконуваного програмного додатку. Наведено короткий опис та обгрунтування використаних для розробки програмних засобів.

3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1. Розробка системи

Між провідним й підлеглими пристроями можуть встановлюватися канали зв'язку двох типів:

- Орієнтований на встановлення з'єднання синхронний канал (**Synchronous Connection Oriented - SCO**). Виділяє фіксовану смугу двохточковому з'єднанню, у якому задіяні провідний пристрій й один з підлеглих пристроїв. Провідний пристрій підтримує канал SCO шляхом використання через постійні інтервали зарезервованих слотів. Основна одиниця резервування - два послідовних слота (по одному в кожному напрямку передачі). Провідний пристрій може одночасно підтримувати до трьох каналів SCO, а підлеглий пристрій - два або три канали SCO. Пакети SCO ніколи не передаються повторно.
- Асинхронний канал без встановлення з'єднання (**Asynchronous Connectionless - ACL**). Багатоточковий канал між ведучим і всіма підлеглими пристроями пікомережі. У слотах, не зарезервованих для каналів SCO, провідний пристрій може обмінюватися пакетами з будь-яким підлеглим пристроєм, у тому числі й із пристроями, з якими вже встановлені канали SCO. Може існувати тільки єдиний канал ACL. Внаслідок різних збоїв більшість пакетів ACL передаються повторно.

В основному канали SCO використовуються для обміну терміновими даними, що вимагають гарантованої швидкості передачі, але не потребуючих гарантованої доставки. Наприклад, у декількох профілях Bluetooth використовуються оцифровані аудіодані, яким властива допустимість втрати даних. Гарантована швидкість передачі даних досягається за допомогою резервування для передачі певного числа слотів.

Резервування смуги неможливо, і доставка гарантується за допомогою використання схем виявлення помилок і повторної передачі. Підлеглому пристрою дозволяється повертати пакет ACL у слоті передачі від підлеглого пристрою провідному тоді й тільки тоді, коли він був переданий у попередньому слоті передачі від провідного пристрою підлеглому. Для каналів ACL визначені 1-слотові, 3-слотові й 5-слотові пакети. Дані можуть посилати незахищеними (хоча на більш високих рівнях може використовуватися схема ARQ) або захищеними кодом прямого виправлення помилок зі ступенем кодування 2/3. Максимальна швидкість передачі даних досягається при використанні 5-слотових незахищених пакетів й асиметричного розподілу пропускної здатності, це швидкість 721 Кбіт/с у прямому напрямку й 57,6 Кбіт/с - у зворотньому. Всі існуючі можливості наведені в таблиці 3.1 [17,18,19].

Формат, структура й типи пакетів Bluetooth

Формат всіх пакетів Bluetooth складається із трьох полів рисунок 3.1:

• Код доступу. Використовується для тимчасової синхронізації, компенсації зрушення, опитування й запиту.

• Заголовок. Використовується для визначення типу пакета і переносу керуючої інформації протоколу.

• Корисне навантаження. При наявності цього поля в ньому містяться мова або дані користувача й (у більшості випадків) заголовок корисного навантаження.

Доступні швидкості передачі даних по каналі ACL показані в таблиці 3.1.

Таблиця 3.1.

Тип	Симетрична схема	Асиметрична схема	
DM1	108,8	108,8	108,8
DH1	172,8	172,8	172,8
DM3	256,0	358,0	54,4

DH3	384,0	576,0	86,4
H5	432,6	721,0	57,6

Зупинимося докладніше на кожному з них. Існує три типи кодів доступу:

• Код доступу до каналу (channel access code - CAC). Визначає пікомережу (унікальний для пікомережі).

• Код доступу до пристрою (device access code - DAC). Використовується для вибіркового доступу й передачі наступних відгуків.

• Код доступу до опитування (inquiry access code - IAC), Використовується при опитуваннях.

У код доступу входять початкова комбінація бітів, що синхронізує слово й завершувач. Початкова комбінація бітів (preamble) використовується для компенсації постійної складової сигналу передачі й складається з послідовності 0101, якщо молодший (крайній лівий) біт у синхронізуючому слові - 0, і послідовності 1010, якщо молодший біт у синхронізуючому слові - 1. Подібним чином, завершувач (trailer) дорівнює 0101, якщо старший (крайній праворуч) біт у синхронізуючому слові - 1, і 1010, якщо старший біт у синхронізуючому слові – 0 [26,27].

64-бітове синхронізуюче слово складається із трьох компонентів, рисунок 3.2, і його варто розглянути докладніше.

Кожному пристрою Bluetooth привласнюється глобально унікальна 48-бітова адреса. 24 молодших біта називаються нижньою частиною адреси (lower address part - LAP) і використовуються для формування синхронізуючого слова. Для коду CAC використовується LAP провідного пристрою; для коду DAC - LAP пристрою, до якого йде обіг. Існує два різних коди IAC. Загальний код IAC (General LAC - GIAC) - це повідомлення загального опитування, що використовується для виявлення всіх пристроїв Bluetooth у радіусі дії; для цього є спеціальне зарезервоване значення LAP. Виділений IAC (Dedicated IAC - DIAC) належить виділеній групі пристроїв Bluetooth, які мають загальні характеристики, і в цьому випадку використовується адреса LAP, раніше визначена для цих характеристик [26,27].

Використовуючи відповідну адресу LAP, що синхронізує слово створюється в такий спосіб:

1. До LAP додається 6 біт 001101, якщо старший біт LAP дорівнює 0, і 110010, якщо старший біт - 1. У результаті виходить 7-бітова послідовність Баркера. Ціль введення послідовності Баркера - поліпшити автокореляційні властивості синхронізуючого слова.

2. Генерується 64-бітова шумоподібна послідовність, p_0, p_1 ..., p_{63}. Послідовність визначається рівнянням $P(X) = 1 + X + X^3$, і її можна реалізувати за допомогою лінійного регістра зрушення зі зворотним зв'язком. Початкове значення для генерації псевдовипадкової послідовності - 1000004.

3. До послідовності p_{34}, p_{35}, ..., p_{63} й 30-бітової послідовності, синтезованої на етапі 1, застосовується операція виключення АБО, що дозволяє скремблювати інформацію, видаляючи небажані регулярні послідовності.

4. Для скрембльованного блоку інформації генерується 34-бітовий код з корекцією помилок, він міститься перед блоком, у результаті чого формується 64-бітове кодове слово. Таким чином, маємо код (64, 30). Для генерації даного коду варто почати з коду БХЧ (63, 30). Потім визначається багаточлен, що породжує, $g(X) = (1 + X)g(X)$, де $g'(X)$ - це багаточлен, що породжує, коду БХЧ (63, 30). У результаті маємо бажаний 34-бітовий код.

5. До послідовностей p_0, p_1 ..., p_{63} й 64-бітової послідовності, синтезованої на етапі 4, застосовується операція виключення АБО. На цьому кроці дешифрується інформаційна частина кодового слова, так що передаються вихідні частина LAP і послідовність Баркера. На даному етапі також скремблюється блоковий код. Скремблювання інформаційної частини кодового слова на етапі 3 відбувається з метою посилення стійкості блокового коду до помилок. Наступне дешифрування дозволяє одержувачеві легко відновлювати LAP. З погляду специфікації, дешифрування 34-бітового коду з корекцією помилок усуває циклічність вихідного коду, завдяки чому поліпшуються спектральні характеристики передачі й автокореляційні

властивості [23,24].

Заголовок пакета. Формат заголовка всіх пакетів Bluetooth складається з таких полів:

• AM_ADDR. Нагадаємо, що пікомережа може містити не більше семи активних підлеглих пристроїв. 3-бітове поле AM_Addr містить адреси "активного режиму" (тимчасова адреса, привласнена даному підлеглому пристрою у даній пікомережі) одного з підлеглих пристроїв. Передача від ведучого до підлеглого пристрою містить дану адресу підлеглого пристрою, передача від підлеглого пристрою - адресу цього підлеглого пристрою. Значення 0 зарезервоване для передачі від провідного пристрою всім підлеглим пристроям у пікомережі.

• Тип. Визначає тип пакета, таблиця 3.2. Для контрольних пакетів зарезервовані чотири коди, загальні для каналів SCO й ACL. Інші типи пакетів використовуються для передачі інформації користувача. Пакети HV1, HV2, HV3 у каналах SCO застосовуються для передачі мови зі швидкістю 64 Кбіт/с. Відрізняються ці пакети ступенем захисту від помилок, що, у свою чергу, визначає, наскільки часто повинні посилати пакети для підтримки швидкості передачі даних 64 Кбіт/с. Пакет DV переносить і мову, і дані. Для каналів ACL визначені 6 різних пакетів. Ці пакети плюс пакет DM1 переносять користувальницькі дані з різним ступенем захисту від помилок і різною швидкістю передачі інформації таблиця 3.1. Існує ще один тип пакета, загальний для обох фізичних каналів; він складається тільки з коду доступу, а його розмір дорівнює 68 біт (не включаючи завершувач). Цей пакет позначається ID і використовується в процедурах опитування й доступу [35].

• Потік. Надає 1-бітовий механізм керування потоком тільки для трафіка ACL. При прийомі пакета з нульовим значенням даного поля станція повинна тимчасово зупинити передачу пакетів ACL по цьому каналі. При прийомі пакета зі значенням 1 передача може відновлятися.

• ARQN. Забезпечує 1-бітовий механізм підтвердження для трафіка

ACL, захищеного CRC таблиця 3.2. Після успішного прийому повертається підтвердження ACK (ARQN = 1); у противному випадку повертається негативне підтвердження NAK (ARQN = 0). При відсутності відповіді передбачається негативне підтвердження, а якщо отримано (або передбачається) негативне підтвердження, що відповідає пакет передається повторно [35].

Таблиця 3.2.

Типи пакетів Bluetooth

Код типу	Фізичний канал	Назва	Число слотів	Опис

Продовження таблиці 3.2.

0010	Загальний	FHS	1	Спеціальний керуючий пакет для визначення адреси пристрою й часу відправника. Використовується у відповіді провідного пристрою на запит, на опитування й при синхронізації перебудови частоти. Захист: FEC 2/3
0011	Загальний	DM1	1	Підтримує керуючі повідомлення, може також переносити дані користувача. Захист: 16-бітова CRC, FEC 2/3
0101	SCO	HV1	1	Переноситься 10 байт інформації; звичайно використовується для передачі мови зі швидкістю 64 Кбіт/с. Захист: FEC 1/3
0110	SCO	HV2	1	Переноситься 20 байт інформації; звичайно використовується для передачі мови зі швидкістю 64 Кбіт/с. Захист: FEC 2/3
0111	SCO	HV3	1	Переноситься 30 байт інформації; звичайно використовується для передачі мови зі

				швидкістю 64 Кбіт/с. Не захищається FEC
1000	SCO	DV	1	Дані (150 біт) і мова (50 біт). Поле даних захищається схемою FEC 2/3
1000	ACL	DH1	1	Переноситься 28 байт інформації плюс '16-бітове поле CRC. Не захищається FEC. Звичайно використовується для передачі високошвидкісних даних
1001	ACL	AUX1	1	Переноситься 30 байт інформації без захисту CRC або FEC. Звичайно використовується для передачі високошвидкісних даних .

Продовження таблиці 3.2.

1010	ACL	DM3	3	Переноситься 123 байт інформації плюс 16-бітове поле CRC, Захист: FEC 2/3.
1011	ACL	DH3	3	Переноситься 185 байт інформації плюс 16-бітове поле CRC. Не захищається FEC
1110	ACL	DM5	5	Переноситься 226 байт інформації плюс 16-бітове поле CRC. Захист: FEC 2/3.
1111	ACL	DH5	5	Переноситься 341 байт інформації плюс 16-бітове поле CRC. Не захищається FEC.

• SEQN. Надає 1-бітову схему послідовної нумерації. Передані пакети по черзі позначаються 1 або 0. Це потрібно адресатові для фільтрації повторних передач; якщо повторна передача викликана втратою підтвердження АСК, адресат одержує один пакет двічі.

• Контроль помилок у заголовку (Header Error Check - HIC). 8-бітовий код виявлення помилок. Використовується для захисту заголовка пакета.

Формат корисного навантаження. Для деяких типів пакетів вузькосмугова специфікація визначає формат поля корисного навантаження.

Якщо нею є мова, заголовок не визначений. У той же час заголовок визначається для всіх пакетів ACL і даних у пакеті DV SCO. Формат корисного навантаження для даних складається із трьох полів:

Заголовок корисного навантаження. 8-бітовий заголовок визначається для пакетів з одного слота, і 16-бітовий заголовок - для багатослотових пакетів.

Тіло корисного навантаження. Містить користувальницьку інформацію.

CRC. 16-бітовий код перевірки парності із залишком, що використовується для захисту всіх полів корисного навантаження, крім полів у пакетах AUX1.

Заголовок корисного навантаження (якщо він є) складається із трьох полів:

L_CH. Визначає логічний канал. Можливі наступні опції: повідомлення LMP (11); нефрагментоване повідомлення L2CAP або початок фрагментованого повідомлення L2CAP (10); продовження фрагментованого повідомлення L2CAP (01); інше (00).

Потік. Використовується для керування потоком на рівні L2CAP. Аналогічний механізму включення/вимикання, що пропонує поле потоку в заголовку пакета трафіка ACL.

Довжина. Число байтів даних у корисному навантаженні, крім заголовка корисного навантаження й CRC [19, 20].

3.2. Програмна реалізація комплексу

Коли відбувається старт програми-сервера відкривається так званий потік очікування, що підтверджує готовність сервера до встановлення зв‘язку з будь-яким ініціатором сеансу. Проте коли встановлено зв‘язок з одним клієнтом, іншим ініціаторам необхідно чекати, доки цей клієнт не припинить свою роботу та не звільнить потік очікування. Далі, встановивши зв‘язок із

клієнтом, сервер відсилає запит на отримання даних про логін і пароль користувача для автентифікації . Фактично, сервер отримує не сам пароль, а його хеш-код. Отримавши ці дані сервер надсилає запит до своєї бази даних щодо наявності у ній користувача з таким логіном. Якщо у базі такого користувача не існує, сервер припиняє сеанс зв‘язку з даним клієнтом, звільняючи тим самим потік очікування для інших користувачів. Якщо користувач зареєстрований у базі даних, автентифікація переходить до наступного етапу. Блок-схема алгоритму серверної частини програми показана на рисунку 3.3.

Рис. 3.3. Блок-схема алгоритму серверної частини програми

Після успішної перевірки логіну користувача програма-сервер переходить до співставлення отриманого від клієнта хешу з тим хешем клієнта, що зберігається у базі сервера. При повному співпаданні значень хеш-кодів сервер зберігає відомості про даного клієнта та цей сеанс роботи та відкриває доступ до необхідних клієнтові баз даних. Якщо хеш-коди не співпали, тобто користувачем було введено невірний пароль, сервер виводить на екран повідомлення про помилку, а Bluetooth з‘єднання із цим клієнтом розривається. При цьому сервер знову відкриває потік очікування для отримання запитів від інших клієнтів. . Блок-схема алгоритму клієнтської частини програми показана на рисунку 3.4.

Рис. 3.4. Блок-схема алгоритму клієнтської частини програми

Тепер розглянемо алгоритм роботи клієнтської частини розробленого програмного забезпечення.

Клієнт ініціює з‘єднання із сервером, відкриваючи віртуальний COM-порт та відправляючи запит щодо сеансу зв‘язку на сервер. Якщо сервер вільний, він у відповідь відсилає запит на логін та пароль користувача, що ініціював зв‘язок. Клієнт відсилає на сервер логін та хеш-код паролю користувача. Після того, як необхідні дані були надіслані, клієнт починає

опитувати сервер щодо того, чи співпало значення надісланого хеш-коду із тим значенням, що зберігалося на сервері. Якщо сервер надсилає негативну відповідь, клієнт повертається до режиму опитування сервера доти, доки не отримає позитивну відповідь або сервер не припинить сеанс зв‘язку. У випадку отримання позитивного результату проходження автентифікації , клієнт авторизується на сервері та отримує доступ до відповідних баз даних.

Припустимо, деякому користувачу необхідно отримати певну інформацію, що знаходиться на сервері. Для цього необхідно встановити між клієнтським та серверним ПК Bluetooth-з’єднання та об‘єднати ці ПК у бездротову мережу. Далі користувач запускає клієнтську частину розробленої системи захисту та проходить аутентифікацію на сервері. У разі успішного підтвердження автентичності користувача йому надається доступ до інформаційних ресурсів відповідно до встановленого адміністратором безпеки рівня його доступу. Програмно цей процес виглядає наступним чином.

Сеанс зв‘язку розпочинається з того, що програма-клієнт відкриває віртуальний COM-порт та ініціює з‘єднання із програмою сервером і очікує від нього відповіді.

```
namespace BluetoothClient
{
 public partial class MainForm : Form
 {
 private string Port;
 private string FileName = Application.StartupPath + "\\clients.xml";
 private string newData = string.Empty;
 private object response;
 SerialPort clientPort;
 public MainForm()
 {
 InitializeComponent();
```

```
try
{
Port = Properties.Settings.Default.comNumber;
spinEdit1.Value = decimal.Parse(Port);
response = new object();
```

В цей час, якщо програма-сервер завантажена, вона запускає так званий потік очікування. Суть його у тому, що сервер відкриває лише один потік для одного клієнта і доки відбувається обмін даними з однією програмою-клієнтом, інші повинні чекати. Намагаючись підключитись до сервера, вони отримують інформацію про те, що програма-сервер працює, але зайнята іншим клієнтом. Якщо ж протягом встановленого часу клієнт не посилає ніяких запитів на сервер, він автоматично перемикається на іншого клієнта.

```
Thread waitThread = new Thread(new ThreadStart(ReceiveLine));
currentThreadNumber = currentNumber;
waitThread.Start();
lock (response)
{
Monitor.Wait(response, 2000);
}
waitThread.Abort();
if (newData[currentNumber] != "<SYNC>")
{
success = false;
break;
}
```

Сам же запуск потоків для клієнтів програмно виглядає наступним чином:

```
private void StartButton_Click(object sender, EventArgs e)
{
receiveThread = new Thread [comNames.Count];
```

```
response = new object[comNames.Count];
newData = new string[comNames.Count];
mIncoming = new SerialPort[comNames.Count];
for (int i = 0; i < comNames.Count; i++)
{
mIncoming[i] = new SerialPort("COM" + comNames[i]);
receiveThread[i] = new Thread(new ThreadStart(ReceiveProc));
response[i] = new object();
newData[i] = string.Empty;
mIncoming[i].WriteBufferSize = 4096;
mIncoming[i].ReadBufferSize = 4096;
mIncoming[i].Encoding = Encoding.GetEncoding(1251);
mIncoming[i].Open();
currentThreadNumber = i;
receiveThread[i].Start();
}
```

Отже, дізнавшись, що програма-сервер завантажена та доступна для програми-клієнта, остання ініціює отримання інформації, що зберігається на сервері. В цьому випадку, отримавши запит від програми-клієнта, програма-сервер робить запит на логін та пароль користувача. Користувач вводить необхідні дані і сервер вилучає їх з програми-клієнта. В дійсності вилучається не сам пароль, а його хеш. Усі паролі хешуються SHA-256, що дає чергову гарантію цілісності та захищеності даних, що передаються. Програмно вилучення даних виглядає наступним чином:

```
private string GetHash(string inputString)
{
SHA256CryptoServiceProvider crypt =
new SHA256CryptoServiceProvider();
Encoding enc = Encoding.GetEncoding(1251);
byte[] input = enc.GetBytes (inputString);
```

```
byte[] output = crypt.ComputeHash(input);
return enc.GetString(output);
}
```

Після вилучення даних про користувача, тобто його логін та хеш-код паролю, програма-клієнт переходить у режим очікування відповіді, а серверна частина розробленого програмного забезпечення розпочинає процес перевірки автентичності цього користувача. Він складається з двох етапів. На першому етапі програма-сервер проводить пошук даного користувача у власній базі даних. Пошук ведеться за логіном. Якщо користувач з таким логіном знайдений у базі, програма-сервер переходить до другого етапу перевірки.

Другий етап являє собою перевірку хеш-коду паролю, що його ввів даний користувач. Якщо хеш-код, що зберігається у базі програми-сервера повністю збігається з хеш-кодом, отриманим з програми-клієнта, то програма-сервер відкриває доступ до інформаційних баз для даного користувача.

```
private void SendData(string lUser, string lPass)
{
try
{
int currentNumber = currentThreadNumber;
bool success = true;
DataRow[] clients = ClientsSet.Tables["Clients"].Select("Name = '" + lUser + "'");
if (clients.Length == 1)
{
string myHash = clients[0]["PasswordHash"].ToString();
string myStatus = clients[0]["Status"].ToString();
if (GetHash(lPass) == myHash) {
DataSet sendData = new DataSet();
```

```
sendData.Tables.Add("Clients");
sendData.Merge(ClientsSet.Tables["Clients"].Select(filter));
string dataAsString = sendData.GetXml();
```

Якщо логін користувача не знайдено у базі або якщо для існуючого у базі користувача хеш-код його паролю не збігся з еталонним хеш-кодом для цього користувача, то програма-сервер повідомляє користувача про помилку та розриває з‘єднання.

```
{
mIncoming[currentNumber]. WriteLine("<DENIDE>");
LogEdit.Text = LogEdit.Text + lUser + " ввів неправильний пароль." + "\r\n";
}
}
catch (Exception ex)
{
MessageBox.Show("Помилка:" + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error); }
```

У разі успішної авторизації користувача, програма-сервер надсилає програмі-клієнту відповідне повідомлення та завантажує у нього всю доступну, відповідно до рівня доступу цього користувача, інформаційну базу (у даному випадку це відомості про заробітну платню працівників підприємства):

```
private void ReadXml(string FileName)
{
ClientsSet.Clear();
ClientsSet.ReadXml(FileName);
bindingSource.DataSource = ClientsSet;
bindingSource.DataMember = "Clients";
mainGrid.DataSource = bindingSource;
cardView.Columns["Name"].Caption = "Ім'я";
```

```
cardView.Columns["PasswordHash"].Visible = false;
cardView.Columns["Status"].Caption = "Посада";
cardView.Columns["Salary"].Caption = "з/плата";
cardView.Columns["Image"].ColumnEdit=new
DevExpress.XtraEditors.Repository.RepositoryItemPictureEdit();
cardView.Columns["Image"].Caption = "Фото";
cardView.RefreshData();
}
```

При цьому програма-клієнт отримує повідомлення про успішно виконану операцію:

```
{
StreamWriter file = new StreamWriter(FileName);
file.Write(retData);
file.Close();
ReadXml(FileName);
MessageBox.Show("Інформацію успішно завантажено")
```

Якщо клієнт не пройшов аутентифікацію, то сервер надсилає йому повідомлення про те, що доступ для цього користувача закрито:

```
else
{
MessageBox.Show("Доступ закрито", "Відповідь сервера",
MessageBoxButtons.OK, MessageBoxIcon.Hand);
}
```

При будь-яких інших збійних ситуаціях програма-сервер також інформує клієнта за допомогою повідомлень:

```
catch (Exception ex)
{
MessageBox.Show("Помилка:" + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

Також до функціональних можливостей розробленого програмного забезпечення відноситься можливість сервера стежити за портом клієнта, оскільки для кожного нового клієнта відкривається новий власний віртуальний порт:

```
try
 {
 int currentNumber = currentThreadNumber;
 while (true)
 {
 if (mIncoming[currentNumber].BytesToRead > 0)
 {
 string lUser = mIncoming[currentNumber].ReadLine();
 string lPass = mIncoming[currentNumber].ReadLine();
 currentThreadNumber = currentNumber;
 Invoke(sendEvent, lUser, lPass);
 }
 }
 }
```

В свою чергу, клієнт має право вибору власного порту, але при цьому обов‘язково необхідно синхронізувати інформацію щодо портів власне на клієнті та на самому сервері. Для клієнта програмно це має такий вигляд:

```
try
 {
 string Value = spinEdit1.Value.ToString();
 Properties.Settings.Default.comNumber = Value;
 Properties.Settings.Default.Save();
 }
 catch (Exception ex)
 {
```

Для серверної частини програмного забезпечення це виглядає так:

```
try
 {
 string Value = comSpin.Value.ToString();
 comNames.Add(Value);
 listBoxControl1.Items.Add(Value);
 }
 catch (Exception ex)
 {
```

Однією із можливостей розробленого програмного забезпечення є аудит усіх процесів, що відбувалися під час роботи клієнтів із сервером. Дана можливість реалізована у вигляді звіту, що починає автоматично поповнюватися інформацією з моменту завантаження програми-сервера.

При бажанні цей звіт можна зберегти у вигляді простого текстового файлу.

```
try
 {
 SaveFileDialog sfDialog = new SaveFileDialog();
 sfDialog.Filter = "Text files | *.txt";
 sfDialog.DefaultExt = "*.txt";
 if (sfDialog.ShowDialog() == DialogResult.OK)
 {
 StreamWriter fileData = new StreamWriter(sfDialog.FileName);
 fileData.Write(LogEdit.Text);
 fileData.Close();
 }
 }
 catch (Exception ex)
 {
```

Для завершення сеансу роботи достатньо зупинити сервер. При цьому програма-сервер залишається повністю функціональною і у такому режимі

роботи можна проводити оновлення або редагування баз даних. Однак в цей час будь-які передачі даних між клієнтом та сервером неможливі. Про зупинку сервера сигналізує спеціальне системне повідомлення. Програмно процес зупинення серверу виглядає так:

```
private void StopButton_Click(object sender, EventArgs e)
{
if ((comNames != null) && (mIncoming != null) && (receiveThread != null))
{
for (int i = 0; i < comNames.Count; i++)
{
if (mIncoming[i] != null) mIncoming[i].Close();
if (receiveThread[i] != null) receiveThread[i].Abort();
}
}
applyButton.Enabled = true;
DeleteComButton.Enabled = true;
LogEdit.Text = LogEdit.Text + "Сервер зупинено" + "\r\n";
}
```

Треба також зазначити, що за допомогою серверної частини програми адміністратор безпеки може вільно додавати нових або видаляти існуючих користувачів з бази даних, може редагувати саму базу даних, додавати або видаляти віртуальні порти для клієнтів, а також вести повний аудит роботи данного програмного забезпечення за допомогою постійно автоматично оновлюваного звіту сервера, який при бажанні можна зберегти у вигляді текстового файлу.

3.3. Тестування розробленого додатку

Розглянемо інтерфейс даного програмного продукту.

Програма складається з клієнтської та серверної частин. Клієнтська частина програми являє собою головне вікно, логічно розділене на три функціональні блоки рисунок 3.5.

У першому блоці представлені поля для вводу логіна та паролю користувача, а також кнопка «Вхід» для отримання доступу до програми-сервера. Другий функціональний блок призначений для вибору нового або зміни існуючого віртуального COM-порту, через який буде відбуватися підключення до сервера та відбуватиметься інформаційний обмін із ним. Будь-які зміни номеру порту необхідно зафіксувати натисненням кнопки «Прийняти зміни». Третій блок являє собою інформаційне поле, куди при успішній авторизації завантажуються бази даних. Натиснувши кнопку «Customize» можна змінювати опції перегляду інформації у вікні.

Серверна частина програмного забезпечення являє собою головне вікно з трьома закладками у верхній частині: «З‘єднання», «Працівники» та «Додатково» рисунок 3.6.

При виборі закладки «З‘єднання» відкривається вікно, логічно розділене на два функціональні блоки. У верхньому блоці присутнє поле для введення номеру COM-порту (номер COM-порту у клієнта і сервера повинні співпадати) та кнопки «Додати» та «Видалити». Трохи нижче розташовані кнопки запуску чи зупинки серверу — відповідно «Старт» і «Стоп».

Другий функціональний блок представляє собою поле зі списком номерів використовуваних сервером та його клієнтами COM-портів.

Для того щоб додати до списку новий порт, необхідно у полі для введення номеру порту вказати новий номер порту, який ще не зайнятий іншими клієнтами, та натиснути кнопку «Додати». При успішному виконанні цієї операції номер нового порту буде додано до списку. Щоб видалити будь-який порт зі списку, необхідно виділити його у цьому списку та натиснути кнопку «Видалити». В результаті порт буде видалено зі списку сервера.

При виборі закладки «Працівники» відкривається вікно, логічно

розділене на два функціональні блоки рисунок 3.7.

Перший функціональний блок являє собою поле зі списком працівників підприємства, в якому може відображатися інформація про ім‘я працівника, пароль у зашифрованому вигляді, посада, заробітна платня та фото. Даний перелік можна відсортувати за кожним із зазначених параметрів у порядку зростання чи убування. Ці дані використовуються при перевірці автентичності користувача, що ініціює доступ до бази даних.

Другий функціональний блок містить поля для введення інформації у базу даних сервера. Треба зазначити, що обрана посада — «Працівник», «Сфера послуг», «Менеджер» та «Адміністратор», — мають різні рівні доступу до баз даних від найнижчого до найвищого рівня відповідно. Також у цьому блоці розміщені кнопки «Додати», «Змінити», «Видалити» та «Змінити фото». Для того щоб додати нового користувача у базу даних, необхідно заповнити всі наведені поля і натиснути кнопку «Додати» — нового користувача буде додано до списку. Для того, щоб видалити користувача зі списку необхідно виділити цього користувача у списку та натиснути «Видалити». При необхідності змінити дані про користувача, необхідно виділити цього користувача у списку та натиснути «Змінити» — всі заповнені поля будуть доступні для редагування.

Ознайомившись з інтерфейсом клієнтської та серверної частин розробленого програмного забезпечення, розглянемо далі приклад роботи даної захищеної Bluetooth- системи (у даному випадку йдеться про захист бази даних з відомостями про заробітну платню працівників підприємства).

Коли користувачу необхідно отримати доступ до інформаційних баз даних, що захищені даним програмним забезпеченням, йому треба завантажити програму-клієнт та ініціювати зв‘язок із сервером за допомогою бездротової технології Bluetooth. Для цього достатньо ввести у спеціальних полях свої логін та пароль і натиснути кнопку „Вхід” рисунок 3.8.

Рисунок 3.8 ілюструє вхід на сервер користувача з рівнем доступу адміністратора, тобто такий користувач має доступ до всіх баз даних, що зберігаються на сервері. Для наочності він має логін “Administrator”. Після натискання кнопки “Вхід”, клієнт відправляє запит на сервер. Сервер у відповідь зчитує дані про логін та пароль (хеш-код паролю) користувача, що намагається отримати доступ до баз даних.

Якщо логін дійсно відповідає даному користувачу, а хеш-код введеного паролю повністю співпадає із хеш-кодом паролю для цього користувача, що зберігається на самому сервері, то клієнт отримує доступ до інформаційних ресурсів сервера. Успішність виконання операції підтверджує системне повідомлення “Інформацію успішно завантажено”. У даному випадку користувач “Administrator” отримує доступ до всієї наявної інформації. На рисунку 3.9 видно, як програма-клієнт завантажила всю наявну на сервері інформацію, що складає інформаційні картки-відомості про працівників деякого підприємства, зокрема дані їхню про заробітну плату.

Для іншого користувача з правами доступу більш низькими, ніж адміністраторські, при успішній автентифікації та авторизації на сервері, буде доступно менше інформації, ніж більш високопоставленим користувачам, рисунок 3.10. Таким чином бачимо, що легальні користувачі, які пройшли додаткову аутентифікацію, легко отримують доступ до необхідної їм інформації через захищений канал бездротового зв‘язку типу Bluetooth.

При спробі несанкціонованого доступу до інформації за допомогою бездротового зв‘язку Bluetooth, розроблена система захисту ефективно протистоїть зловмиснику. Якщо припустити, що атакуючий успішно подолав бар‘єр у вигляді стандартної процедури автентифікації , характерної для стандартної системи захисту технології Bluetooth, то етап додаткової автентифікації має його зупинити та не дати можливості отримати доступ до

захищених даних. При введені ним незареєстрованого у базі даних сервера логіну чи паролю, сервер миттєво повідомляє про помилку, закриває доступ та розриває з‘єднання з цим клієнтом до наступного запиту. Клієнта про спробу несанкціонованого доступу сповіщає системне вікно рисунок 3.11. Невід‘ємною частиною процесу розробки будь-якого програмного забезпечення є його тестування. Зрозуміло, що програмні продукти, пов‘язані із захистом інформації, тестуються більш вимогливо та серйозно аніж інші. Розроблена система захисту комп‘ютерної мережі від несанкціонованого доступу за допомогою бездротового зв‘язку типу Bluetooth не стала виключенням і також була піддана тестуванню.

Для тестів було обрано два персональні комп‘ютери під управлінням операційних систем Windows і які включені до загальної локальної обчислювальної мережі. Також було використано два Bluetooth-адаптери зі стандартними драйверами для об‘єднання цих ПК у віртуальну приватну мережу WPAN. На один ПК було встановлено серверну частину розробленого програмного забезпечення, на другий ПК — відповідно клієнтську частина програми.

Оскільки недоліки захисту стандартної системи захисту технології Bluetooth добре відомі та вже піддавалися численним тестам багатьох спеціалістів, дане тестування стосувалося лише перевірки ефективності роботи розробленої додаткової системи автентифікації , призначеної для усунення недоліків стандартного механізму захисту Bluetooth.

Під час тестування було змодельовано кілька найбільш поширених ситуацій, які виникають при експлуатації програмного забезпечення такого типу, а в першу чергу — спроба несанкціонованого доступу до захищених баз даних.

3.4. Розгортання програмного продукту

У дипломній роботі при створенні програмного забезпечення використано стандартну базу даних Oracle, яка забезпечує максимальну швидкість і точність усіх виконуваних операцій і транзакцій.

Для доступу до даних відповідного додатку (наприклад, Excel, Lotus Notes) необхідні SQL*Net і ODBC (Open DataBase Connectivity) драйвери. Але, деякі програми, наприклад, Brio Explorer, працюють лише з SQL*Net. При використанні Excel для вибору інформації з БД КІС, користувач отримує доступ до даних у реальному часі, що забезпечує їм зручність у роботі.

Як зазначалось, створена КІС може функціонувати як монопольно, так у мережі. Якщо у якості ТЗ в такій мережі використовується IBM-сумісні ПК, то прикладна частина програмного забезпечення КІС в операційній системі Windows спілкується із сервером по мережі, використовуючи мережний протокол TCP/IP. При цьому, користувач використовує стандартний інтерфейс Windows Sockets для зв'язку з програмним забезпеченням протоколу TCP/IP. Це дозволяє їм КІС використовувати програмне забезпечення протоколу TCP/IP будь-якого постачальника, що підтримує стандарт Windows Sockets. Взаємозв’язки між про-
токолом TCP/IP і інтерфейсом Windows Sockets API представлені на рисунку 3.12.

Програмне забезпечення протоколу TCP/IP призначено для організації взаємодії користувача із сервером. Воно встановлюється і на комп'ютерах-користувачів, і на комп'ютерах-серверах.

Інтерфейс Windows Sockets по суті своєї є інтерфейсом (на ПК, що працюють під Windows) між прикладними додатками КСУ і ПЗ протоколу TCP/IP.

Інтерфейс Windows Sockets працює на різних Windows-платформах. ПО протоколу TCP/IP, що підтримує Windows Sockets, існує для усіх версій

Microsoft Windows, з якими можуть працювати користувачі КІС. Але, для різних версій Windows використовується різне прикладне ПО.

Якщо інтерфейс розроблено на базі Novell NetWare, для зв'язку в мережі використовує протокол SPX/IPX.

Для генерації складених користувачем звітів призначено SQL-генератор. Він використовує стандартні математичні операції і функції контролю форматів даних. Звіти можна складати за даними, які містяться у різних таблицях бази даних у різних розрізах. Іншим додатковим інструментальним засобом є Редактор форм виводу даних. Його призначено для зміни зовнішнього вигляду стандартних друкованих форм ІСУ, таких як підтвердження замовлень, зведені звіти і т.п., до виду, зручного для користувача.

У КІС існує можливість організувати безліч форм виводу даних з використанням специфічної для компанії термінології, що служить для спрощення перегляду і використання даних. Ці вікна виводу зв'язуються з конкретними групами користувачів, у вікнах виводиться лише необхідна їм інформація, а непотрібна залишається невидимої, у них використовується відповідна термінологія. Так, форма для перегляду зайнятості службовців для використання в бухгалтерії може значно відрізнятися від форми, що бачать менеджери проектів. Оскільки користувач у системі не прив'язаний до конкретного робочого місця (комп'ютеру), а користається лише спеціальним паролем для входу, ці форми виводу інформації зв'язані з конкретним користувачем чи групою користувачів і будуть супроводжувати йому завжди, за яким би комп'ютером він ні сидів.

Крім того, за допомогою спеціальної програми Active Scripting (створення сценаріїв) користувач може створювати сценарії, за яким будуть генеруватися звіти в ІСУ (КІС) і виконуватися будь-які зовнішні програми.

Висновки

В даному розділі детально описана розробка програмного засобу, протестовано виконання всіх функціональних вимог. Здійснено апробацію за загальний аналіз використання програмного засобу.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У даній атестаційній роботі було розглянуто проблему доступу до мережевих каталогів засобами бездротових комп‘ютерних мереж типу Bluetooth.

Відповідно до поставленої мети вирішені такі завдання:

1. проведено аналіз технології, архітектури протоколів й основних елементів організації мереж та визначити вимоги до механізмів обміну бездротових мереж типу Bluetooth, що дозволило визначити недоліки системи автентифікації Bluetooth та визначити подальші напрями удосконалення;
2. розроблено метод проведення додаткового механізму автентифікації Bluetooth-пристроїв;
3. розроблено алгоритм і програмний модуль, що дозволило провести тестування удосконаленого методу та реалізувати додатковий доступ до мережевих каталогів засобами бездротових мереж типу Bluetooth;
4. проведено тестування розробленого програмного модуля на основі удосконаленого методу, що дозволило провести процедуру блокування доступу при спробі НСД.

Таким чином, у ході виконання даної атестаційної роботи було розроблено програмний продукт, що реалізовує механізм доступу до мережевих каталогів та у поєднанні зі стандартними засобами захисту самої технології Bluetooth вирішує поставлену задачу створення ефективної системи захисту комп‘ютерної мережі від несанкціонованого доступу за допомогою бездротового зв‘язку типу Bluetooth.