МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАНЯ
КФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

на тему: «*КЛІЄНТ-СЕРВЕРНИЙ ДОДАТОК ДЛЯ ОБМІНУ ТЕКСТОВОЮ І ГРАФІЧНОЮ ІНФОРМАЦІЄЮ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ НА ОСНОВІ WEB-СЕРВІСУ*»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Ободяк В.К.**

Студента групи ІНЗ – 61с **Скоробогатов М.С.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 г.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІНЗ-61с спеціальності "Інформатика" заочної форми навчання Скоробогатова М.С.

Тема: " КЛІЄНТ-СЕРВЕРНИЙ ДОДАТОК ДЛЯ ОБМІНУ ТЕКСТОВОЇ І ГРАФІЧНОЇ ІНФОРМАЦІЄЮ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ НА ОСНОВІ WEB-СЕРВІСУ "

Затверджена наказом по СумДУ

№_____ от _____2020 г.

Зміст пояснювальної записки: 1) Огляд існуючих рішень; 2) постановка завдання й формування завдань дослідження; 3) Вибір середовища розробки; 5) Розробка проекту програми.

Дата видачі завдання "_____"_____2020 г.

Керівник випускної роботи _____ Ободяк В.К.

Завдання приняв до виконання _____

Реферат

Записка - 60 стр., 5 рис., 15 джерел, 1 дод.

Об'єкт дослідження – процесс обміну текстів і графічної інформації на основі web-сервісів

Мета роботи - створення клієнт-серверного додатка для обміну текстів і графічною інформацією в реальному часі

Методологія проведення дослідження – Розроблено додаток, що забезпечує можливість мережевого спілкування. Цей додаток дозволяє використовувати широкий набір інструментів для малювання, підтримує роботу з графічним планшетом. Програми розширює можливості мережевого спілкування за рахунок надання користувачам унікальну можливість обміну не тільки текстової, але і графічної інформацією.

Основні конструктивні і техніко-експлуатаційні характеристики – застосовані сучасні технології побудови мережевих додатків на ос-нове платформи Microsoft .NET. Програма в більшості випадків не потребує настойки брандмауера і підтримує введення з графічного планшета.

Рекомендації по впровадженню – рекомендується використовувати в складі web-сайтів і як окремий додаток не тільки для спілкування по мережі, але і для організації навчального процесу в дитячих установах, як інструмент для створення і спостереження за процесом створення графічних зображення-ний в співтоваристві онлайн художників, для дистанційного взаємодії розробників дизайну комп'ютерних ігор.

ЗМІСТ

ВСТУП

Сучасне суспільство немислимо без засобів зв'язку: експрес-пошта, телефонний і відео зв'язок і, мабуть, самий багатофункціональний - зв'язок за допомогою інтернету. Електронна пошта, VoIP, відео-та аудіоконференції[1], чати, форуми, служби миттєвих повідомлень - всі ці можливості роблять спілкування сучасної людини різноманітним за змістом і широтою контактів. Електронний зв'язок може забезпечувати комунікацію в режимі реального часу, при одночасній участі співрозмовників у процесі обговорення, або в асинхронному режимі, є взаємодією між користувачами в міру їх появи в мережі.

При розгляді спілкування в мережі традиційно виділяють обмін текстів повідомленнями, статичними зображеннями, аудіо та відео. У деяких програмах використовується комбінований підхід.

Останнім часом широкого поширення набувають webконференції , які використовуються для проведення групових нарад та «живих» презентацій в Інтернеті. В недалекому минулому цей термін частіше використовувався для позначення групових обговорень за допомогою текстів повідомлень в межах дощок оголошень, але в даний час ним користуються, маючи на увазі саме «живі» зборів, тоді як спілкування на web-сайтах називають форумами або дошками повідомлень. Типовими можливостями програмного забезпечення web-конференцій є поділ додатків, слайдові презентації, поділ екрану користувача і можливість вказати на фрагменти, які потребують уваги. Все це доповнюється телефонною розмовою (по звичайній телефонній мережі або через інтернет), і, нерідко, відео зображенням. Програми, реалізуючі переважно обмін аудіо та відеоданими, виробляють великий обсяг мережевого трафіку і вимагають швидкісного підключення до інтернету.

Як правило, невелика увага приділяється обміну графічною інформацією. Часто в процесі обговорення ідеї або при спробі пояснити його нить співрозмовнику складну схему, внаслідок обмежених описових можливостей людини, краще і простіше доповнити свої слова малюнком, хоча б у вигляді начерку. Добре, якщо співрозмовник знаходиться поруч і під рукою є олівець і аркуш паперу, але що робити, якщо ви перебуваєте в різних точках земної кулі і спілкуєтеся через інтернет? В цьому випадку дуже корисною виявиться програма, яка забезпечує передачу співрозмовникам графічного введення користувача в реальному часі. Також необхідно, щоб вона дозволяла іншим користувачам не тільки спостерігати за процесом малювання, а й брати в ньому активну участь. Найчастіше важливу роль при спільному малюванні або при спостереженні за малюванням іншої людини грає не тільки сама передача зображення, але і послідовність створення зображення. Колективне малювання розширює можливості спільної творчості, робить розширення ідей більш продуктивним, дозволяє в повній мірі проявити уяву і, в цілому, сприяє креативноій взаємодії.

Найбільш популярні на сьогоднішній день програмні продукти для мережевого спілкування, в яких реалізована можливість колективного малювання, надають їй другорядне значення. Одні надають тільки частину з 4 необхідних для повноцінного малювання функцій. В інших виникають труднощі із забезпеченням з'єднання, взаємодій-наслідком з міжмережевими екранами, одночасної комунікацією деяких користувачів. У жодній з програм немає підтримки графічного планшета, хоча його використання істотно спрощує введення користування земельними ділянками, робить його більш природним і реалістичним. Крім того, в цих програмах відсутне зручне сполучення обміну текстовими повідомленнями і графічним введенням. Жодна з них не надає достатнього обсягу коштів для художнього вираження.

Дана робота присвячена розробці та створенню зручного і корисного в першу чергу з точки зору користувача додатку для мережевого спілкування. До даного додатку були пред'явлені такі необхідні вимоги як, можливість одночасного обміну текстової та графічної інформацією, сприяння колективної творчості, спостереження за процесом малювання в динаміці. При цьому було необхідно врахувати недоліки аналогічних програм.

1.ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

На даний момент програми для мережевого спілкування представлені в широкому асортименті, різноманітні за функціональним можливостям і цілям, для яких застосовуються. При цьому на сьогоднішній день на ринку програмного забезпечення не представлено якісних і зручних програм для мережевого спілкування, які б задовольняли вимогам більшості користувачів. Крім того, асортимент програм для мережевого спілкування, які надають можливість малювання, представлений бідно. У той же час багато які з представлених програмних продуктів, не дають якісних можливостей малювання. Розглянемо найбільш поширені, популярні продукти, які надають пріоритетну можливість колективного спілкування, в тому числі можливість не тільки текстового, а й візуального (графічного) обміну інформацією.

1.1. Network Assistant

Network Assistant [3] - програма для мережевого спілкування, розрахована на користування в умовах домашньої або невеликої офісної локальної мережі.Взаємодія з мережею засновано здійснюється наступним чином: всі комп'ютери локальної мережі, на яких в конкретний момент часу запущена ця програма, утворюють мережеве оточення Network Assistant. Комп'ютерами в цьому оточенні можуть використовуватися всі надані додатком можливості для обміну інформацією, наприклад, багатоканальний чат, загальна дошка для малювання, обмін повідомленнями (заснованими на WinPopup), розміщенням ого-лений і т.п. Network Assistant не вимагає підключення до мережі інтернет і виділеного сервера, обмін повідомленнями здійснюється за допомогою про-токола UDP в рамках мережевого оточення, при цьому використовуються широкомовні і multicast повідомлення. Однак при цьому, тільки права установки програми захисту мережевого дає можливість спілкуватися за допомогою Network Assistant, тобто якщо користувач не має прав адміністратора на своєму комп'ютері, то він не може налаштувати firewall, а,

отже, і використовувати Network Assistant.

У багатокористувацькому чаті можливе створення каналів-кімнат різного типу: звичайних, захищених паролем, загальних для всіх і особистих (один на один).

Далі розглянемо можливості, які представляє багатокористувацька дошка для малювання. Дошка є спільною для всіх користувачів оточення. У даній програмі бідно представлений набір інструментів: олівець, заливка, текст і геометричні примітиви (лінія, еліпс, прямокутник).

Є можливість включити прив'язку до сітки для більш точного малювання. Відсутність гумки або можливості скасувати останню дію є безсумнівним мінусом даного програмного продукту, так як навіть непрофесіонали не можуть обійтися без цих інструментів при малюванні. Незважаючи на невеликий набір інструментів, використання дошки дозволяє доповнити бесіду схематичним зображенням, начерком або вставленим з буфера обміну малюнком.

При роботі з Network Assistant у користувача є можливість задавати свій поточний стан, в залежності від якого змінюється поведінка програми: прийом повідомлень, реакція на них, звукові сповіщення тощо. Кожен користувач може надати докладну інформацію про себе, включаючи ім'я, прізвище, контакти, фотографію тощо. Статус користувача в мережі змінюється як вручну, так і автоматично, в відповідності до різних подій, наприклад, включенням і відключенням комп'ютера від мережі , простоєм комп'ютера протягом деякого проміжку часу або запуском зберігача екрана.

Програма також має наступні можливості: передача файлів, перегляд знімків екрану, буфера обміну і статистики використання локального або віддаленого комп'ютера, підтримує кілька мов. Набір інструментів в Network Assistant таким чином, про програмний продукт Network Assistant можна зробити наступні висновки:

1. Широкий набір функцій дозволяє налагоджувати комплексну комунікацію (обмін файлами, інформація про користувача, передача текстової інформації і т.д.)

2. У програмі є можливість створення каналів-кімнат, що дозволяє користувачам спілкуватися в більш вузькому, орієнтованому на інтереси, колі. Незважаючи на це в даній програмі слід зазначити наступні істотні недоліки: • Відсутність гумки або скасування останньої дії істотно знижує зручність обміну інформацією і змушує користувачів кожен раз малювати поверх вже існуючого малюнка.

• Невеликий набір інструментів знижує якість одержуваних зображень і не дозволяє користувачам представити свої думки в адекватної графічній формі.

• Програма дозволяє спілкуватися тільки по локальній мережі і не може бути використана в мережі Інтернет

1.2 Imagination Cubed

Imagination Cubed [4] поряд з Network Assistant є програмою для спільного малювання. Реалізація даного продукту як flash-додатку, вбудованого в web-сторінку, дозволяє цього продукту бути доступним з будь-якого підключеного до Інтернету комп'ютера, що має встановлений браузер. Для зв'язку з компаньйоном доведеться відправити йому запрошення по електронній пошті або AOL службі миттєвих повідомлень, що збільшує час, необхідний для виходу на зв'язок, і не підходить для швидкої бесіди. Тільки коректна посилання є доступом до дошки малювання. Інтерфейс і інструментарій в Imagination Cubed

Набір інструментів для малювання в даному продукті представлений більш широко, ніж в попередньому. Доступні олівець, геометричні форми, штампи - відбитки різних форм, лінія, текст. Все инструменти мають настройки розміру, кольору, безліч форм фігури або лінії.

Доступний і зручний інтерфейс дозволяє легко малювати і спілкуватись за допомогою малювання, однак текстовий чат представлений занадто маленьким вікном на невигідному місці, що серйозно ускладнює текст спілкування.

Результати роботи з програмою можна вислати по електронній пошті або роздрукувати. Корисною функцією є повтор послідовності створення малюнка, дозволяє крок за кроком простежити за формування зображення від початку і до кінця.

Одночасно можуть бути присутніми до 4-х чоловік, але вже при трьох, навіть при наявності швидкісного підключення, спостерігаються помітні проблеми з швидкодією: «випадають» фрагменти ліній, плавна крива не зображується плавно, сповільнюється реакція сервера по відображенню фігур. При тривалості сесії в 10 хвилин і помірної активності 3х чоловік, кожен виробляє близько 100 Кб трафіку.

В цілому, додаток підходить для розважальних цілей, колективного творчості, і дозволяє, наприклад, створити анімовану відкритку для привітання і здивувати кого-небудь своїми художніми здібностями. Таким чином, для даного програмного продукту можна відзначити ряд плюсів і мінусів:

Плюси:

- Широкий набір інструментів.
- Продукт доступний без наявності встановленого програмного забезпечення і використовується за допомогою інтернет-браузера і не вимагає настройки firewall.

Мінуси:

- Сеанс здійснюється тільки при наявності коректної посилання, отриманої по електронній пошті або за допомогою служби миттєвих повідомлень, що виключає можливість швидкого спілкування і спілкування з незнайомими людьми.
- Текстовий чат представлений невигідно і незручно.

• Imagination Cubed розрахований максимально на чотирьох чоловік, що робить неможливим навіть перегляд дошки малювання великою кількістю користувачів.

1.3 Microsoft Windows Messenger

Microsoft Windows Messenger [5] входить в комплект поставки Microsoft Windows XP, тому широко доступний і поширений. Для виконання необхідна реєстрація в Microsoft Passport Network і, таким чином, наявність зареєстрованого там адреси електронної пошти. Також необхідно підключення до .NET Messenger service. Windows Messenger являє собою додаток для обміну миттєвими спілкуваннями, голосової та відео комунікації, відправлення файлів, а також спільного доступу до додатків.

Програма дозволяє обмінюватися повідомленнями і інший інформацією з віддаленим користувачем в режимі «один-на-один». Можливість графічного введення забезпечує додаток «дошка», для доступу до якої необхідно запросити співрозмовника і отримати від нього під-підтвердженням. Після підтвердження між вузлами відбувається пряме з'єднання по протоколу UDP. Також як і Network Assistant Microsoft Windows Messenger вимагає правильного налаштування 9 брандмауера, тобто затрудняет використання даної програми користувачами, які не мають-ми на робочому місці привілеїв адміністратора.

Графічні можливості є урізані можливості Microsoft Paint, які не мають в належній мірі коштів художнього вираження. У дошці малювання Windows Messenger немає масштабування малюнка, внаслідок чого у користувача немає можливості промалювати дрібні деталі. Не можна перемістити зображення і розташувати малюнок на потрібному місці вікна, що також ускладнює малювання. До віз-можливостям Microsoft Paint в дошці малювання Microsoft Windows Messenger додалися лише такі функції як блокування області введення і можливість використання декількох сторінок для малювання.

Таким чином, одним їх найбільш значущих переваг даного програмного продукту є те, що він широко доступний і поставляється з Windows XP.

До мінусів Windows Messenger можна віднести наступне:

1. Можливість малювання надається тільки в режимі один-на-один, хоча аналогічні програми дозволяють використовувати дошку декільком користувачам одночасно.

2. Для малювання на дошці використовується стандартний набір інструментів Microsoft Paint, тобто програма успадковує всі недоліки програми Paint, в тому числі неможливість плавного масштабування малюнка і розташування його в місці відмінному від лівого верхнього кута.

3. Вимагає налаштування firewall.

1.4. Постановка задачі

За результатми проведеного аналізу недоліків існуючих додатків для обміну графічною і текстовою інформацією:

1. Обрати програмне забезпечення для створення додатків;
2. Вибрати метод створення додатку;
3. Вибрати метод запровадження можливості обміну графічною інофрмацією в текстовому чаті.
4. Розробити програмне забезпечення і протестувати розроблену інформаційну систему.

2. ВИБІР СЕРЕДОВИЩА РОЗРОБКИ

Обрана середовище має поєднувати в собі простоту і легкість розробки, і хорошу підтримку методології об'єктно-орієнтованого підходу. Далі розглянемо деякі засоби для розробки.

2.1. Мова програмування C#

C# (кажучи російською, сі Шарп) - це об'єктно-орієнтоване програмування. Вона була розроблена в 2001 році, інженерами під керівництвом Андерса Хейлсберг в компанії Microsoft. На даний час існує 4 версії мови «Сі Шарп».

Назва «Сі Шарп» (від англ. Sharp - дієз) походить від музичної нотації, де знак дієз, що додається до основного позначення ноти, означає підвищення відповідного цій ноті звуку на півтон. Це аналогічно назвою мови C ++, де «++» позначає, що змінна повинна бути збільшена на 1.

До числа принципово важливих рішень, які реалізовані корпорацією Microsoft в мові програмування C #, можна віднести наступні: - компонентно-орієнтований підхід до програмування (який характерний і для ідеології Microsoft .NET в цілому); - властивості як засіб інкапсуляції даних (характерно також в цілому для ООП); - обробка подій (маються розширення, в тому числі в частині обробки винятків, зокрема, оператор try); - уніфікована система типізації (відповідає ідеології Microsoft .NET в цілому); - делегати (delegate - розвиток покажчика на функцію в мовах C і C ++); - індексатори (indexer - оператори індексу для звернення до елементів класу-контейнера); - перевантажені оператори (розвиток ООП); - оператор foreach (обробка всіх елементів класів-колекцій, аналог Visual Basic); - механізми boxing і unboxing для перетворення типів; - атрибути (засіб оперування метаданими в COM-моделі); - прямокутні масиви (набір елементів з доступом за номером індексу і однаковою кількістю стовпців і рядків).

Переваги сі-шарпа по книзі Білліг:

1. C # створювався паралельно з каркасом Framework .Net враховуючи всі його можливості - як FCL, так і CLR;

2. C # є об'єктно-орієнтованою мовою, де типи, вбудовані в мову, представлені класами;

3. C # є потужною об'єктною мовою з можливостями успадкування та універсалізації;

4. C # є спадкоємцем мов C / C ++, зберігаючи кращі риси цих популярних мов програмування. Загальний з цими мовами синтаксис, знайомі оператори мови полегшують перехід програмістів від C ++ до C #;

5. Зберігши основні риси свого великого батька, мова стала простіше і надійніше. Простота і надійність, головним чином, пов'язані з тим, що на C # хоча і допускаються, але не заохочуються такі небезпечні властивості C ++ як покажчики, адресація, розіменування, адресна арифметика;

6. Завдяки каркасу Framework .Net, який став надбудовою над операційною системою, програмісти C # отримують ті ж переваги роботи з віртуальною машиною, що і програмісти Java. Ефективність коду навіть підвищується, оскільки виконавче середовище CLR є компілятор проміжного мови, в той час як віртуальна Java-машина є інтерпретатором байт-коду;

7. Потужна бібліотека каркаса підтримує зручність побудови різних типів додатків на C#, дозволяючи легко будувати Web-служби, інші види компонентів, досить просто зберігати і отримувати інформацію з бази даних і інших сховищ даних;

8. Реалізація, що поєднує побудова надійного і ефективного коду, є важливим фактором, що сприяє успіху C #.

2.2 Microsoft Visual Studio.

Є засобом швидкої розробки додатків (RAD), що дозволяє створювати веб-додатки і веб-служби наступного покоління. Visual Studio.NET дозволяє розробникам швидко створювати широкомасштабні веб-додатки для будь-яких пристроїв і будь-яких платформ.

Крім того, Visual Studio.NET [10] повністю інтегрована із середовищем розробки Microsoft.NET Framework, надаючи підтримку декількох мов програмування і автоматичне вирішення багатьох завдань програмування, звільняючи розробників для швидкого створення веб-додатків за допомогою тієї мови, який вони вважають найбільш підходящим. Visual Studio.NET включає єдину інтегровану середу розробки із засобами RAD для побудови веб-додатків і бізнес-логіки проміжного рівня, а також RAD XML-засобами для роботи з даними.

2.3 Omnis Studio 4.2.0.8 [14]

· Кроссплатформенная об'єктно-орієнтована інтегрована середовище швидкої розробки - IDE RAD

• Інтегрована по ODBC / JDBC з СУБД Oracle, Sybase, DB2, Informix, MySQL і MS SQL Server

• Платформи development / runtime: Microsoft Windows, Linux, Mac

• Дозволяє розробляти універсальний - графічний / Web інтер-фейс користувача

• Підтримує технології XML, .NET, JavaИз вышеперечисленных сред разработки, выбран Microsoft Visual Studio 2008 из-за широкого набора средств разработки и простоты интерфейса.

3. РОЗРОБКА ПРОЕКТУ ПРОГРАМИ

Розглянувши наявні аналоги, ми виробили список вимог, необхідних для реалізації програми графічної мережевої комунікації. Розглянемо технологічні рішення, що застосовуються для реалізації цих вимог.

3.1 Клієнт-серверна архітектура

Додаток є розрахованим на багато користувачів і призначене для використання в мережі. Традиційно для такого роду додатків застосовуються два підходи: централізоване управління передачею повідомлень між користувачами або розподіл функцій синхронізації повідомлень між усіма учасниками мережевої взаємодії.

Перший підхід є, по суті, застосуванням технології «клієнт-сервер». При цьому підході створюється виділений сервер, який обслуговує запити клієнтів, зберігає дані для управління станом клієнтів і виконує синхронізацію даних між ними. Він відповідає за прийом даних від кожного клієнта і поширення цих даних між іншими. В цьому випадку всі клієнти є рівноправними щодо розподілу навантаження і взаємодіють виключно з сервером.

Іноді серверна частина може включатися в клієнтську програму. В цьому випадку відбувається взаємодія типу «клієнт-клієнт». Тоді один з клієнтів за попередньою домовленістю несе на собі функції сервера, в результаті будучи і клієнтом, і сервером. Така реалізація в разі великої кількості користувачів призводить до зайвого навантаження на одного з учасників. Це, безсумнівно, є вузьким місцем такого підходу. Ще одним з його недоліків є «обваження» клієнтської частини. На додаток до сказаного зазначимо, що поєднання функцій клієнта і сервера в одному додатку може бути виправдане при реалізації підключення «точка-точка», коли обидві сторони в різний час можуть мінятися ролями.

Другий підхід заснований на мережевій взаємодії без використання сервера, як такого. У цьому випадку кожен учасник комунікаційного процесу сам здійснює розсилку даних всім іншим учасникам, наприклад, за допомогою широкомовлення. Це дозволяє заощадити необхідні на забезпечення підтримки сервера ресурси. Але така реалізація, в основному, підходить для комунікації в умовах локальної мережі.

Наше додаток має дозволяти користувачам спілкуватися за допомогою інтернету. У цих умовах для забезпечення комунікації безлічі користувачів застосування клієнт-серверної технології забезпечить наступні переваги:

• мінімізація навантаження на клієнта,

• відсутність необхідності встановлювати пряме з'єднання між клієнтами,

• централізація діяльності з обслуговування клієнтів в одному місці.

3.2 Загальна структура програми

Для повного опису загальної структури програми вдамося до опису платформи Microsoft .Net, так як ця платформа надає гнучкі засоби для розробки розподілених додатків. У їх числі CLR (Common Language Runtime), єдине середовище виконання, заснована на виконанні коду, реалізованого на різних мовах програмування. При цьому використовується перетворення такого коду в платформонезавісимость проміжна мова - MSIL (Microsoft Intermediate Language), а компіляція під конкретну платформу відбувається на вимогу, в момент виконання. Також звернемо увагу на наявність потужної бібліотеки класів .Net Framework, в якій реалізовано безліч функцій, необхідних для створення широкого спектра пріложеній_ . Архітектура .Net пропонує різнопланові засоби виконання мережевої взаємодії.

У .Net серверна частина нашого застосування може бути реалізована у вигляді віддаленого об'єкта, який використовується безліччю клієнтів. Цей об'єкт буде забезпечувати поділ і синхронізацію даних між клієнтами.

.Net Remoting - технологія віддаленого взаємодії, яка дозволяє звертатися до об'єктів, розташованих в різних доменах додатків і навіть на різних машинах. Об'єкти клієнту передаються за значенням або за посиланням. Для передачі об'єкта за значенням створюється його повна копія, яка потім відправляється клієнтові. При зверненні до віддаленого об'єкту за посиланням на стороні клієнта створюється об'єкт-проксі, використовуючи який, клієнт отримує доступ до необхідного об'єкту прозорим способом, минаючи деталі взаємодії з віддаленим додатком.

Web-сервіси (XML Web Services) теж можуть розглядатися як віддалені об'екти_ . У широкому сенсі, web-сервіс - це клас, розташований на web-сервері і використовує для надання доступу до своїх методів HTTP - як протокол передачі даних і XML - як їх формат. Клієнт може звертатися до web-сервісу за допомогою URL і взаємодіяти з ним за допомогою проксі-об'єкта. Проксі містить всі ті ж методи, що і оригінальний клас, але його реалізація просто перенаправляє виклики методів оригінальному об'єкту.

Для динамічного створення проксі-об'єктів в .Net використовуються метадані. Зокрема, web-служба може надати клієнту свій опис на мові WSDL (Web Services Description Language). Будь-який клієнт, що володіє можливістю розуміти і направляти SOAP-запити відповідно до цього опису, може викликати методи web-сервісу і взаємодіяти з ним за допомогою SOAP_. В якості альтернативи динамічному створенню проксі можна статично згенерувати вихідні файли, що описують віддалений web-сервіс, і включити їх до складу клієнтської програми.

Web-сервіс .Net може бути розміщений на web-сервері IIS (Рис 1.1). Клієнт має опис web-служби на WSDL і викликає її методи, виконуючи SOAP запити відповідно до цього опису. IIS обробляє запити, направляючи їх екземпляру web-сервісу за допомогою ISAPI розширень.

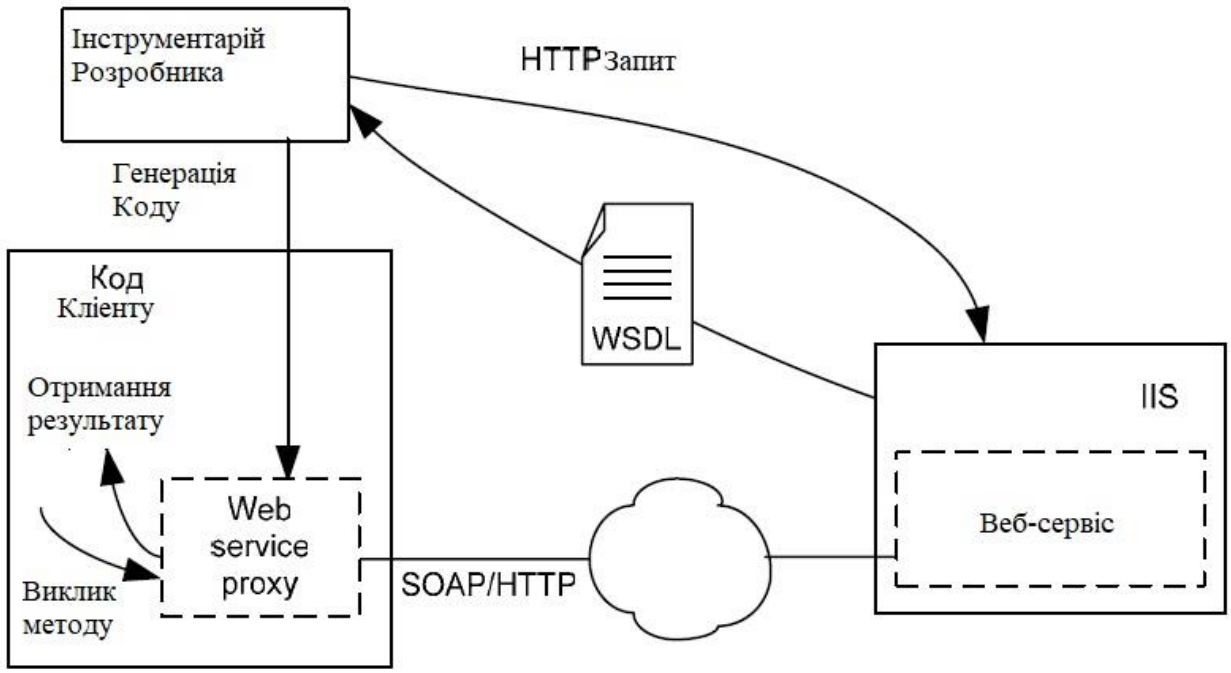

Рисунок 3.1 - Статична генерація проксі-класу web-сервісу

Така конфігурація підходить для взаємодії з клієнтом, що знаходяться за фаєрволом, а також дозволяє використовувати широку інфраструктуру .Net Framework.

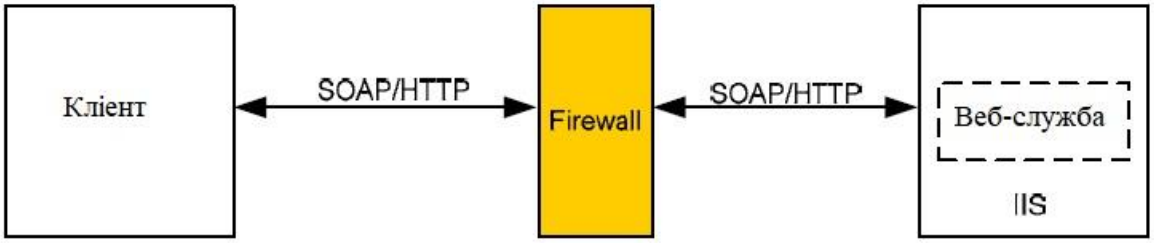

Рисунок 3.2 - Приклад виклику web-сервісу через firewall

Реалізація серверної частини в якості web-служби дозволяє не турбуватися про специфіку реалізації клієнтської частини, операційної системи користувача. Єдиними вимогами до клієнта з точки зору взаємодії з web-службою є реалізація протоколу HTTP в клієнтської середовищі і його здатність «розуміти» XML. Ще одним плюсом використання web-служби можна назвати підхід до клієнт-серверному взаємодії як до віддаленого виклику методів класу. Це дозволяє абстрагуватися від деталей реалізації мережевої частини і зосередитися на функціональної частини.

Клієнтську частину реалізуємо у вигляді Windows додатку на платформі .Net. Це дає можливість скористатися потужним інструментарієм розробника, бібліотеками .Net, підтримкою генерації проксі-класу web-служби, складанням сміття та іншими зручностями Microsoft. Хоча, як уже згадувалося, длявзаємодії з web-службою немає особливих обмежень на мову і платформу реалізації клієнтської частини, крім розуміння HTTP і XML.

3.3. Взаємодія клієнта і сервера

3.3.1. Етапи взаємодії

Взаємодія між клієнтом і сервером засноване на виклики методів сервера і обміні інформаційними повідомленнями, які організовуються в чергу. Групи користувачів логічно об'єднуються для спілкування в кімнати. Сервер містить кілька черг повідомлень, окремих для кожної кімнати і забезпечує синхронізацію між клієнтами в її межах. Клієнт, який взаємодіє з сервером, проходить 4 етапи: (див. Рисунок 3.3)

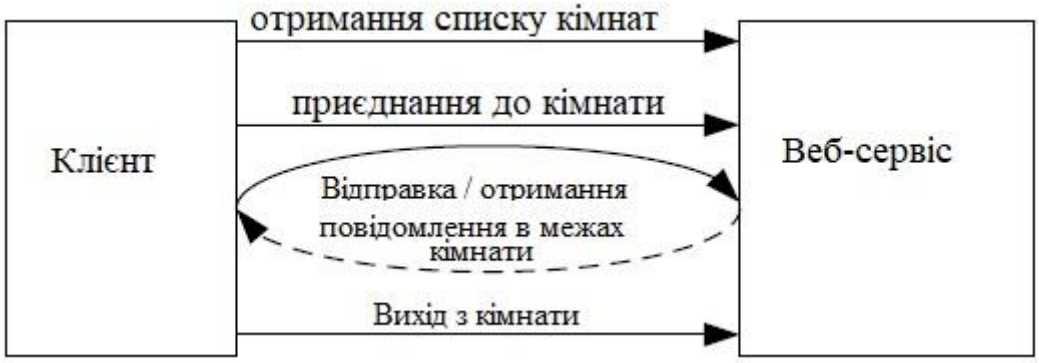

Рисунок 3.3 - Основні етапи взаємодії клієнта і півночі

1. а) Запит на отримання списку кімнат.
   б) Отримання відповіді зі списком кімнат.
2. а) Запит на приєднання до кімнати.
   б) Отримання відповіді на запит приєднання до кімнати.
3. Цикл відправки / отримання повідомлень в межах кімнати.

4. а) Запит на вихід з кімнати.

   б) Отримання відповіді на запит виходу з кімнати.

Перед приєднанням до конкретної кімнаті клієнт запитує електронної пошти їх список і деяку пов'язану з ними інформацію, грунтуючись на якій він приймає рішення про приєднання до кімнати або підключається до іншого сервера. Так як в цей момент ще не бере участь в процесі комунікації з іншими користувачами, ця операція має ізольований характер і ніяк не пов'язана з попередніми запитами клієнта. Тому сервера не потрібно знати нічого про стан клієнта.

Згадаймо, що клієнт і сервер взаємодіють за допомогою протоколу HTTP, в рамках якого між ними не підтримується постійного з'єднання. З'єднання існує тільки під час виконання запиту і повернення клієнту результату. Внаслідок цього, для того, щоб клієнт міг організувати пов'язану послідовність викликів, серверу буде потрібно знати про стан клієнта. Це реалізується за допомогою механізму сесії користувача. В рамках сесії сервер може зберігати деяку порцію даних про клієнта і зіставляти виклики клієнта з цими даними.

Після дослідження списку кімнат і прийняття рішення про приєднання до однієї з них, клієнт викликає відповідний метод, передаючи сервера ідентифікаційні дані користувача і номер кімнати. Сервер реєструє користувача в кімнаті і починається стадія обміну повідомленнями. У момент приєднання до кімнати також відбувається ініціалізація сесії, пов'язаної з користувачем.

На стадії обміну клієнт відправляє на сервер повідомлення, а сервер заносить їх в чергу відповідної кімнати. Ідентифікацію користувача і приналежність до кімнати сервер визначає за даними сесії. Потім сервер переглядає чергу повідомлень, і, відповідно до деякого критерію, виділяє нові повідомлення інших користувачів, що підлягають пересилання клієнту.

Відправлення та одержання повідомлень клієнт здійснює за одне звернення до сервера, що дозволяє скоротити загальну кількість викликів. При

цьому для виконання цієї процедури клієнт не пов'язаний необхідністю мати вихідні повідомлення для відправки, а сервер не зобов'язаний повернути клієнту нові повідомлення - їх може і не бути в черзі. Це означає, що як вихідний, так і входить набір команд, щодо клієнта, може бути порожнім. Клієнт викликає метод обміну повідомленнями в міру надходження вихідних команд від користувача або з певною періодичністю при їх відсутності.

Так як сервер не має можливості дізнатися про здатність користувача продовжувати комунікацію, через брак постійного підключення, на сервері задається тайм-аут сесії, після закінчення якого сесія переривається. Тайм-аут відміряється з моменту попереднього запиту. Обмін повідомленнями проводиться синхронно в окремому потоці клієнтського додатка і, крім власне функції відправки-отримання повідомлень, забезпечує підтримку сесії користувача на сервері.

На завершення сеансу комунікації користувач виходить з кімнати, сервер видаляє його зі списку користувачів кімнати і перериває сесію.

Зауважимо, що користувач може приєднатися не тільки до вже існуючої кімнаті. Він має можливість одночасно створити нову кімнату і увійти в неї. При цьому сервер додає створену кімнату в список вже наявних. Спочатку користувач, який створив кімнату, є в ній єдиним. Надалі в цю кімнату можуть входити і виходити інші користувачі. У той момент, коли з кімнати виходить останній користувач, вона знищується сервером і видаляється зі списку кімнат.

Клієнт і сервер можуть обмінюватися різними типами повідомлень. До їх складу входять:

• повідомлення про підключення і відключення користувача,

• повідомлення про зміну статусу користувача,

• текстові повідомлення, що містять текстовий введення користувача - повідомлення чату,

• повідомлення, що включають дані про графічному введенні - графічні команди.

Зауважимо, що повідомлення про підключення-відключення користувача і зміні його статусу використовуються виключно для інформування членів кімнати. Події ж, викликають розсилку таких повідомлень, є результатом виклику методів сервера клієнтом або внутрішньою логікою сервера. Події ж, викликають розсилку таких повідомлень, є результатом виклику методів сервера клієнтом або внутрішньою логікою сервера.

3.3.3 Доступ до методів web-служби за допомогою SOAP

Як згадувалося в п.2.2, клієнт взаємодіє з серверної частиною за допомогою доступу до методів web-служби, також званим web-методами, по протоколу SOAP. Таким чином, усувається проблема блокування фаєрволом нестандартних користувальницьких портів. В результаті для більшості користувачів забезпечується безперешкодне підключення до сервера.

У коді клієнта створюється проксі-клас web-служби, на основі якого клієнт може створювати об'єкти і викликати методи цих об'єктів. Він генерується відповідно до WSDL описом web-служби (див. Рис. 6) за допомогою утиліти надається середовищем розробки. Проксі-клас містить такий же набір методів, що і у оригінального класу (тут маються на увазі методи, помічені в реалізації web-служби атрибутом WebMethod, тобто можливі для віддаленого виклику). Для клієнта, таким чином, виклик web-методу нічим не відрізняється від виклику звичайного методу. Принциповою відмінністю тут є механізм передачі параметрів і значення, що повертається.

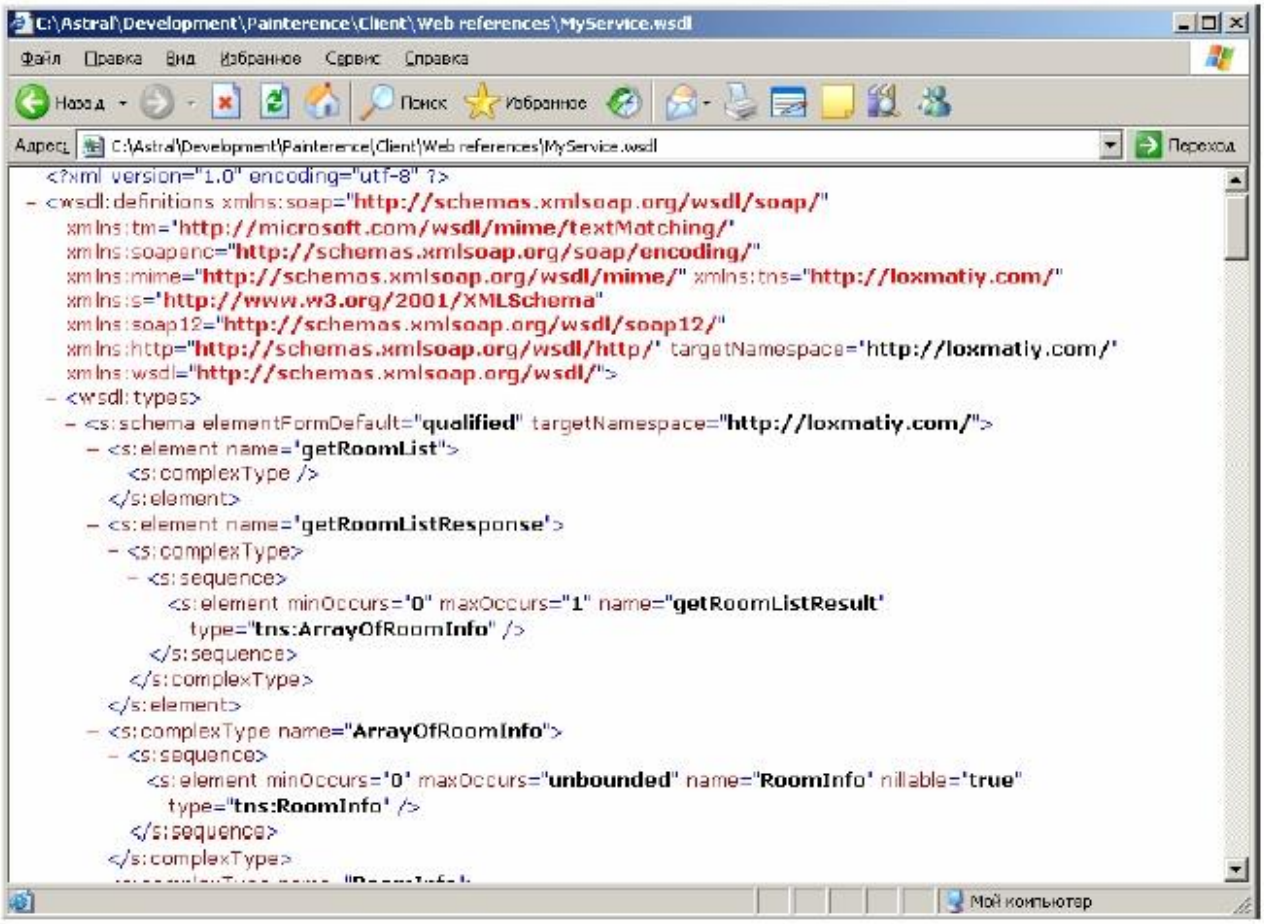

Рисунок 3.4 - Приклад фрагмента опису web-служби на WSDL

Технологія web-служб дозволяє передавати різні типи параметрів. Використовуючи протоколи HTTP-GET і HTTP-POST можливо передавати прості типи CLR, масиви з цих типів і рядки. Однак використання протоколу

SOAP видається більш гнучкою альтернативою [7]. На додаток до вже перерахованих типів даних він дозволяє передавати масиви складних класів і структур, будь-які вузли XML і призначені для користувача типи. Передача даних здійснюється у форматі XML. Об'єкт, який потрібно проходить процедуру сериализации в XML і вставляється в тіло SOAP-повідомлення. На приймаючій стороні відбувається зворотний процес - з повідомлення SOAP витягується XML вузол, що містить параметри об'єкта і піддається десеріалізації. В результаті на приймаючій стороні ми маємо копію об'єкта в точно такому ж стані, що і на передавальної.

Дані SOAP можна передавати використовуючи різні інтернет-протоколи (HTTP, SMTP та ін.) У нашому випадку, SOAP-повідомлення міститься в заголовку HTTP.

3.2.3. Сесія користувача

Сесія - це цикл роботи клієнта з сервером, протягом якого клієнт може виробляти запити, результат яких логічно взаємопов'язаний з попередніми запитами. Для забезпечення взаємозв'язку таких запитів застосовується управління станом, яке може реалізовуватися як на клієнтської, так і на стороні сервера.

Для того щоб сервер міг ідентифікувати конкретного клієнта або додаток і дізнатися про його стан використовуються cookies. Коли клієнт робить первісний запит до сервера, сервер ініціалізує сесію і передає клієнту невеликий шматок даних - cookie. Ці дані згодом зберігаються на стороні клієнта у вигляді файлу або в оперативній пам'яті. Клієнт, який здійснює запит, додає cookie до запиту, а сервер може прочитати cookie, виділити необхідну інформацію і співвіднести запит з відповідними даними про стан користувача.

Іншим варіантом надання сервера інформації про стан клієнта є зберігання параметрів стану на стороні клієнта і передача цих параметрів сервера з кожним запитом. Але при цьому, якщо дані стану мають складну структуру і великий обсяг, істотно зростає обсяг переданої сервера інформації.

Наприклад, для виконання обміну повідомленнями, на сервері необхідно знати ім'я користувача, кімнату, до якої він належить і час останньої синхронізації повідомлень. Ми можемо передати цю інформацію в якості параметрів web-методу, але тоді збільшується ймовірність некоректної обробки запиту. Так як клієнт здатний поставити будь-які значення переданих параметрів, свідомо чи внаслідок помилки, ці значення можуть неправильно відображати його стан. Наприклад, передача неправильного номера кімнати веде до відповідної невірної синхронізації повідомлень. На додаток до цього,

може існувати така інформація про стан користувача, яку не можна дозволити редагувати клієнту. До неї відносяться, наприклад, права користувача. У такому випадку дані стану клієнта повинні зберігатися на стороні сервера, та змінюватись у відповідності з його внутрішньою логікою.

3.3. Структура серверної частини

3.3.1. Web-служба в складі ASP.Net додатку

Серверну частину нашого застосування реалізуємо у вигляді XML web-служби, що розгортається в складі web-додатку на web-сервері Internet Information Services. Основними функціями серверної частини є: управління кімнатами, підключеннями клієнтів і їх станом, виконання синхронізації між клієнтами. Web-служба також містить набір методів для надання клієнтам додаткової інформації.

Web-служба є класом, що містить набір методів, деякі з яких доступні для віддаленого виклику. Цей клас може бути включений до складу web-додатку у вигляді файлу з вихідним кодом або збірки .Net. Web-додаток, по суті, є пов'язаною набором файлів, що містять інтерфейс, що надається клієнтові у вигляді web-сторінок або методів web-служб, логіку обробки запитів клієнтів, що виникають при перегляді цих web-сторінок або виклики методів, а також налаштування конфігурації програми і опис реакції на внутрішні події web-додатку. Ці файли розміщені в межах віртуального каталогу IIS.

ASP. Net - технологія Microsoft, призначена для розробки web-додатків. ASP. Net розширює базову функціональність IIS, дозволяючи web-серверу обробляти запити елементів ASP.Net web-додатку. Клієнти звертаються до ресурсів web-додатку за допомогою URL. Web-сервер транслює URL в фізичний шлях на диску сервера і ім'я файлу. ASP. Net є ISAPI розширенням IIS.

Для того щоб зрозуміти, як відбувається виклик web-сервісу і управління станом в рамках web-додатку, розглянемо життєвий цикл додатка ASP.Net.

3.3.2. Життєвий цикл ASP.Net додаток

Життєвий цикл ASP.Net додатку починається, коли клієнт відправляє web-серверу запит. IIS аналізує розширення запитуваного файлу і, відповідно до нього, визначає, яке ISAPI розширення повинно обробляти цей запит. Потім запит передається цьому розширенню для обробки. Наприклад, ASP.Net обробляє такі типи файлів, як aspx, .ascx, .ashx, і .asmx. Крім цих стандартних розширень, до ASP.Net можуть бути прив'язані інші розширення файлів. Для цього розширення потрібно зареєструвати в IIS і поставити йому у відповідність обробник ASP.Net.

Коли ASP.Net отримує перший запит ресурсу web-додатку, в процесі ASP.Net клас ApplicationManager створює домен додатку. Домени додатки забезпечують ізоляцію додатків на рівні глобальних змінних. В межах домену створюється екземпляр класу HostingEnvironment, який надає інформацію про програму, таку як каталог, в якому воно розташовується.

Після ініціалізації HostingEnvironment, ASP.Net створює об'єкти класу HTTPContext, HTTPResponse і HTTPRequest. Ці об'єкти створюються для кожного запиту. HTTPRequest містить інформацію про запит, включаючи cookies, HTTPResponse - відповідь, що посилається клієнтові, а HTTPContext включає в себе ці об'єкти.

Після цього під час запуску програми, шляхом створення об'єкта HTTPApplication. Web-додаток може містити файл Global.asax, в якому описаний клас-спадкоємець HTTPApplication з реалізацією специфічної для web-додатків реакції на внутрішні події. В цьому випадку для створення екземпляра додатка ASP.Net використовує не оригінальний клас HTTPApplication, а похідний від нього.

Спочатку для кожного запиту створюється новий екземпляр HTTPApplication, але згодом ASP.Net може повторно використовувати ці екземпляри для підвищення продуктивності. Ставлення запитів клієнтів і примірників додатків відображено на рисунку 3.5.

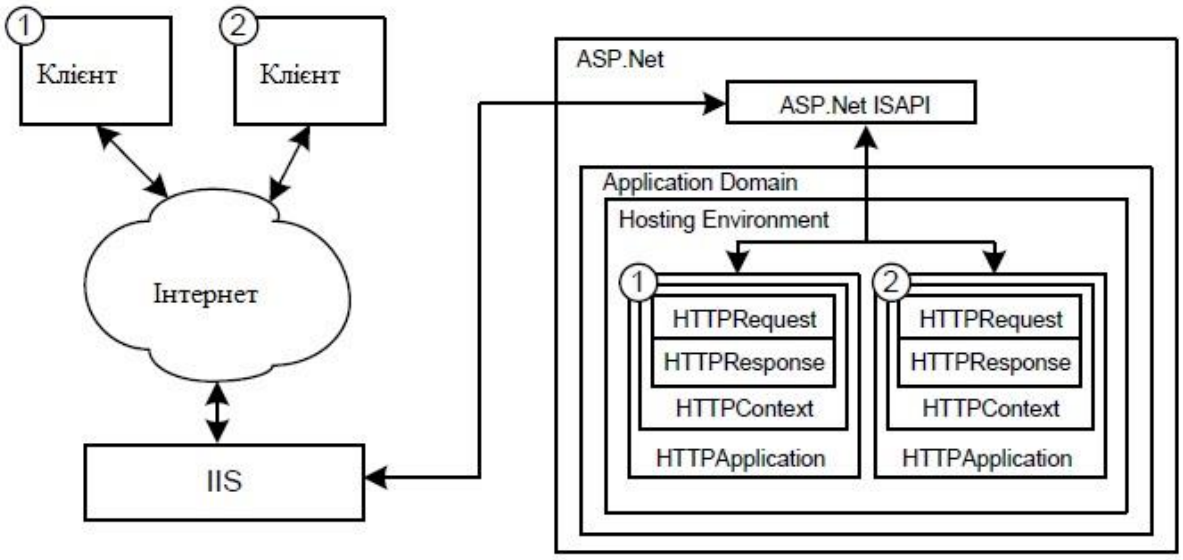

Рисунок 3.5. - Схема відносини примірника HTTPApplication і запиту клієнта

На наступній стадії додаток обробляє запит і породжує події. Реалізуючи обробники цих подій, можна розширити клас HTTPApplication.

Примірник HTTPApplication одночасно обробляє тільки один запит. Це дозволяє зберігати пов'язані з запитом параметри в нестатичних членах класу і спрощує обробку подій додатів, так як при доступі до цих членам не потрібно їх блокувати.

Процес обробки запиту протягом життєвого циклу можна розширити за допомогою реалізації HTTP модулів. ASP.Net надає кілька реалізацій модулів HTTP, в їх числі SessionStateModule. Цей модуль викликає події, пов'язані з сесії користувача, такі як Session_Start і Session_End. Додаток може підписатися на ці події і робити відповідну обробку, деяким чином реагуючи на них.

Для обробки доступу до методу web-служби, додаток створює екземпляр класу web-служби і викликає відповідний метод. Клас web-служби може бути похідним від базового класу System.Web.Services.WebService, що дозволяє

отримати прямий доступ до об'єктів Application і Session (типів HttpApplicationState і HttpSessionState). Ці об'єкти є колекціями пар «ключ-значення» і надають спосіб збереження інформації про стан сесії користувача і тан додатків між викликами. Стан додатків розділяється між всіма сеансами

Дані сесії зберігаються в пам'яті процесу ASP.Net, на сервері станів (State server) або в базі даних. Зберігання даних в пам'яті процесу забезпечує високу швидкість доступу за умови, що обсяг даних не настільки великий, щоб витратити оперативну пам'ять сервера і задіяти віртуальну дискову пам'ять.

3.3.3. Використання стану сесії при синхронізації

Серверна частина повинна забезпечувати управління кімнатами, обробку запитів користувача на підключення до кімнат і синхронізацію повідомлень користувачів в межах кімнати. Для цього вона повинна мати список кімнат і список користувачів. Кожній кімнаті належить окрема черга повідомлень.

Примірник web-служби містить об'єкт Session, який дозволяє отримає доступ до об'єкта HttpSessionState, асоційованого з поточним запитом. У ньому міститься інформація про користувача, про кімнату, до якої він належить і про час останньої синхронізації. В результаті ми отримуємо доступ до необхідної черги повідомлень.

Синхронізація повідомлень відбувається наступним чином. На сервері зберігається чергу повідомлень, упорядкованих за часом надходження на сервер. Клієнти періодично роблять виклики методу обміну повідомленнями. При прийомі такого виклику від клієнта, сервер отримує зі стану сесії час останньої синхронізації. Воно відповідає останньому повідомленням, отриманим клієнтом з черги. Всі наступні за ним повідомлення сервер відправляє клієнту. Потім він дописує в кінець черги повідомлення, отримані від клієнта при цьому виклику, і позначає їх поточним часом.

3.3.4. Структура клієнтської частини

Клієнтська частина являє собою Windows-додаток на платформі .Net, яке взаємодіє з сервером і надає інтерфейс користувача. Структура клієнтської програми буде містити кілька взаємопов'язаних модулів, кожен з яких буде служити для виконання однієї з функцій:

- взаємодія з користувачем, забезпечення введення текстових і графічних команд,
- виклик методів web-служби відповідно до команд користувача, прийом і передача інформаційних повідомлень,
- перетворення введення користувача в послідовність графічних команд та відображати графічні команд.

Введення графічних команд буде здійснюватися за допомогою миші або графічного планшета. Графічний планшет розширює можливості введення користувача. Крім координат курсору, графічний планшет може надавати інформацію про натискання на перо, поворот пера, обертання пера і т.п. Можливості планшетів варіюються в залежності від конкретної моделі. Так як планшет є специфічним пристроєм введення, в .Net не існує вбудованої підтримки роботи з ним.

Для забезпечення єдиного інтерфейсу взаємодії додатків і різноманітних по функціональності графічних планшетів була розроблена специфікація WinTab . В рамках цієї специфікації описується система, що надає API для взаємодії з пристроєм. Центральним компонентом системи є бібліотека wintab32.dll, яка з одного боку надає користувачеві набір стандартних функцій, а з іншого взаємодіє зі специфічним драйвером пристрою. Ця бібліотека встановлюється в систему разом з драйверами графічного планшета.

В рамках цієї специфікації додаток маніпулює спеціальними об'єктами - контекстами оцифровки (або просто контекстами), які чимось схожі з контекстами пристрою GDI. Контексти використовуються для установки розмірів області введення додатка, масштабування області введення і

управління подіями. Для створення контексту додаток передає бібліотеці дескриптор вікна і параметри контексту, бібліотека створює контекст і пов'язує його з цим вікном. Після цього вікно починає отримувати повідомлення windows, які містять інформацію про події, пов'язані з введенням графічного планшета. Додаток обробляє ці події і викликає функції WinTab для отримання пакетів з даними від графічного планшета. Формат пакета налаштовується в залежності від потреб програми.

Так як середовище виконання платформи .Net є середовищем виконання керованого коду, а використання WinTab має на увазі виклик некерованих функцій, підтримку введення з графічного планшета виділимо в окремий компонент. Він буде здійснювати спілкування з бібліотекою WinTab, а користувачеві надавати інтерфейс на основі подій .Net. Це дозволить взаємодіяти з графічним планшетом таким чином, який схожий на звичайну обробку повідомлень миші в .Net додатках.

3.4. Реалізація програми

3.4.1. Реалізація серверної частини

Серверна частина реалізована у вигляді класу web-служби, класу web-додатки і бібліотеки типів, що містить допоміжні структури даних і класи.

Розглянемо реалізацію web-служби. Клас web-служби є похідним від класу System.Web.Services.WebService, і містить методи, доступні для віддаленого виклику і методи описують внутрішню логіку. Для того, щоб метод був доступний для віддаленого виклику, він позначається атрибутом WebMethod. Якщо метод повинен мати доступ до стану сесії, властивість SessionEnabled цього атрибута встановлюється в true.

Клас містить два статичних елемента даних:

• Hashtable rooms - таблиця кімнат, ключем є ідентифікатор кімнати, значенням - клас Room,

• Hashtable users - таблиця користувачів, ключем є ім'я користувача, значенням - клас User.

Клас Room описує кімнату на сервері, опис містить унікальний номер кімнати, ім'я кімнати, список команд, розмір області малювання і список знаходяться в кімнаті користувачів.

Список команд реалізований у вигляді класу CommandList, породженого від базової колекції SortedList <long, Command>, і містить список пар <ключ, команда>. Реалізація похідного класу дозволяє вибрати або додати до списку масив команд

• Command [] getNextItems (ref long key) - вибирає зі списку і повертає команди, що знаходяться після зазначеного ключа, змінює значення переданого параметра key на ключ останнього елемента в списку,

• void insertItems (long key, Command [] cmds) - додає в кінець списку масив команд, послідовно позначаючи кожну команду ключем key, щоразу збільшуючи його на 1.

Клас User містить ім'я користувача, номер кімнати, в якій він знаходиться і його статус. Статус користувача може приймати одне із значень перерахування UserStatus. Наведемо їх у порядку зростання прав:

• SPERCTATOR - спостерігач, може тільки спостерігати за малюванням інших учасників і спілкуватися в чаті,

• DRAWER - художник, на додаток має право малювати,

• OWNER - власник кімнати, також може змінювати статус інших користувачів.

Клас web-служби містить наступні web-методи:

• RoomInfo [] getRoomList () - повертає список описів кімнат. Опис містить ім'я кімнати, список користувачів в кімнаті і оцінку активності в кімнаті.

• int Connect (string userName, int roomId, string roomName, int canvasWidth, int canvasHeight) - приєднання користувача до кімнати roomId або створення нової кімнати, якщо такої кімнати не існує. У разі, якщо користувач вже здійснював вхід, йому повертається помилка за допомогою SoapException. Функція повертає номер кімнати, в яку доданий користувач.

• string Disconnect () - вихід користувача з кімнати. Користувач видаляється з кімнати і списку користувачів. Сесія переривається.

• Command [] ExchangeCommands (Command [] inputCommands) - обмін командами. Вхідний масив фільтрується відповідно до статусу користувача, додається до списку команд кімнати користувача і позначається поточним часом, перетвореним в long. Повертається масив нових команд. В дані сесії заноситься ключ останньої команди в списку.

• void GrantUser (string userName, int state) - призначення статусу користувача з ім'ям userName. Метод успішно виконується, якщо викликає користувач є власником кімнати і цільової користувач також знаходиться в ній.

Клас додатки містить обробник події Session_End, який викликається при закінченні сесії користувача. Якщо сесія була перервана після закінчення тайм-ауту, викликається метод web-служби disconnectBySessionTimeOut (), який виробляє примусове відключення користувача.

3.4.2. Реалізація клієнтської частини

З точки зору взаємодії з сервером клієнт повинен виконувати наступні функції:

• підключення до віддаленого сервера,

• забезпечення підключення через проксі-сервер,

• отримання списку доступних кімнат,

• перетворення введення користувача в команди і відправка на сервер,

• отримання команд з сервера.

Інтерфейс користувача повинен дозволяти:

• налаштувати параметри підключення,

• зберегти настройки користувача і відновлювати їх при наступних запусках програми,

• переглянути список кімнат на сервері,

• приєднатися до існуючої кімнаті або створити нову,

• спілкуватися з іншими користувачами за допомогою текстової та графічної інформації,

• скористатися набором різних інструментів для введення графічної інформації,

• переглянути список користувачів,

• обмежувати можливості користувача відповідно до його правами,

• змінювати права користувачів,

• скористатися для введення графічним планшетом.

Реалізація клієнтської програми розділена на кілька частин: модуль взаємодії з сервером, модуль обробки графічних команд, модуль інтерфейсу користувача і модуль ядра, призначений для зв'язку між ними.

3.4.3. Модуль взаємодії з сервером

Модуль призначений для взаємодії з сервером за допомогою виклику методів web-служби. Для виклику методів web-служби за допомогою середовища розробки створюється проксі-збірка, яка представляє собою вихідний код класу, похідного від базового класу SoapHttpClientProtocol. У цьому базовому класі визначено ті можливості, за допомогою якої наша проксі-збірка буде взаємодіяти з web-службою.

Розглянемо деякі методи і члени базового класу:

• BeginInvoke () - виробляє асинхронний виклик методу web-служби,

• EndInvoke () - завершує асинхронний виклик методу web-служби,

• Invoke () - виробляє синхронний виклик методу web-служби,

• Proxy - дозволяє отримати або встановити інформацію про налаштування проксі-сервера при підключенні через firewall,

• TimeOut - дозволяє отримати або встановити тайм-аут очікування повернення при синхронному виклику,

• Url - дозволяє отримати або встановити URL-адресу web-служби.

Згенерована проксі-збірка містить методи для синхронного і асинхронного виклику кожного методу web-служби. При синхронному виклик виконання потоку зупиняється до повернення службою результату або закінчення тайм-ауту. При асинхронному виклику управління відразу ж повертається клієнтові, а після виконання викликається методу відбувається зворотний виклик проксі-збірки.

Проксі-збірка також містить визначення типів web-служби і методи, призначені для виклику web-служби. Ці методи мають таку ж сигнатуру, як і викликаються методи web-служби, що дозволяє клієнту зручним і звичним способом виробляти їх виклик.

Звернемо увагу на властивість Proxy базового класу проксі-збірки. Це властивість містить інформацію, що описує проксі-сервер при підключенні через firewall. За замовчуванням ця властивість містить настройки проксі-сервера з операційної системи. Так що в деяких випадках користувачеві, заимодействующему з web-службою через проксі-сервер, навіть не доведеться самостійно налаштовувати його параметри.

При ініціалізації модуля йому передаються посилання на вхідний і вихідний буфери команд, модуль створює екземпляр проксі-збірки і встановлює установки для з'єднання з налаштувань користувача. Підключення

до кімнати відбувається за допомогою виклику методу Connect. При успішному підключенні породжується подія ConnectionEstablished і в окремому потоці стартує цикл обміну повідомленнями. При провал спроби з'єднання викликається подія ConnectionFailed.

Якщо буфер вихідних повідомлень порожній, потік призупиняє виконання на деякий час. При отриманні повідомлень викликається подія CommandReceived. Прийняті повідомлення записуються у вхідний буфер.

Після успішного відключення виникає подія ConnectionFinished.

3.4.4. Модуль обробки графічних команд

Модуль обробки графічних команд призначений для формування та для демонстрації зображень команд. Графічна команда складається з параметрів інструменту малювання, набори крапок на площині малювання і набору пов'язаної з точками динамічної інформації. До параметрів інструменту відноситься тип інструменту, колір, товщина лінії, мінімальний і максимальний множник динаміки, наявність динаміки. До кожній точці може бути прив'язаний коефіцієнт динаміки, який відображає зміну товщини лінії в даній точці по відношенню до базової товщині лінії інструменту.

Інструменти реалізовані у вигляді класів, породжених від абстрактного класу ToolBuffer. Розглянемо його найбільш важливі члени і методи.

• Bitmap buffer - растровий буфер, в який відбувається отрисовка,

• List <Point> points - список точок,

• addPoint (Point pt) - додавання точки у відповідність з логікою інструменту,

• drawBuffer () - отрисовка масиву точок в буфер,

• getBuffer () - отримання буфера інструменту.

Реалізації абстрактного класу містять властивості і реалізації методів відображають специфіку конкретного інструменту. Наприклад, клас кисті

додатково містить масив динамічної інформації, що дозволяє малювати лінії змінної товщини; методу додавання точок для прямокутника реалізований таким чином, щоб список точок завжди містив тільки першу і останню додану точку, і т.п.

Модуль обробки графічних команд містить растровий буфер фону, і два інструменти - первинний, призначений для відображення поточної вводиться користувачем команди і вторинний, службовець для відтворення команд з сервера. Буфер фону містить повністю промальовані команди.

Формування графічної команди відбувається в наступних методах:

• startCommand (ToolParameters tool, int buf) - початок введення команди, ініціалізація інструменту на отснове параметрів tool, другий параметр - PRIMARY або SECONDARY інструмент,

• addPoints (Point [] pts, int [] pressures, int buf) - додавання кількох точок і динамічної інформації до буферу інструменту,

• DrawCommand endCommand (int buf) -формування команди на основі інструменту і збереження буфера інструменту в фоновий буфер.

Модуль також містить методи для відтворення фонового буфера і буферів інструментів на переданий з модуля інтерфейсу елемент управління. При цьому проводиться необхідне масштабування і зміщення зображення.

3.4.5. Модуль інтерфейсу користувача

Інтерфейс користувача являє собою реалізацію вікна програми. Містить обробники взаємодії з органами управління, обробники подій миші і графічного планшета, а також обробники подій про наявність команд різних типів.

3.4.6. Модуль ядра

Модуль ядра призначений для зв'язку інших модулів, зберігання загальної інформації, зчитування і збереження налаштувань користувача.

Модуль містить буфер вхідних і вихідних команд, які заповнює мережевий модуль, надає модулю інтерфейсу інформації про стан з'єднання, містить події про прихід команд, а також виробляє створення і ініціалізацію графічного і мережевого модулів.

3.4.7. Компонент відповідає за роботу з графічним планшетом

Компонент призначений для організації обробки введення з графічного планшета. Він дозволяє використовувати звичний спосіб обробки введення, заснований на реалізації обробників подій. У реалізації класу компонента Інкапсульована вся робота з бібліотекою WinTab. Компонент надає набір функцій для відкриття і закриття контексту, зміни параметрів контексту, отримання інформації про пристрій і обробки віконних повідомлень. Клас містить набір закритих функцій, імпортованих з бібліотеки WinTab, а також список констант WinTab і опис структур даних Axis, Context і Packet.

Структура Axis містить дані діапазону і дозволу осі координат графічного планшета. Цією структурою показані не тільки осі X і Y площині введення, а й інші параметри, наприклад, «вісь» натискання на перо.

Структура Packet описує пакет даних і містить інформацію про натискання кнопок на пере планшета, координатах пера і ступеня натискання.

3.5. Проектування програмного продукту

3.5.1 Логічне проектування програми

На етапі логічного проектування описується організація елементів, що становлять програмне рішення. Модель, отримана на стадії логічного проектування повинна забезпечувати:

• незалежність від засобів розробки;

• простота моделі;

• відображення структури програмного продукту.

У логічної моделі створюються алгоритми чи інші логічні елементи, за допомогою яких здійснюється рішення прикладної задачі.

ВИСНОВКИ

За результатами даної роботи можна зробити наступні висновки:

Було проведено аналіз існуючих програм для мережевої комунікації за допомогою графічної інформації та текстових повідомлень. Були проаналізовані їхні переваги і недоліки.

Було обрано метод ствоерння додатку

Було обрано середовище для розробки додатку на основі мови програмування C#

Був реалізований додаток який може бути як інструмент для створення і спостереження за процесом створення графічних зображень для дистанційної взаємодії.

Додаток розроблювався з урахуванням можливості розширення графічного потенціалу та впровадження в складі web-сайту для художників.

СПИСОК ЛІТЕРАТУРИ

1. Web Conferencing - https://www.3cx.com/phone-system/web-conferencing/

2. Official website of Gracebyte Software - https://www.gracebyte.com/

3. Network Assistant - http://www.gracebyte.com/nassi/

4. Imagination Cubed - https://www.imaginationcubed.com/

5. Official web site of Microsoft Corporation - https://www.microsoft.com/

6. Троелсен, Эндрю мова програмування C# 5.0 та платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 633 с.

8. Understanding SOAP - https://www.netiq.com/documentation/operations-center-57/web2connect/data/bli2d6u.html

9. Wacom PC Software Development Support - http://www.wacomeng.com/devsupport/downloads/pc/wt1pt1.zip

10. Страуступ, Б. Мова програмування C++. Специальне видання / Б. Страуступ. — М.: Бином, 2015. — 1136 с.

11. Мейерс, Скотт «Эффективный и современный C++». 42 рекомендації по використанню C++11 и C++14 / Скотт Мейерс. - М.: Вильямс, 2015. - 304 с.

12. Omnis Studio - https://www.omnis.net/

ДОДАТОК

Додаток 1

Сервер

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.Net;
using System.Net.Sockets;
using System.IO;
namespace ChatServer
{
public partial class Form1 : Form
{
private delegate void UpdateStatusCallback(string strMessage);
public Form1()
{
InitializeComponent();
}
private void btnListen_Click(object sender, EventArgs e)
{
// Parse the server's IP address out of the TextBox
IPAddress ipAddr = IPAddress.Parse(txtIp.Text);
// Create a new instance of the ChatServer object
ChatServer mainServer = new ChatServer(ipAddr);
```

```
// Hook the StatusChanged event handler to mainServer_StatusChanged
ChatServer.StatusChanged += new StatusChangedEventHandler (mainServer_StatusChanged);
// Start listening for connections
mainServer.StartListening();
// Show that we started to listen for connections
txtLog.AppendText("Monitoring for connections...\r\n");
}
public void mainServer_StatusChanged(object sender, StatusChanged EventArgs e)
{
// Call the method that updates the form
this.Invoke(new UpdateStatusCallback(this.UpdateStatus), new object[] { e.EventMessage });
}
private void UpdateStatus(string strMessage)
{
// Updates the log with the message
txtLog.AppendText(strMessage + "\r\n");
}
}
}
```

Клиент

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
```

```
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;
namespace ChatClient
{
public partial class Form1 : Form
{
// Will hold the user name
private string UserName = "Unknown";
private StreamWriter swSender;
private StreamReader srReceiver;
private TcpClient tcpServer;
// Needed to update the form with messages from another thread
private delegate void UpdateLogCallback(string strMessage);
// Needed to set the form to a "disconnected" state from another thread
private delegate void CloseConnectionCallback(string strReason);
private Thread thrMessaging;
private IPAddress ipAddr;
private bool Connected;
public Form1()
{
// On application exit, don't forget to disconnect first
Application.ApplicationExit += new EventHandler(OnApplicationExit);
InitializeComponent();
}
// The event handler for application exit
public void OnApplicationExit(object sender, EventArgs e)
```

```
{
if (Connected == true)
{
// Closes the connections, streams, etc.
Connected = false;
swSender.Close();
srReceiver.Close();
tcpServer.Close();
}
}
private void btnConnect_Click(object sender, EventArgs e)
{
// If we are not currently connected but awaiting to connect
if (Connected == false)
{
// Initialize the connection
InitializeConnection();
}
else // We are connected, thus disconnect
{
CloseConnection("Disconnected at user's request.");
}
}
private void InitializeConnection()
{
// Parse the IP address from the TextBox into an IPAddress object
ipAddr = IPAddress.Parse(txtIp.Text);
// Start a new TCP connections to the chat server
tcpServer = new TcpClient();
```

```
tcpServer.Connect(ipAddr, 1986);
// Helps us track whether we're connected or not
Connected = true;
// Prepare the form
UserName = txtUser.Text;
// Disable and enable the appropriate fields
txtIp.Enabled = false;
txtUser.Enabled = false;
txtMessage.Enabled = true;
btnSend.Enabled = true;
btnConnect.Text = "Disconnect";
// Send the desired username to the server
swSender = new StreamWriter(tcpServer.GetStream());
swSender.WriteLine(txtUser.Text);
swSender.Flush();
// Start the thread for receiving messages and further communication
thrMessaging = new Thread(new ThreadStart(ReceiveMessages));
thrMessaging.Start();
}
private void ReceiveMessages()
{
// Receive the response from the server
srReceiver = new StreamReader(tcpServer.GetStream());
// If the first character of the response is 1, connection was successful
string ConResponse = srReceiver.ReadLine();
// If the first character is a 1, connection was successful
if (ConResponse[0] == '1')
{
// Update the form to tell it we are now connected
```

```
this.Invoke(new UpdateLogCallback(this.UpdateLog), new object[] { "Connected Successfully!" });
}
else // If the first character is not a 1 (probably a 0), the connection was unsuccessful
{
string Reason = "Not Connected: ";
// Extract the reason out of the response message. The reason starts at the 3rd character
Reason += ConResponse.Substring(2, ConResponse.Length - 2);
// Update the form with the reason why we couldn't connect
this.Invoke(new CloseConnectionCallback(this.CloseConnection), new object[] { Reason });
// Exit the method
return;
}
// While we are successfully connected, read incoming lines from the server
while (Connected)
{
// Show the messages in the log TextBox
this.Invoke(new UpdateLogCallback(this.UpdateLog), new object[] { srReceiver.ReadLine() });
}
}
// This method is called from a different thread in order to update the log TextBox
private void UpdateLog(string strMessage)
{
// Append text also scrolls the TextBox to the bottom each time
```

```
txtLog.AppendText(strMessage + "\r\n");
}
// Closes a current connection
private void CloseConnection(string Reason)
{
// Show the reason why the connection is ending
txtLog.AppendText(Reason + "\r\n");
// Enable and disable the appropriate controls on the form
txtIp.Enabled = true;
txtUser.Enabled = true;
txtMessage.Enabled = false;
btnSend.Enabled = false;
btnConnect.Text = "Connect";
// Close the objects
Connected = false;
swSender.Close();
srReceiver.Close();
tcpServer.Close();
}
// Sends the message typed in to the server
private void SendMessage()
{
if (txtMessage.Lines.Length >= 1)
{
swSender.WriteLine(txtMessage.Text);
swSender.Flush();
txtMessage.Lines = null;
}
txtMessage.Text = "";
```

```
}
// We want to send the message when the Send button is clicked
private void btnSend_Click(object sender, EventArgs e)
{
SendMessage();
}
// But we also want to send the message once Enter is pressed
private void txtMessage_KeyPress(object sender, KeyPressEventArgs e)
{
// If the key is Enter
if (e.KeyChar == (char)13)
```

серверна частина

Реалізація класу веб-служби знаходиться в файлі ProjectChat.asmx.cs.

клас ProjectChatService:

Hashtable users; // таблиця користувачів

Hashtable rooms; // таблиця кімнат

RoomInfo [] getRoomList (); // отримання списку описів кімнат

int Connect (string userName, int roomId, string roomName, int canvasWidth, int canvasHeight); // підключення до кімнати

void Disconnect (); // вихід з кімнати і відключення

Command [] ExchangeCommands (Command [] inputCommands); // обмін повідомленнями

void GrantUser (string userName, int state); // зміна статусу користувача

User getSessionUser (); // отримання об'єкта користувача зі стану сесії

Room getSessionRoom (); // отримання об'єкта кімнати

Room getRoomByRoomId (int roomId); // отримання об'єкта кімнати

Room getRoomByUsername (string username); // отримання об'єкта кімнати

void killRoom (int roomId) // видалення кімнати

void disconnectUser (string userName) // відключення користувача

void disconnectBySessionTimeout (string userName) // відключення користувача після закінчення часу очікування сесії

клас Command:

byte cmdType; // тип команди

string message; // повідомлення

string points; // координати точок

string pressures; // значення динаміки

int timespan; // інтервал введення команди

Tool toolParameters; // параметри інструменту

клас CommandList: SortedList <long, Command>

object SyncRoot; - об'єкт синхронізації доступу

Command [] GetNextItems (ref long key); // отримання списку нових повідомлень

void InsertItems (long key, Command [] cmdList); // вставка масиву повідомлень

клієнтська частина

Модуль ядра реалізований в файлі CoreModule.cs

клас CoreModule:

public DrawModule drawer; // модуль обробки графічних команд

public NetModule net; // модуль взаємодії з сервером

public Options options; //налаштування користувача

public CommandList sendBuffer; // буфер вихідний повідомлень

public CommandList receiveBuffer; // буфер вхідних повідомлень

public bool connected; // прапор з'єднання

private string myUserName; // поточне ім'я користувача

private UserStatus myStatus; // поточний статус користувача

private string myRoomName; // поточна кімната

public event CommandProcessHandler Draw; // подія відтворення команди

public event CommandProcessHandler Chat; // подія чату

public event CommandProcessHandler UserConnected; // подія приєднання користувача

public event CommandProcessHandler UserDisconnected; // подія відключення користувача

public event CommandProcessHandler UserStatusChanged; // подія зміни статусу користувача

private void OnDraw (Command cmd); // генерація події Draw

private void OnChat (Command cmd); // генерація події Chat

private void OnUserConnected (Command cmd); // генерація події UserConnected

private void OnUserDisconnected (Command cmd); // генерація події UserDisconnected

private void OnUserStatusChanged (Command cmd); // генерація події UserStatusChanged

public void initDrawModule (Size canvasSize); // ініціалізація модуля обробки графічних команд

public void initNetModule (); // ініціалізація модуля взаємодії з сервером

void net_ConnectionEstablished (object sender, string url); // обробка події установки з'єднання

void net_ConnectionLost (object sender, string url); // обробка події втрати з'єднання

void net_CommandReceived (); // обробка події отримання команди

public void PushChatCmd (string message); // додавання команди чату в буфер вихідної пошти

public void PushDrawCmd (int timespan); // додавання графічної команди в буфер вихідної пошти

private void processCommand (Command cmd); // обаботка отриманої команди

Модуль взаємодії з сервером реалізований в файлі NetModule.cs

Клас NetModule:

private ProjectChatService svc; // проксі-збірка веб-служби

private Thread exchangeThread; // потік обміну повідомленнями

private const int TIMEOUT = 2000; // час очікування введення команди

public ConnectSync status; // об'єкт для синхронізації статусу підключення

public event CommandReceivedHandler CommandReceived; // подія отримання команди

public event ConnectEventHandler ConnectionEstablished; // подія встановлення з'єднання

public event ConnectEventHandler ConnectionLost; // подія втрати з'єднання

public event ConnectionErrorEventHandler ConnectionFailed; // подія помилки з'єднання

public void initialize (string serviceUrl, bool useProxy, bool authReq, string proxyUrl, string login, string password, string authType); // ініціалізація модуля

public void Connect (string username, int roomId, string roomName, Size canvasSize); // приєднання до кімнати

public void Disconnect (); // вихід з кімнати і відключення

public RoomInfo [] getRoomsInfo (); // полеченной споска кімнат

public void grantUser (string username, UserStatus status); // зміна статусу користувача

private void startCommandsExchange (); // точка входу потоку обміну повідомленнями

Модуль обробки графічних команд реалізований в файлі DrawModule.cs

Клас DrawModule:

private ToolBuffer [] toolBuf = new ToolBuffer [2]; // буфери інструментів

private Bitmap canvas; // буфер фону

private Graphics canvasGc; // об'єкт відтворення в буфер

private Command currentCommand; // поточна команда

private PointF origin = new PointF (0,0); // зміщення початку координат

private float aspect = 1.0f; // коефіцієнт масштабування

public void initialize (Size canvasSize); // ініціалізація модуля

public void startCommand (Tool commandTool, int buffer); // початок введення команди

public void addPoint (Point addedPoint, int buffer, int pressure, bool local); // додавання точки до буферу інструменту

public void addPointRange (Point [] pts, int buffer, int [] pressures, bool local); // додавання масиву точок

public Command endCommand (int buffer); // кінець формування команди

private void moveTo (float x, float y); // абсолютне зміщення початку координат

public void moveRel (float dx, float dy); // відносне зміщення початку координат

public void zoomTo (PointF center, float zoom); // масштабування щодо точки

public Point ConvertToLocal (Point globalpt); // перетворення точки в локальну систему

public Point ConvertToGlobal (Point localpt); // перетворення точки в глобальну систему

public RectangleF RectToLocal (Rectangle globalRect); // перетворення прямокутника в локальну систему

public Rectangle RectToGlobal (RectangleF localRect); // перетворення прямокутника в глобальну систему

public void drawToGraphics (Graphics gc, bool highQuality, Rectangle destRect, Size totalSize); // отрисовка на елемент управління

Клас ToolBuffer:

protected Bitmap buffer; // буфер інструменту

protected Rectangle clipRect = new Rectangle (0,0,0,0); // обмежує прямокутник

protected List <Point> points = new List <Point> (); // список точок

protected bool invalid = false; // ознака "брудного" буфера

protected virtual void initialize (Bitmap buffer); // ініціалізація буфера інструменту

protected virtual void calcClipRect (); // обчислення прямокутника

protected virtual void drawBuffer (); // отрисовка буфера

public virtual void addPoint (Point point); // додавання точки

public virtual void addPointRange (Point [] pts); // додавання масиву точок

public Bitmap getBuffer (); // отримання буфера інструменту

Компонент роботи з графічним планшетом реалізований в файлі Tablet.cs.

Клас Tablet:

// {Читання різноманітної інформації про планшет}

[DllImport ("Wintab32.dll")]

public extern static uint WTInfo (uint wCategory, uint nIndex, [MarshalAs (UnmanagedType.AsAny), Out] Object lpOutput);

// {Відкриття контексту і початок доступу до планшета}

[DllImport ("Wintab32.dll")]

public extern static int WTOpen (IntPtr hWnd, LogContext lpLogContext, BOOL fEnable);

// {Закриття контексту і кінець доступу до планшета}

[DllImport ("Wintab32.dll")]

public extern static BOOL WTClose (HCtx hCtx);

// {Отримання параметрів контексту}

[DllImport ("Wintab32.dll")]

public extern static BOOL WTGet (HCtx hCtx, LogContext lpLogContext);

// {Установка параметрів контексту}

[DllImport ("Wintab32.dll")]

public extern static BOOL WTSet (HCtx hCtx, LogContext lpLogContext);

// {Отримання пакета даних}

```
[DllImport ( "Wintab32.dll")]
public extern static BOOL WTPacket (HCtx hCtx, uint wSerial, Packet lpPkts);

// {Перемикання обробки повідомлень планшета}
[DllImport ( "Wintab32.dll")]
public extern static int WTEnable (HCtx hCtx, BOOL enable);
public event PacketEventHandler OnPacket; // подія отримання пакету даних
public event PacketEventHandler OnCursorChanged; // подія зміни курсора
public event ProximityEventHandler OnProximity; // подія входу пера в область чутливості
public event InfoChangedEventHandler OnInfoChanged; // подія зміни параметрів планшета
public event ContextEventHandler OnOpenContext; // подія створення контексту
public event ContextEventHandler OnCloseContext; // подія закриття контексту
public event ContextEventHandler OnOverlapContext; // подія перетину контекстів
public event ContextEventHandler OnUpdateContext; // подія поновлення контексту
private string deviceName; // ім'я пристрою
private byte packetSize; // розмір пакета даних
private bool present; // встановлений в системі планшет
private bool enabled; // включена обробка подій планшета
private HCtx contextHandle; // дескриптор контексту
public LogContext context; // параметри контексту
public void processMessage (Message msg); // обробка віконних повідомлень планшета
public void initialize (); // ініціалізація компонента
```

```
public bool SetContext (); // установка параметрів контексту
public bool GetContext (); // отримання параметрів контексту
public void SetEnabled (bool fEnabled); // перемикання обробки подій
public void SetPacketSize (byte size); // установка розміру пакета
public void Open (IntPtr hWnd); // створення контексту
public void Close (); // закриття контексту
```