НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК 004.514.62

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Програмне забезпечення інформаційно-комунікаційних систем»
зі спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Система керування відеостіною з підтримкою обладнання споживчого сегменту»

Виконав:
студент VI курсу, групи ІТ-з93мп
Степанов Дмитро Віталійович _____

Керівник:
доцент каф. АУТС, к.т.н., доц
Сокульський Олег Євгенович _____

Рецензент:
доцент каф. АСОІУ, к.т.н., доц.,
Попенко Володимир Дмитрович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Степанову Дмитру Віталійовичу

1. Тема дисертації «Система керування відеостіною з підтримкою обладнання споживчого сегменту», науковий керівник дисертації Сокульський Олег Євгенович, доцент, к.т.н., затверджені наказом по університету від «26» 10 2020 р. №3132-с

2. Термін подання студентом дисертації 22.12.2020

3. Об’єкт дослідження відеостіна, що створена з компонентів споживчого сегменту

4. Вихідні дані система керування відеостіной

5. Перелік завдань, які потрібно розробити:

- опис предметної області;

- огляд наявних аналогів;

- дослідження обмежень споживчого обладнання та можливостей їх вирішення;

- проєктування та розробка компонентів системі;

- впровадження системи.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу 8 креслень:

- діаграма послідовності запуску додатку;

- структурна схема клієнтської частини;

- логічна схема системи;

- структурна схема серверної частини;

- діаграма сценаріїв використання;

- ER-діаграма бази даних;

- діаграма класів серверної частини;

- діаграма основних класів клієнтської частини.

7. Орієнтовний перелік публікацій

8. Дата видачі завдання 01.09.2020

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Огляд існуючих рішень	20.09.2020	
2	Формування вимог	05.10.2020	
3	Розробка структурних схем	15.10.2020	
4	Розробка програмної частини	10.11.2020	
5	Оформлення текстової та графічної інформації	29.11.2020	
6	Представлення до захисту	02.12.2020	

Студент Дмитро СТЕПАНОВ

Науковий керівник Олег СОКУЛЬСЬКИЙ

РЕФЕРАТ

Магістерська дисертація складається зі вступу, 5 розділів, висновку, переліку посилань з 20 найменувань і містить 63 рисунки, 26 таблиць. Повний обсяг магістерської дисертації складає 100 сторінок, з яких перелік посилань займає 2 сторінки.

У зв'язку із зростанням багатьох галузей від промисловості до рекламних агентств, сьогодні все більш актуальною стає тема візуалізації великих обсягів даних, які завжди повинні бути перед очима. Використання відеостін можна побачити в диспетчерських центрах, конференц-залах, торгових центрах, на вокзалах і навіть на багатоповерхових будинках.

Відеостіни в диспетчерських центрах дозволяють бачити весь процес виробництва і логістики в одному місці, уникати аварій і оптимізувати процеси, що в кінцевому підсумку дозволяє уникнути збитків та отримати додатковий прибуток. Так само складно переоцінити користь відеостін у рекламних цілях, вони дозволяють підтримувати контент в актуальному стані, а з сучасними технологіями розпізнавання обличчя, показувати таргетовану рекламу.

Всі ці системи об'єднує одне – програмна або апаратна система управління контентом, яких у даний час існує безліч варіантів, але всі вони є пропрієтарними та прив'язаними до певного апаратного рішення. Тому створення універсальної системи управління, яка дозволить будувати системи з будь-яких компонентів, є дуже актуальною.

Метою даної роботи є побудова нової сучасної універсальної системи, яка дозволить створювати відео-стіни не прив’язані до апаратної складової, мати необхідний функціонал керування та управління контентом.

Об’єкт досліджень – відеостіна, що створена з компонентів споживчого сегменту

Предмет досліджень – програмна система керування відеостінами

Для вирішення проблеми в даній роботі використовуються наступні методи:

- використання комп'ютерних компонентів споживчого сегменту;

- комп’ютерне програмування з використовуванням клієнт-серверної розробки на мові .NET C#;

- розробка системи з урахуванням можливого масштабування та розширення функціоналу;

- використання API у серверної частині, що дозволить у майбутньому розробити клієнтську частину для будь якої ОС.

Найбільш суттєвими науковими результатами магістерської дисертації є:

- розгляд сучасних пропозицій digital signage, та аналіз їх функціоналу з точки зору можливості застосування як рішень, які не мають залежності від апаратної частини;

- розробка системи керування відеостінами, яка дозволить будувати їх з будь яких апаратних компонентів, та масштабувати розміри згідно з сучасними потребами.

Практичне значення одержаних результатів полягає у значному здешевленню систем digital signage, можливості їх самостійної побудови та використання у будь якій сфері.

Ключові слова: ВІДЕОСТІНА, ВІДЕОПАНЕЛ, ВІДЕОКАРТА, РАЙЗЕР, ВІДЕОСЕРВЕР, ДОДАТОК КЕРУВАННЯ.

ABSTRACT

Master’s dissertation consists of introduction, 5 sections, conclusion, list of sources of 20 items and contains 63 images, 26 tables. The total dissertation volume is 100 pages, out of which the list of sources takes 2 pages.

Due to the growth of many sectors from industry to advertising agencies, today the theme of visualization of large amounts of data, which should always be before the eyes, is becoming more and more relevant. The use of video walls can be seen in control centers, conference rooms, shopping centers, railway stations and even in multi-storey buildings.

Video walls in control centers allow you to see the entire process of production and logistics in one place, avoid accidents and optimize processes, ultimately, you can avoid losses and gain additional profit. It is also difficult to overestimate the benefits of video walls for advertising purposes, they allow you to keep content up to date, and with modern face recognition technologies, to show targeted advertising.

All these systems are united by one thing – a software or hardware content management system, which currently has many options, but all of them are proprietary and tied to a specific hardware solution. Therefore, the creation of a universal management system, which will allow to build systems with any components, is very relevant.

The purpose of this work is to build a new modern universal system that will allow you to create video walls that are not tied to the hardware component, to have the necessary functionality of management and content management.

The object of research is a video wall, created from components of the consumer segment.

The subject of dissertation is a videowall management software

Research methods

To solve the problem, the following methods are used in this paper:

- computer programming using client-server development in the .NET C # language;

- system development taking possible scale-up and functional extension;

- use of API in the server part, which will allow in the future to develop a client part for any OS.

Scientific novelty

The most significant scientific results of the master's thesis are:

- consideration of modern digital signage proposals, and analysis functionality from the point of view of the possibility of application as solutions that do not depend on hardware;

- development of a universal video wall management system, which will allow to build them with any hardware components and scale their size according to modern needs.

The practical value of the results obtained is in a significant reduction in the cost of digital signage systems, the possibility of their independent construction and use in any sphere.

Keywords: VIDEOWALL, VIDEOPANEL, VIDEOCARD, RAISER, VIDEOSERVER, CONTROL SOFTWARE.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ОС – операційна система

ПК – персональний комп’ютер

RSS (Rich Site Summary) – збагачене зведення сайту

API (Application Programming Interface) – інтерфейс прикладного програмування

RAM (Random Access Memory) – оперативна пам’ять

HDD (Hard Disk Drive) – жорсткий диск

IDE (Integrated Development Environment) – інтегроване середовище розробки

ПЗ – програмне забезпечення

ФС – фізичний сервер

АЗ – апаратні засоби

LCD (Liquid Crystal Display) – рідкокристалічний дисплей

GPU (Graphics Processing Unit) – відеокарта

CPU (Central Processing Unit) – центральний процесор

SCADA (Supervisory Control And Data Acquisition) – диспетчерське управління та збір даних

WSL (Windows Subsystem for Linux) – підсистема операційної системи Linux для Windows

БД – база даних

DDoS (Distributed Denial-of-service) – розподілена атака на відмову

ВСТУП

З розвитком технологій та відео індустрії з'явилася необхідність відображення контенту видимого на великій відстані з достатньою яскравістю та чіткістю. Якщо для реклами найчастіше досить відображення статичного контенту, у вигляді намальованого або надрукованого зображення, то для більшості диспетчерських це не підходило. Використання існуючих моніторів було обмежено їх розміром і не дозволяло бачити всю картинку цілком, через що доводилося постійно перемикати зображення. Деякі використовували надруковані схеми, які доповнювалися лампами різних кольорів, що відображають поточний статус об’єкту. Трохи пізніше з'явилися великі TV-екрани, але навіть зараз їх розміри за мірками відеостін недостатні. Все це тривало до тих пір, поки в 1980 не придумали об'єднати кілька екранів в один логічний, отримавши тим самим аналог сучасної відео стіни.

Технологія відеостін відмінно підходить для різних напрямків, починаючи від технологічних диспетчерських і рекламних щитів, до оформлення інтер'єру і засобів віртуальної присутності, наприклад, тренажери для навчання льотчиків, кранівників та ін.

Найбільш типовим використанням, є диспетчерські центри, в яких можуть стежити як за виробництвом, так і за літаками, аваріями на підстанціях, причому все це може бути в розрізі міста або цілої країни. Їх область застосування дуже широка, можна застосовувати скрізь, де потрібно дивитися на процес цілком, розуміючи в будь-який момент часу, що відбувається в глобальному масштабі.

Контрольні та командні пункти дозволяють прослідковувати величезні обсяги інформації, щоб побачити будь-який збій в системі, вчасно відреагувати на якусь подію, стежити за безпекою або якимись даними. Можна організовувати пункти відеоспостереження, де вийде стежити за ситуацією на всьому об'єкті з одного місця.

Зазвичай відео стіни використовують на вокзалах, в аеропортах, школах, для відображення розкладу, з можливістю оперативно відображати додаткову інформацію про затримку транспортного засобу, кількість вільних місць та будь-яку іншу інформацію. Причому все це може відбуватися в реальному часі.

В сучасних реаліях відео стіни також застосовуються до оформлення інтер'єру і навіть мистецтва. Все це стало звичайним явищем, тепер багато впроваджують унікально виглядаючи конструкції лише щоб виділиться та привернути увагу.

Головним недоліком систем які пропонують на ринку є використання пропрієтарних компонентів як у апаратних рішеннях, так і у програмних. Завдяки цьому, при поламці, купити зламаний компонент дуже непросто. Велика частина виробників впроваджують спеціалізований захист і заміна комплектуючих несе за собою втрату працездатності, через заблоковані ліцензії, або обладнання взагалі не вмикається. Доводиться замовляти його у вендора, та чекати до кількох тижнів заміну, тому що в вільному продажу їх не знайдеш, а отримання з країн виробника нешвидке.

Взагалі, сьогодні існує досить велика кількість дешевих відеокарт з підтримкою декількох дисплеїв, що в теорії дозволяє зібрати відео стіну самостійно зі звичайних комплектуючих. Однак є ряд обмежень з якими доведеться боротися.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Опис предметного середовища

Відеостіна – це система декількох пристроїв відображення об'єднаних в єдиний логічний екран, що дозволяє виводити зображення з різних джерел і керувати ним.

Завдяки своїй модульній конструкції стало легко збирати відео стіни величезних розмірів. Причому форма конструкції може бути будь-який, від кола зі смуги екранів, до реальної стіни у вигляді прямокутника [1]. А сучасні гнучкі дисплеї дозволяють будувати практично будь-які форми, обмежені лише фантазією. Приклад звичайної відеостіни можна побачити на рисунку 1.1.

Рисунок 1.1 – Відеостіна на базі LCD паналей

Пристрій відеостін досить простий, що є однією з їхніх переваг. Найчастіше це чотири основні складові – засіб відображення, контролер та джерело сигналу та засоби кріплення пристроїв відображення [2].

Як засоби відображення, можуть служити різні LCD-панелі, проєктори, відео-куби та ін. джерела [3]. При цьому їх можна комбінувати, в залежності від складності поставленого завдання. У кожного джерела відображення є свої плюси і мінуси, які розглянемо трохи пізніше, але основний плюс більшості – це модульність, завдяки чому стіну можна збирати і розбирати. При цьому збирається вона досить компактною, і легко транспортується.

Для контролю відображення використовуються спеціальні контролери. Основна їх функція передати зображення з джерела на пристрої виведення, в потрібній позиції і потрібного розміру. Контролерів існує безліч варіантів, при цьому принцип роботи використовуються різний.

Наприклад, є контролери, які, з точки зору джерела подаваного сигналу, об'єднують панелі в єдиний екран, будь-то відеокамера або ПК, вони будуть вважати, що працюють з простим монітором або телевізором. Є контролери, які дозволяють приймати кілька джерел відображення і розміщувати їх в певних позиціях відео стіни. Найбільш просунуті мають можливість перемикати джерела, компенсувати шви між моніторами і багато іншого.

Джерело сигналу – це може бути, контент від побутових пристроїв з відеовиходом, до веб-сторінок і віддалених комп’ютерних робочих столів користувачів. Все залежить від функціоналу контролера.

На поточний момент найбільш популярними є чотири технології. Це LCD панелі, відеокубів, проектори та LED панелі. Розглянемо більш докладно їх плюси і мінуси.

LCD панелі – найбільш поширений варіант для побудови відеостін. Приклад панелей з їх характеристиками можна побачити на рисунку 1.2. Сучасні екрани давно навчилися якісно відображати картинку під будь-яким кутом, мають достатній запас яскравості, для показу зображення як вдень, так і вночі, можуть працювати в режимі 24/7 [4]. Панелі легко монтуються на більшість стін, не вимагаючи додаткових конструкцій, але найчастіше для них формують спеціальні конструкції, які дозволяють легко відсувати панелі від стіни, для легкого демонтажу, ремонту на місці або підключенню кабелів.

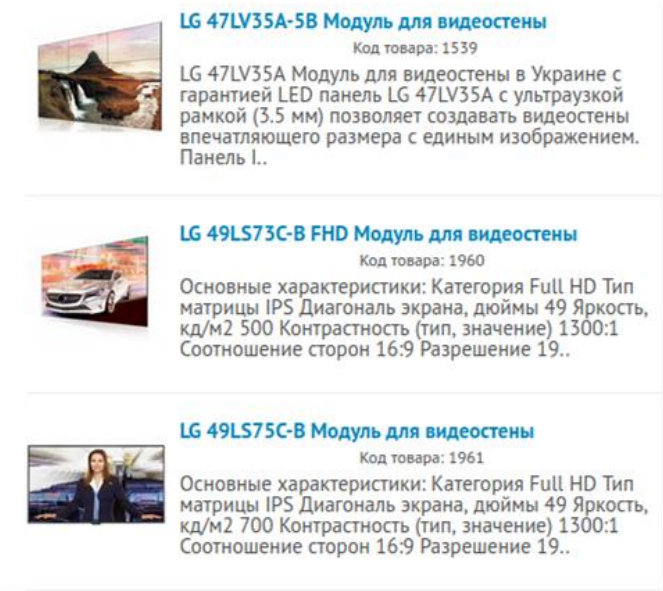


Рисунок 1.2 – Приклад сучасних панелей

Більшість виробників має свій набір панелей, спеціально призначених для цілодобової роботи протягом тривалого часу, вимірюваного роками. Такі панелі мають невелику вагу, відчутно малі габарити і низьке енергоспоживання. Найчастіше панелі несуть в собі розширений функціонал, що дозволяє без додаткових засобів, об'єднати кілька панелей в одну логічну, завдяки чому можна значно знизити вартість контролера.

Основні плюси при використанні панелей – це чудова якість зображення, тому що роздільна здатність п'ятдесяти дюймових панелей може доходити до 7680×4320 пікселів, а використання AMOLED технологій дозволяють їм впевнено почувати себе в сонячний день.

Панелі займають середню цінову категорію, у порівнянні з опонентами. Застосовуються в більшій мірі у відеостінах середніх розмірів, де потрібна висока якість при перегляді з близької відстані.

З мінусів можна відзначити вигорання, після тривалого використання, обмеження в розмірі однієї панелі та найголовніший недолік – наявність швів між ними. Хоча варто відзначити, що сучасні панелі мають шви по півтора міліметра, а останні розробки дозволяють від них позбутися. Тому, враховуючи їх розміри та вагу, найскладніше буде правильно їх змонтувати.

Найбільш дешевий аналог панелям – це проектори.

Проектор працює практично на будь-які поверхні, легко масштабується до потрібних розмірів. За великим бажанням, можна замінити кілька панелей одним проектором, який спроектує зображення на всю стіну.

Типів проекторів існує досить багато, є як звичайні лампові так і короткофокусні лазерні. За аналогією з панелями, у виробників зазвичай є моделі призначені для цілодобової роботи. При цьому найбільш просунуті моделі можуть мати кілька джерел світла, які дозволяють мати резерв, на випадок виходу одного джерела з ладу, та протриматися на половині яскравості до заміни. Контролери дозволяють оперувати проекторами як звичайними ТВ моніторами, що дозволяє використовувати кілька проекторів одночасно. Сучасні лампи та лазери дозволяють використовувати проектори у режимі 24/7. Все це робить їх досить привабливими для використання при побудові відеостени. Приклади проекторів можна побачити на рисунку 1.3.

Основні плюси проекторів – серденя ціна, можливість легкого масштабування зображення під потрібний розмір, дуже компактні габарити, використання одного проєктору замість декількох LCD-панелей [5].

Рисунок 1.3 – Приклади проекторів

При цьому мінусів досить багато, один з них – це фокусна відстань, яка вимагає розміщення проекторів на певній відстані від місця проектування. Другий – це дуже

низька фактична роздільна здатність, що зводить нанівець можливість збільшення зображення, тому що з близької відстані розглянути дрібні деталі буде проблематично. Третій і мабуть головний мінус – низька яскравість, тому сонячним днем відображення сильно страждає.

Область застосування – приміщення та, переважно, відео контент.

Ще одним типом пристроїв виводу зображення є відеокуби. Це такий гібрид панелей та проекторів. Джерелом виведення зображення служить невеликий проектор, який за допомогою системи дзеркал проектує зображення на вбудований екран. Основною перевагою служить порівняна компактність і безшовність зображення. Застосовують у закритих приміщеннях, де потрібна висока якість та чітке зображення. Один з мінусів таких пристроїв – ціна. Так само такі пристрої мають обмежений максимальний розмір куба, але він дещо біліший за LCD панелі. Цей недолік можна обійти використовуючи їх велику кількість, але це потребує більшої кількості відеокарт, що також значно збільшує кінцеву вартість. Приклад сучасних відеокубів доступних на ринку можна побачити на рисунку 1.4.

Рисунок 1.4 – Приклад сучасних відеокубів

В даний час набирає популярність ще один тип відображення – світлодіодні панелі та їх варіації.

Як джерело світла використовується досить велика трійка (rgb) над’яскравих світлодіода, яскравість яких регулюється, в залежності від необхідного відтінку [6]. Такі «трійки» збираються в кластери і за допомогою спеціальних контролерів керуються. Все це дозволяє домогтися колосальної яскравості та відображати контент в будь-який час доби, при цьому можна практично необмежено масштабуватися. Такі

пристрої часто можна побачити на стінах торгових центрів, розміром в декілька поверхів.

Головний мінус такого рішення – це кінцева роздільна здатність зображення, тому що через вимоги до яскравості, застосовуються світлодіоди величезного розміру, особливо в порівнянні з розміром пікселя у LED-панелях. Але з урахуванням, що застосовується дана технологія на понад великих відео стінах, а перегляд контенту передбачається на дальніх відстанях, низька роздільна здатність нівелюється.

Існують так само досить недорогі, екзотичні рішення, наприклад, Samsung Magic Info та її аналоги від LG [7]. Вони використовують вбудоване в панелі спеціалізоване ПО, яке може відображати відеоконтент, погоду, веб сторінки і управляти ними централізовано. Це робить їх дуже привабливими для використання в рекламних цілях. Але вони мають один великий недолік – запустити стороннє програмне забезпечення, наприклад, для відображення будь-якого технологічного процесу, неможливо.

За формування і розподіл зображення відповідають спеціальні контролери. Вони існують у вигляді кінцевих рішень, але найчастіше їх виготовляють під конкретне завдання. Їх можна порівняти з потужними відеостанціями, що мають кількість виходів, яка відповідає встановленим панелям або іншим пристроям відображення.

Контролери так само мають певну кількість відеовходів, які служать для джерел сигналів, що плануються до виводу на відео стіну. Контролери можуть управлятися як і внутрішнім спеціалізованим ПО, так і встановленим додатком на підключеному до нього ПК.

Основне завдання контролера, отримання будь-якого відеосигналу з набору джерел і трансляція його на відео стіну. Зазвичай контролер формує зображення, роздільна здатність якого становить сумарну роздільну здатність всіх пристроїв відображення. Тому вони використовують спеціалізовану залізну начинку і досить дорогі. Часто займають майже половину бюджету всього проєкту. Контролери дозволяють накладати кілька джерел друг на друга, управляти ними, переміщувати по відео стіни, реагувати на події та змінювати контент щодо подій. Так само деякі

контролери враховують зазори між панелями, роблячи текст більш читабельним і зображення більш цілісним та гладкішим.

Ключові вимоги до більшості контролерів – можливість динамічного, ручного або зв’язаного з подіями, контролю відображення.

Не варто забувати про необхідність монтажу всієї конструкції. На перший погляд це досить просте завдання, але в більшості випадків потрібно передбачити можливість легкого доступу до будь-якої панелі, для обслуговування кабелів, врахувати вагу всієї конструкції, мати можливість регулювати зазори і нахил панелей.

Для вирішення цих завдань, ключові виробники мають свої рішення з побудови опорних конструкцій. Всі вони повинні відповідати вимогам пожежної безпеки, мати можливість швидкого і легкого монтажу, без використання зварювання, або інших способів і відповідати всім вимогам вище.

Хоча в деяких випадках допускається простий монтаж на стіну, з використанням спеціального кріплення. Приклад кріплення LCD панелей зображено на рисунку 1.5

Рисунок 1.5 – Приклад кріплення панелей

1.2 Огляд наявних аналогів

1.2.1 Контролери відеостін АМС

Відеоконтролери AMC розроблені для захоплення, обробки і передачі різних типів відеосигналів на відео стіни [8]. Це обладнання забезпечено сучасними процесорами та великим об'ємом оперативної пам'яті, що забезпечує високу продуктивність. Приклад контролеру зображено на рисунку 1.6

Рисунок 1.6 – Контролер AMC

Завдяки великому корпусу має можливість змінювати конфігурацію обладнання від простих моделей, до найбільш потужних, що використовують найпотужніше апаратне обладнання.

Основні характеристики:

- резервування ключових елементів;

RAID 0, 1, 5, 10;

- захват відеосигналів: SD video, HD video, DVI, RGB, компонентний, HD-SDI, IP-потоки;

- роздрібна здатність кожного відео-виходу до 2560 x 1600;

- цілодобовий режим роботи;

- кастомізація за запитом.

Обладнання збирається згідно зі специфікаціями проєкту, тому є можливість підібрати комплектацію під будь який бюджет.

До вибору доступно:

- кількість входів;
- кількість виходів;
- об’єм оперативної пам’яті;
- можливість резервування блоків живлення;
- один чи два процесори;
- RAID 0, 1, 5, 10;
- об’єм жорсткого диску;
- кількість блоків живлення;
- кількість мережевих карт;
- тип корпусу.

Пристрій являє собою досить гнучку до конфігурування пропозицію, що дозволяє зібрати відео стіну під великий спектр потреб. Можна придбати як конфігурацію порівнянну з базовим ПК, так і більш відмовостійкі варіанти з серверних комплектуючих.

Має спеціалізоване власне ПО для управління, але функціонал досить обмежений. Дозволяє лише зберегти розташування джерел сигналу, для виведення на відео стіну. Тому використання зазвичай обмежується демонстрацією відео контенту, чи віддалених робочих столів. Часто це змушує мати додаткові персональні комп’ютери для виводу корпоративних додатків, зображення з яких виводиться за допомогою протоколу RDP, чи VNC.

Мінуси – використання застарілої ОС, дуже завищена ціна у порівнянні з ПК такої ж потужності. Програмне забезпечення потребує активації, тому у разі перестановки операційної системи, потрібно звертатися до постачальника, який наддасть новій серійний номер для активації.

Приклад комплектації контролеру у трьох різновидах від бюджетної до спеціальної, надано у таблиці 1.1.

Таблиця 1.1 – Комплектація ФС контролеру АМС

Специфікація	Бюджетна	Стандартна	Спеціальна

Корпус	Rack 4U/Tower/малошумний - опціональний	Rack 4U/Tower/малошумний - опціональний	Rack 4U/Tower/малошумний - опціональний
Процесор	4-ядерний Intel Core i7 (3-4 gen)	4-ядерний Intel Core i7 (3-4 gen)/Intel Xeon E5-2600	2 x Intel Xeon E5-2600
Оперативна пам'ять	8 Гб	8 Гб	16 Гб (8 Гб кожний процесор)
ЛВС	Один (10, 100, 1000 Мбайт/с)	Два (10, 100, 1000 Мбайт/с)	Два (10, 100, 1000 Мбайт/с)
Жорсткий диск	Від 500 Гб (1 HDD), швидка заміна – опціональна	Від 500 Гб (1 HDD), RAID 1 опціональний, швидка заміна	От 1000 Гб (2 HDD) RAID 0, 1, 5 – опціональний, швидка заміна
Клавіатура и миш	Да	Да	Бездротова
Операційна система	Windows 7 x 64 Bit Professional	Windows 7 x 64 Bit Professional	Windows 7 x 64 Bit Professional
Виходи	До 80 HD	До 80 HD	До 80 HD
Входи	Більш 200 HD (DVI, RGB) або 24 SD	Більш 200 (DVI, RGB) або 24 SD	Більш 200 HD/UHD (DVI, RGB) або 48 SD
ПЗ	Без віддаленого доступу	Віддалений доступ через ЛВМ, зберігання конфігурації робочого столу	Віддалений доступ через ЛВМ, зберігання конфігурації робочого столу

1.2.2 Контролери Christie

Компанія займає одну з головних позицій на ринку у сфері професійних рішень для Enterprise сегменту [9]. Пропонує готові рішення для трансляцій, центрів безпеки, навчання, управління процесами, комунікаціями та ін.

Реалізує повний спектр передових продуктів та технологій, які можна легко інтегрувати для створення найсучасніших рішень для відеостін. Можливо обрати одну технологію дисплея або об'єднати кілька платформ, щоб створити дисплеї, які всі помістять. Продукція Christie підтримується провідним у галузі навчанням, гарантіями та технічною підтримкою. На таблиці 1.2 розглянемо один з контролерів компанії Christie TVC-1700.

Таблиця 1.2 – Контролер Christie TVC-1700

Шассі	2U form factor
CPU	2.4 GHz Quad-Core Xeon CPU
ОС	Microsoft Windows
RAM	8GB DDR3 SDRAM, optional 32GB DDR3 SDRAM
HDD	1 x 1TB 3G SATA 7200RPM, optional 3x 1TB 3 G SATA 7200RPM hot-swappable, redundant
Оптичний привід	DVD SATA Optical Drive
Мережевий доступ	2 x Gigabit Ethernet
Порти	4 x USB 2.0 ports, 1 x RS-232 serial port video-wall control
ПЗ	Full featured Christie MASTERSuite™
Демонстрування віддалених ПК, шт	100 (simultaneous), nearly unlimited (defined)
Кількість дисплеїв	64
Роздрібна здатність на дисплей	2560 x1600 @ 60Hz
Графічна пам'ять	512MB per card

Це один з потужних контролерів компанії. Використовує серверні комплектуючи, свої пропрієтарні відеокарти та ПО для управління.

ПО має досить великий функціонал, дозволяючи створювати сценарії запусків, має планувальник, віддалене управління, функції безпеки і ін. Але ціна втроє вище аналогічної пропозиції вище.

1.2.3 Контролери VuWall

Екосистема продуктів VuWall збудована на унікальній гібридній технології VuTrexTM, яка забезпечує найбільш ефективне розповсюдження візуальної інформації, від будь якого джерела до будь якого типа дисплея в організаціях. VuTrex з'єднує AV, IT та IP системи с гібридним та заснованим на стандартах підходом до взаємодії [10]. Приклади конфігурацій зображені на рисунку 1.7

VuScape	VS10	VS20	VS60	VS120/280
IDEAL FOR	Single display or small video wall with limited space & network sources only	Small to midsize video wall requiring rackmount unit and additional outputs	Small video wall with limited space and need for GPU & decoding	Midsize video wall with more inputs & outputs and need for GPU & more decoding power
FORM FACTOR	VESA Mount	Rackmount 1RU	Tower	Rackmount 4RU
MAX OUTPUTS	2x 4K	4x 4K	4x 4k	16x 4K
MAX INPUTS	–	–	4x 4K	24x 4K
MAX SIMULTANEOUS IP STREAMS	16x HD / 2x 4K	16x HD / 2x 4K	16x HD / 4x 4K	96x HD / 24x 4K
NETWORK SOURCES	RTSP, websites, VNC, pictures, text, video files	RTSP, websites, VNC, pictures, text, video files	RTSP, websites, VNC, pictures, text, video files	RTSP, websites, VNC, pictures, text, video files
WINDOWS 10 APPLICATION	✔	✔	✔	✔

Рисунок 1.7 – Приклади контролерів VuWall

Компанія пропонує серію контролерів VuScape – це високопродуктивний відеостінний процесор, який ідеально підходить для диспетчерських, кімнат для

співпраці та корпоративних вивісок. Легко контролює та розподіляє усі джерела вмісту на будь-якому дисплеї у мережі за допомогою максимальної гнучкості та повнофункціонального програмного забезпечення для управління.

Контролери базуються на серії з п’яти базових модулів, які мають гнучку конфігурацію. Функціонал програмного забезпечення на рівні контролерів Christie. Контролер має можливість встановлення відеоадаптерів з можливістю 3d прискорення (NVIDIA Quadro GPU)

Програмне забезпечення прив’язане до обладнання, та у разі зміни комплектуючих, потрібно його активувати знову.

1.3 Постановка задачі

Наведені вище приклади показують, що багато пропозицій на ринку, є звичайними ПК або серверами, укомплектованими декількома відеокартами, основною цінністю яких є програмне забезпечення. Все це стало можливим завдяки розвитку комп'ютерних комплектуючих та значного збільшення їх продуктивності за останні роки.

Отже, по-перше, потрібно проаналізувати існуючи на ринку споживачеві апаратні комплектуючи, щоб обрати оптимальні варіанти для побудови апаратної складової та уникнути можливих обмежень. Створити тестовий стенд та перевірити можливість використання даного рішення.

По-друге, потрібно розробити програмне забезпечення, яке задовільнить базові потреби використання відеостін.

1.3.1 Апаратна частина

Потрібно проаналізувати обмеження з боку достатності ліній шини PCI Express, як правило процесор і південний міст, як зображено у таблиці 1.3, можуть підтримувати декілька десятків ліній, але материнські плати, доступні кінцевому споживачеві, мають досить обмежену кількість роз'ємів для відеокарт. Частина з них

зайнята різною периферією, наприклад, USB або SATA роз'єми, проте велика частина вільна.

Таблиця 1.3 – Кількість ліній PCIe

Процесор	Кількість ліній
AMD Ryzen 3000 series	24 for GPU
Intel 9800	16 for GPU + півд. міст
AMD Threadripper	Up to 48 for GPU

Розвиток майнінгу криптовалют в останні роки змусив багатьох вендорів випустити спеціалізовані материнські плати (наприклад, Biostar TB250-BTC PRO), на яких можуть розміщуватися до 12 роз'ємів PCIe. При цьому ціна їх не набагато більше ніж на звичайні материнські плати.

Є ще одне рішення цієї проблеми – це так звані «Райзери». Райзери дозволяють з одного слоту PCIe x4 зробити 4 слоти PCIe x1, що дозволяє використати практично всі лінії під наші потреби. Їх існує досить багато варіацій, на 1, 2, 4 та 8 слотів, при цьому за досить демократичну ціну. Звичайно, при цьому страждає пропускна здатність, але якщо не використовувати 3D додатки, це не критично. Приклад використання можна побачити на рисунку 1.8.

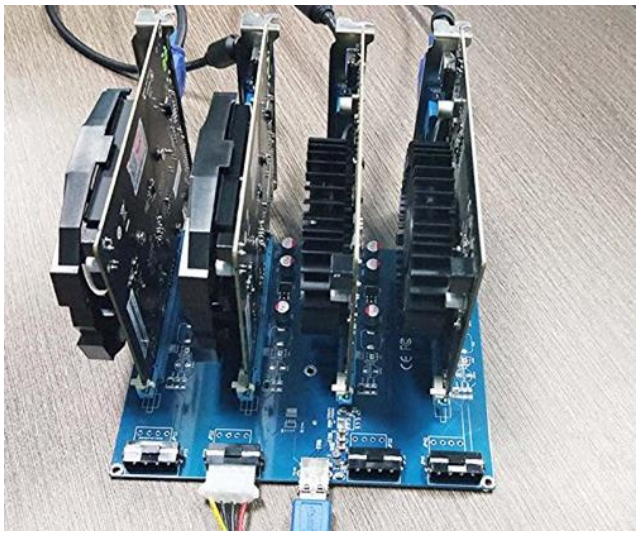


Рисунок 1.8 – Використання райзера на чотири карти

Як виявилося райзери вирішують ще одну проблему – розташування та тепловиділення.

Сучасні відеокарти мають великі габарити і дуже часто займають декілька слотів розширення, при цьому не використовують їх. Тому, завдяки райзерам, відеокарти можна розмістити в окремому корпусі паралельно вирішуючи проблему зняття тепловиділення.

Їх можна розмістити як у відкритому корпусі, на спеціальних стійках, так і в закритому, забезпечивши їх належним охолодженням. Приклад такого розташування можна побачити на рисунку 1.9. Така збірка дозволяє побудувати стіну розміром 6x4, при прямому підключенні засобів виводу зображення. Також при такому розташуванні забезпечується гарний теплообмін, та відеокарти не перегріваються.

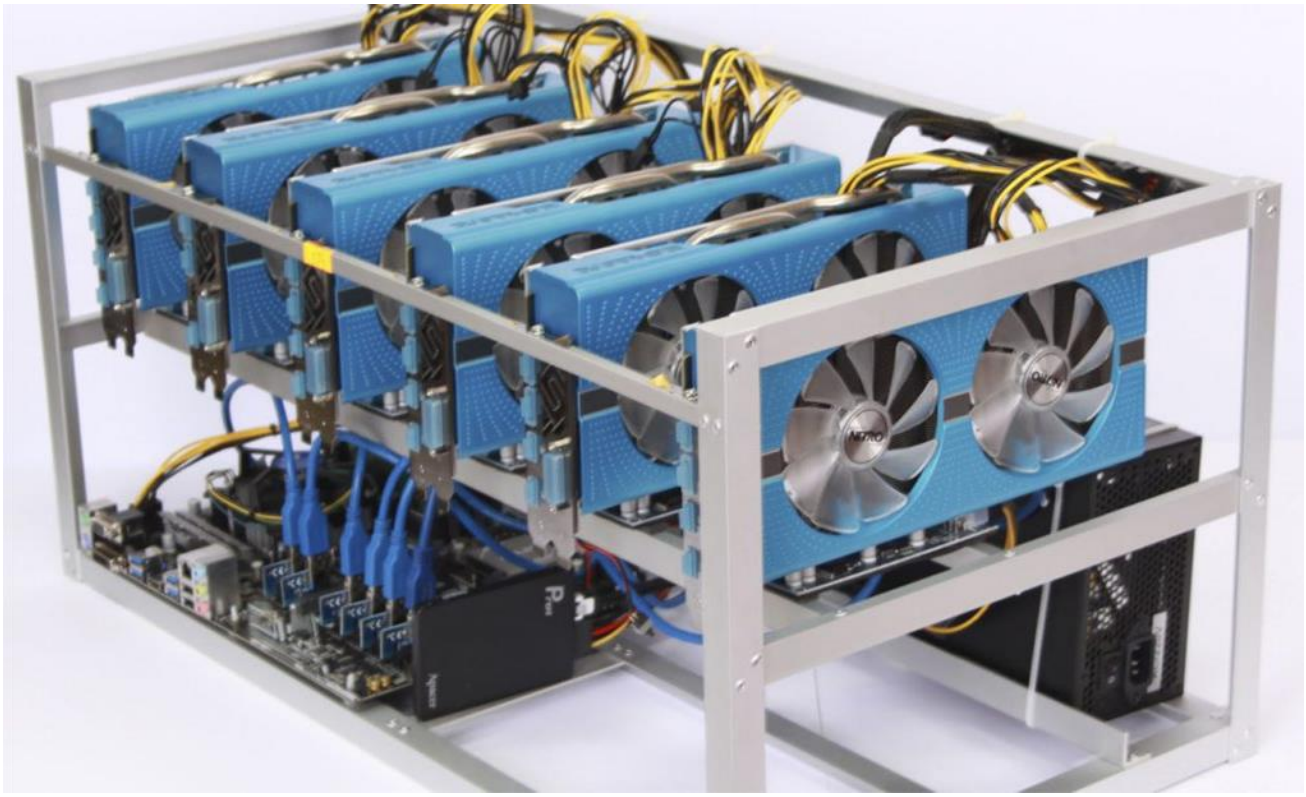


Рисунок 1.9 – Зовнішнє розташування відеокарт

Таким чином ми можемо легко розмістити на звичайній споживчій материнської плати десяток відеокарт, що навіть при наявності усього одного відеовиходу на платі, дозволить побудувати стіну 3x3, з 55 дюймових панелей або проєкторів, що буде достатньо для більшості диспетчерських. А з урахуванням наявності на більшості

відеокарт 3-4х відеовиходів стають доступні величезні стіни розмірами 6х4 та більше, що підходить майже для всіх галузей. Але якщо навіть такої кількості панелей недостатньо, є можливість їх об’єднання у логічні дисплеї. Тобто використовуючи один вихід відеокарти, можливо задіяти відразу декілька панелей, які перетворюються на одну логічну, що робить розмір стіни достатнім до будь якого випадку.

Також існує ще один спосіб наростити кількість відеокарт – це USB рішення, наприклад, від виробника DisplayLink. Вони досить дешеві та, якщо не потрібно відтворювати 3D контент, цілком придатні. Завдяки можливості використання USB хабів та швидкості третьої версії інтерфейсу, можливо додавання десятків таких пристроїв, але при цьому зазвичай росте навантаження на центральний процесор, тому при використанні такого підходу треба мати це на увазі.

У разі потреби виводу зображення з аналогових пристроїв, можна використати TV/DVB-тюнер . На ринку є достатньо пропозицій у виконанні як PCIe карт, так і з USB підключенням. Майже всі моделі мають аналогові входи для антени та S-Video, що дозволяє демонструвати телебачення, або виводити зображення з плеєра оптичних дисків чи аналогових відеокамер.

1.3.2 Програмна частина

Окремо від розробки програмного забезпечення слід розглянути обмеження операційних систем та драйверів.

Сучасні операційні системи не мають обмежень за кількістю встановлюваних відеокарт, але виробники драйверів іноді це роблять.

Наприклад, для найпоширеніших моделей від AMD та NVIDIA, кількість не може перевищувати 8 та 12 карт, для Windows 7 та Windows 10 відповідно. Але це не заважає комбінувати та використовувати ці відеокарти спільно.

Отже, з боку апаратних комплектуючих та операційної системи є обхідні варіанті, але залишається не вирішеним одне питання, як всім цим керувати.

Всі рішення від світових вендорів поставляються разом із програмним забезпеченням для керування відеостін. У деяких випадках це дуже обмежений

функціонал, який являє собою відеоплеєр, для програвання відеоконтенту з заданої папки, або дуже функціональні рішення, які дозволяють інтегруватися з корпоративними сервісами, запускати спеціалізоване ПЗ, мають спільне управління, розширені налаштування безпеки і багато іншого.

Звичайно, можливо обійтися і без всього цього, використовуючи відео стіну як великий робочий стіл, а необхідні програми запускати вручну. Але з огляду на розміри стіни і відстань до неї, розглянути курсор мишки буде практично неможливо, а постійний ручний запуск та розміщення вікон, через перезавантаження, викликане оновленнями ОС, або проблемами електроживлення, зводять таке використання нанівець.

Отже, головною задачею буде створення програмного додатку для керування відеостіною. Додаток повинен мати функціонал схожий до можливостей конкурентів, але залишати можливість змінювати апаратні комплектуючи для швидкого ремонту, модернізації чи масштабуванню.

Висновок до розділу 1

Розглянуті на ринку контролери з різних цінових категорій демонструють, що апаратно вони представляють собою звичайній комп’ютер, основною цінністю якого є програмне забезпечення. Це показує, що рішення з споживчих комплектуючих можуть використовуватися, при цьому вони будуть набагато дешевше запропонованих брендових аналогів. Але для використання необхідна розробка програмного забезпечення керування відеостіною, що і буде основною складністю.

При цьому можна не хвилюватися за працездатність такого рішення, навіть при використанні у режимі 24/7. Майнери криптовалют, за останні кілька років, довели, що в такому режимі звичайне обладнання може працювати роками, особливо якщо врахувати, що на відео стіни навантаження на відеокарту значно нижче, ніж при майнингу.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

У розробці проєкту критично важливою частиною має бути визначення вимог, тому по-перше треба провести їх аналіз. Зазвичай аналіз проводять з участю замовників чи бенефіціарів, але в нашому випадку ми створюємо проєкт який має конкурувати з існуючими на ринку рішеннями. Тому можливо початі процес з аналізу можливостей конкурентів та визначити вимоги на прикладі великого підприємства.

Основними перевагами результатами аналізу буде:

- надання чіткого списку вимог;
- формування контракту між спонсорами проєкту, та розробниками;
- для великої системи можливе надання опис високого рівня [11].

2.1 Функціональні вимоги

Розглянемо функціональні вимоги до системи. Вони мають описати її внутрішню роботу та поведінку. Та у більшості випадків визначаються на початковому етапі розробки.

Отже, система має підтримувати виконання наступного функціоналу:

- управління функціоналом з віддаленого робочого місця;
- авторизація користувача в системі;
- вивід зображення на усі типи пристроїв, які операційна система визначає як дисплей;
- вивід контенту:
    1) додатків операційної системі;
    2) веб контент;
    3) відео контент;
    4) інформативні строки;
    5) слайд шоу;
- позиціювання контенту:
    1) переміщення у задані координати;
    2) зміна розміру;

3) керування накладанням;

- запуск та закриття контенту:

1) додавання нового контенту;

2) редагування існуючого;

3) видалення;

- відображати функціональне та реальне зображення стану відеостени;

- запуск сценаріїв (групи додатків одночасно у заданих координатах дисплею визначеного розміру):

1) додавання нових сценаріїв;

2) редагування існуючих;

3) видалення сценаріїв;

- відображення контенту чи запуск сценаріїв згідно часу (розкладу):

1) додавання нового розкладу;

2) редагування;

3) видалення;

- відображення робочого столу, або його частини з віддаленого персонального комп'ютеру;

- відображення контенту методом переміщення його з віддаленого комп’ютеру на додаток (drag`n`drop);

- можливість масштабування зображення у додатку керуванням;

- підтримка різних мов інтерфейсу;

- функція менеджера підключень:

1) можливість додавати нові контролери;

2) редагувати підключення;

3) видаляти;

- функція налаштувань:

1) збереження налаштувань;

2) зміна часу оновлення у функціональному та графічному режимі;

3) зміна якості зображення;

4) зміна часу мережевого таймауту;

5) зміна мови інтерфейсу додатка;

6) налаштування координатної сітки;

7) зміна поведінки прилипання вікон при переміщенні (snap windows);

8) налаштування кольорів інтерфейсу.

Для успішної реалізації, та кращого розуміння, зробімо діаграму варіантів використання, яка використовується для графічного відображення функціональності програмного забезпечення, рисунок 2.1. Детальна схема представлена у додатку Г.

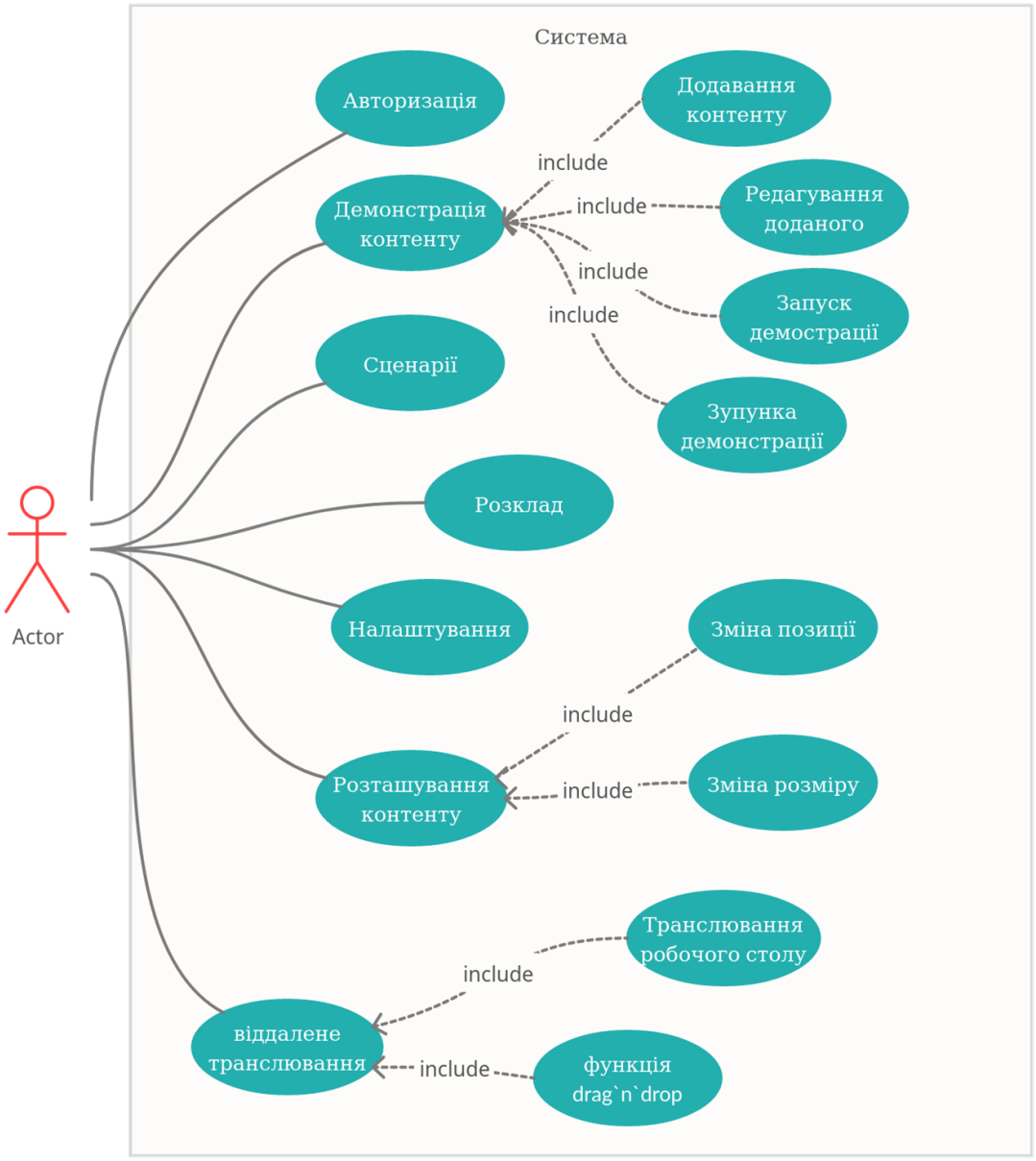


Рисунок 2.1 – Спрощена діаграма варіантів використання

2.2 Нефункціональні вимоги

Для відображення обмежень та показників системи використовують нефункціональні вимоги.

Основними вимогами до нашої системи мають бути:

- безпека:

1) підключення за паролем – забезпечить безпечне користування, та захистить від несанкційованих дій;

2) шифрування трафіку – оскільки додаток е мережевою системою, потрібно захистити інформаційний обмін між додатками;

3) шифрування налаштувань – захистить від несанкційованої зміні та доступу до налаштувань, там як там можуть бути параметри запуску додатків та паролі;

4) підтримка ролей безпеки – наддасть можливість обмежувати права користувачів;

- система має адаптуватися до будь якої кількості дисплеїв, та їх роздільної здатності – це має надати можливість оперативно додавати та змінювати пристрої виводу зображення, без рестарту та переналаштування системи;

- системі потрібен простий та зрозумілий інтерфейс користувача – оскільки за системами можуть працювати люди різного рівня підготовки;

одночасний доступ до системи – система має мати можливість працювати у диспетчерських центрах, де керувати стіною можуть декілька працівників одночасно;

- висока швидкість обробки дій користувача та системи – у деяких галузях використання відеостін, потрібна швидка реакція на зміни, наприклад, охоронні системи;

- автоматичний перезапуск серверного додатку у разі збою (відмовостійкість) – у разі використання системі у критичних напрямках, потрібно захистити систему від збоїв, та забезпечити швидке відновлення працездатності;

- можливість працювати на обладнанні споживчого сегменту – основна мета роботи це здешевлення та спрощення побудови відеостін;

- підтримка багатопоточності – наддасть можливість масштабувати систему, бо сучасні процесори мають багато фізичних ядер;

адаптування до каналу передачі даних – при використанні інтерактивного режиму, або транслювання робочого столу користувача, потрібно передавати графічну інформацію, тому треба мати можливість адаптувати якість зображення до швидкості підключення;

- можливість відновлення останнього стану запущених сценаріїв та додатків – забезпечить швидке відновлення роботи у разі нештатного, або перезавантаження комп’ютеру після оновлень операційної системи, чи додатків;

ведення журналу команд – дає можливість зрозуміти хто та коли робив якісь зміни у налаштуванні та позиціюванні контенту.

Зазначені вище вимоги дадуть можливість створити конкурентоспроможну систему, яка матиме необхідний функціонал, існуючий у більшості вендорів на ринку. Завдяки підтримки обладнання споживчого сегменту, система забезпечуватиме можливість швидкого ремонту та масштабуванню.

Висновки до розділу 2

У даному розділу були визначені основні функціональні та не функціональні вимоги, на основі оцінки функціоналу конкурентів та потреб поточного підприємства. Зображена діаграма варіантів використання, з якої можливо зрозуміти основний функціонал системи керування відеостіною.

Вимоги демонструють основні потреби користувача та програмні обмеження, на базі яких потрібно обрати технології, розробити архітектуру та створити систему керування.

3 ВИБІР ТА ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Для розробки програмного забезпечення потрібно обрати операційну систему, стек технологій та розробити архітектуру програмного забезпечення. Це головним образом матиме вплив на складність та час розробки програмного забезпечення. Також це опосередковано впливатиме на швидкість та функціонал системи. Ще одним пунктом перед початком розробці має бути вибір мінімальних апаратних вимог, на які потрібно орієнтуватися при будуванні системи.

3.1 Вибір операційної системи

Одним з перших кроків перед початком розробці програмного забезпечення повинен бути вибір операційної системи. Це матиме вплив на апаратну платформу та потрібний стек технологій.

Для цього можна почати з аналізу популярності операційних систем. Існує досить багато сервісів, які збирають цю статистику. Якщо подивитися на один з найбільших statcounter.com, то можливо побачити статистику використання різноманітних пристроїв та їх операційних систем, дивись рисунок 3.1.

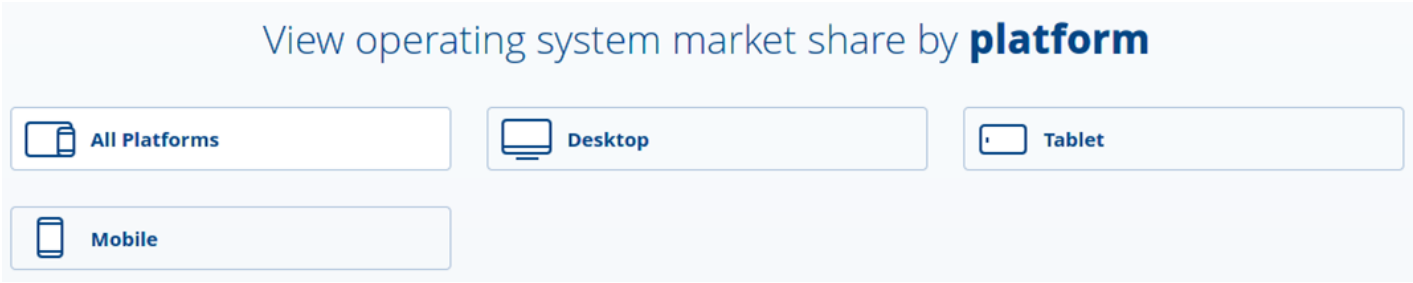


Рисунок 3.1 – Вибір платформи

Для нашої системи потрібно орієнтуватися на персональні комп’ютери, так як доволі велика ніша потенційних клієнтів це корпоративні підприємства та демонстрування корпоративного програмного забезпечення потребує значних ресурсів. Також мобільні пристрої в більшості не мають можливість до підключення багатьох пристроїв виводу зображення.

Тому дивимось статистику по персональним комп’ютерам. Як бачимо з рисунку 3.2, найпопулярнішою операційною системою є Microsoft Windows з 76,32%, а на другому місці йде OS X з 17,65% [12].

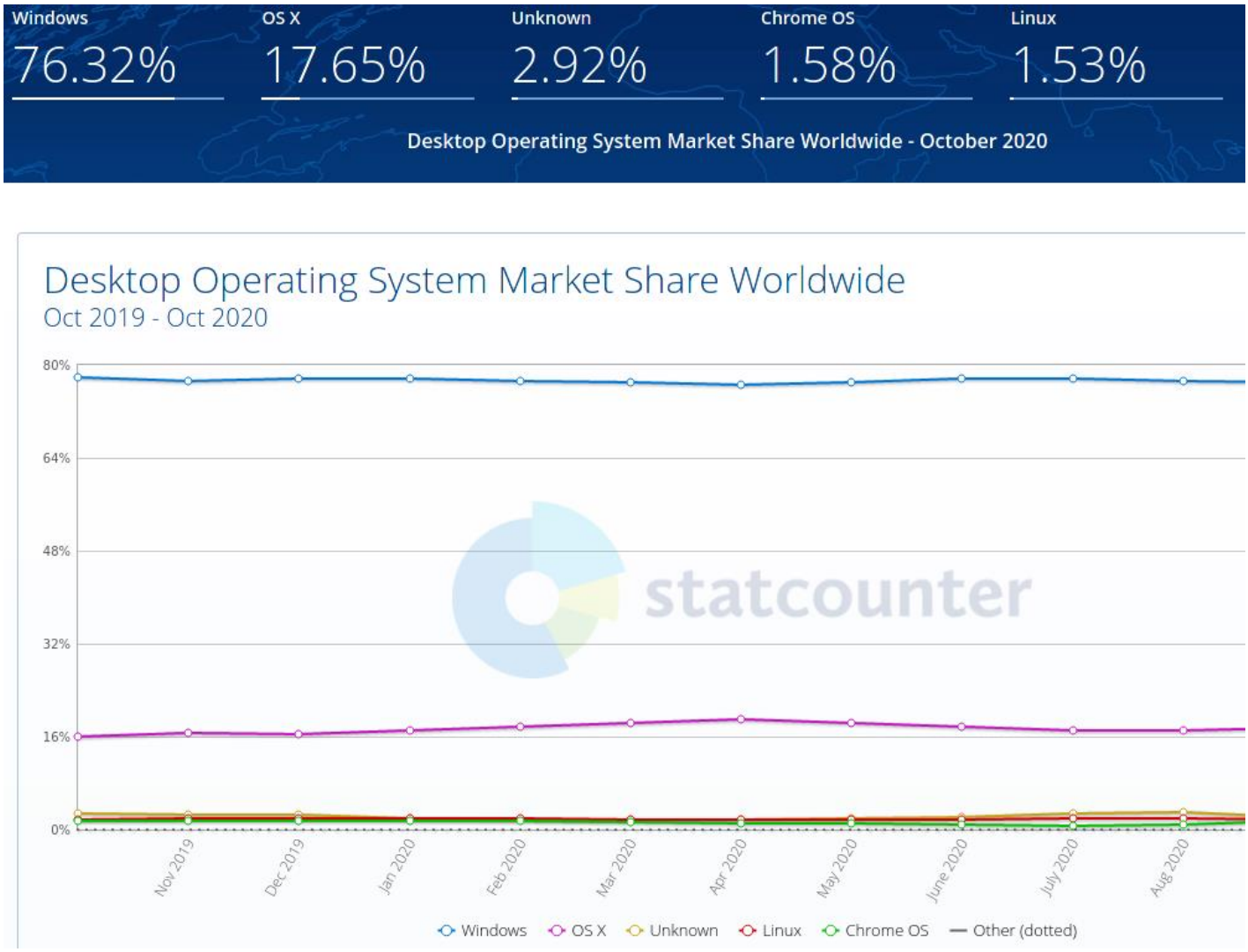


Рисунок 3.2 – Найпопулярніші операційні системи

Якщо подивитися на OS X, то окрім нижчої популярності ніж Microsoft Windows, в системи є ряд недоліків, які не дозволяють використовувати її у розробці цього проєкту. По-перше система має прив’язку до обладнання, що йде всупереч з темою роботи, по-друге багато програмного забезпечення корпоративного сегменту до цієї операційної системи взагалі не існує. Також розробка до цієї операційній системі доволі ускладнена через отримання сертифікатів та публікацію в AppStore. Більшість систем йде з монолітними корпусами, та обмеженою кількістю портів для периферії.

Отже, на даний момент найперспективнішою операційною системою є Microsoft Windows, додатковим аргументом у її користь має корпоративний сегмент, який у більшості працює на даній операційній системі, так як користуються такими технологіями як Active Directory, System Center Datacenter та ін. Також більшість SCADA систем, які потребують велика кількість диспетчерських не має портів під інші операційні системи.

Ще одним напрямком на сьогодні є крос-платформа, але це дуже ускладнить розробку так як система має керувати вікнами, запускати додатки, а це робиться через низькорівневе API операційної системи, яке дуже відрізняється. Тому у даному випадку складність росте у пропорції з кількістю операційних систем. Також необхідно враховувати, що у останніх версіях Windows існує підсистема Linux (Windows Subsistem for Linux), що у парі з X-сервером наддасть можливість виконувати та керувати Linux додатками, дивись малюнок 3.3. Єдине що доведеться зробити це встановити такий сервер у операційну систему.

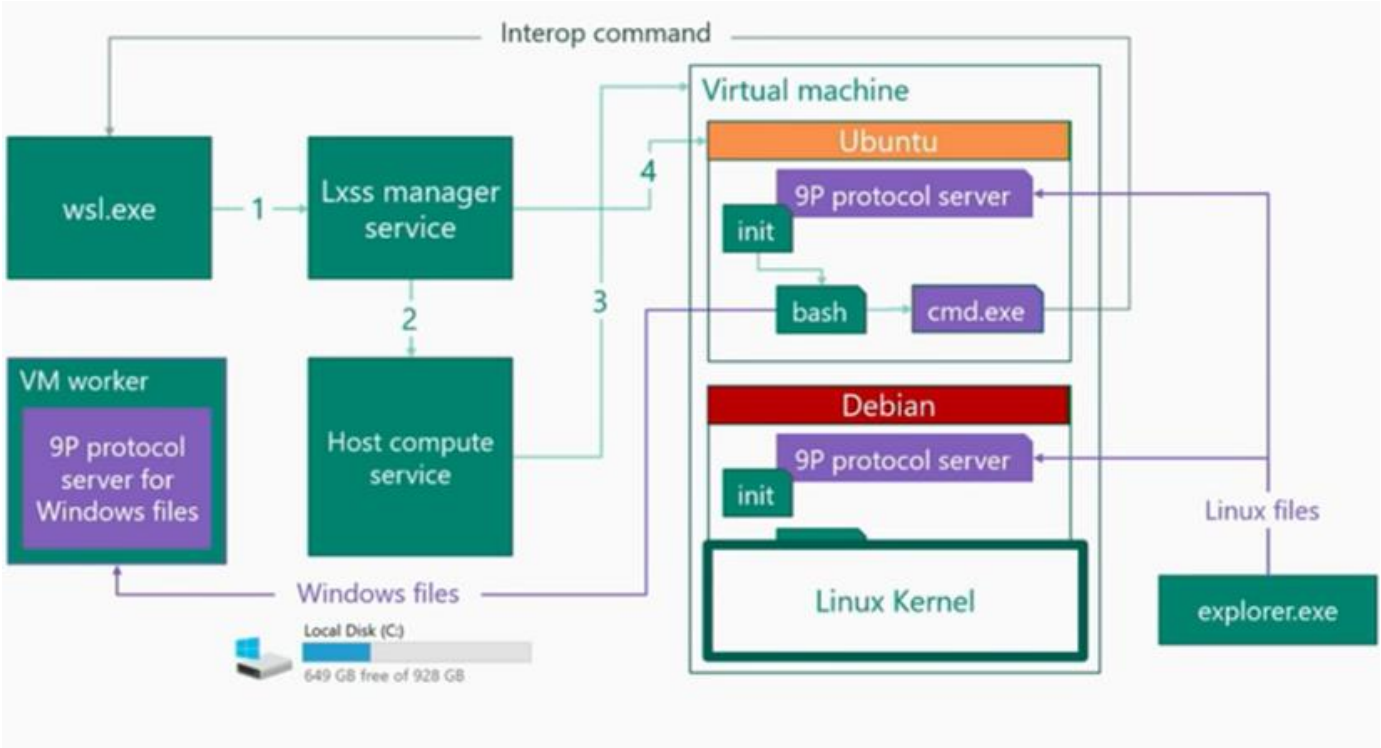


Рисунок 3.3 – Архітектура WSL2

Актуальною на даний момент клієнтською версією є Windows 10, серверною Windows Server 2019, отже, їх підтримка має бути обов'язковою. Також є підприємства

які використовують застаріле програмне забезпечення, яке не працює на Windows 10, тому непогано буде реалізувати підтримку Windows 7.

Microsoft Windows має досить просунуте низькорівневе API, яке дозволяє керувати позиціюванням вікон та їх розміром, а також запускати та завершувати додатки, що є вкрай важливим для реалізації системи.

Також навіть клієнтська редакція операційній системи дозволяє створювати повнофункціональні мережеві додатки.

Тому для розробки даної системі обрана операційна система Microsoft Windows.

3.2 Розробка архітектури програмного забезпечення

Якщо подивитися на проєкт, то становиться зрозумілим, що розробку архітектури треба розділити на дві частини — програмну та апаратну.

Апаратною частиною виступатиме персональний комп’ютер з одною, або декількома відеокарт, до яких підключені пристрої виводу зображення. У часті персонального комп’ютера може виступати будь яка система на базі x86 та x86_64 сумісних апаратних комплектуючих. Це можуть бути як персональні комп’ютери споживчого сегменту так і сервера корпоративного класу. Головне обмеження, це наявність драйверів та сумісність з операційною системою Windows, яка була обрана у попередньому розділи. Також необхідно правильно підібрати блок живлення, бо велика кількість відеокарт потребує високої потужності. У звичайних випадках достатньо пари відеокарт, майже всі моделі за останні п’ять років мають мінімум три виходу для підключення пристроїв виводу зображення. Це дасть можливість створити відестіну 2x3, що при підключенні шести 55 дюймових TV-панелей утворить відестіну 3,6x2 метри.

При недостатності місць або наявності великого тепловиділення у корпусі, можливо використовувати райзери та зовнішній корпус для відеокарт, схематичне зображення можливо побачити на рисунку 3.4. Це наддасть можливість встановити багато відеокарт та покращить їх охолодження, бо можна буде використовувати зовнішні системи охолодження. Детальна логічна схема представлена у додатку В.

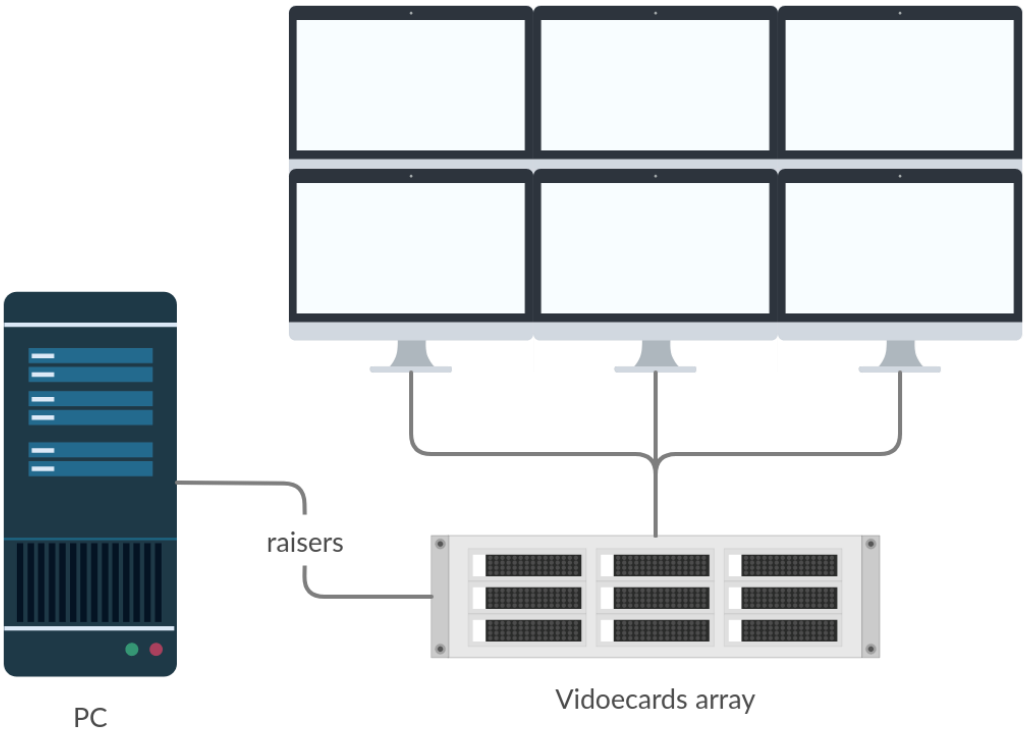


Рисунок 3.4 – Схематичне зображення підключення апаратної частини

Проміжну роль займе операційна система, яка буде керувати відеокартами та пристроями виводу. З технічною точки зору в нас є великій робочий стіл, на якому ми запускаємо додатки та маємо мати можливість ними керувати.

Оскільки управляти контентом на такому розмірі майже не можливо, так як з маленької відстані неможливо побачити все зображення цілком, а з великої не видно курсор миші.

Отже, керувати стіною доведеться з клієнтського персонального комп'ютера, який знаходиться у мережевий доступності з відеостіною.

З боку програмної частини це означає, що доведеться використовувати клієнт серверну архітектуру.

З боку персонального комп'ютера відеостіни потрібно реалізувати програмний додаток серверу, який повинен мати можливість приймати команди від клієнтського додатку та їх виконувати. Серверний додаток повинен працювати непомітно до користувача і не відображатися на відеостені. В вимогах до системи вказано, що керувати системою можуть одночасно декілька користувачів, тобто серверній додаток

повинен бути багатопотоковим. Сервер повинен мати можливість виконувати всі вимоги до проєкту.

Для збереження даних найкраще використати базу даних, тому що її легко масштабувати як під прості, так і великі рішення.

У якості клієнта повинний бути програмний додаток, якій дозволить керувати сервером, посилаючи на нього команди, та відображати поточний стан стіни. Отже, фінальна програмна архітектура має вигляд який зображено на рисунку 3.5. Детальніше схеми побудови клієнтської та серверної частини можна побачити у додатку Б та додатку Г.

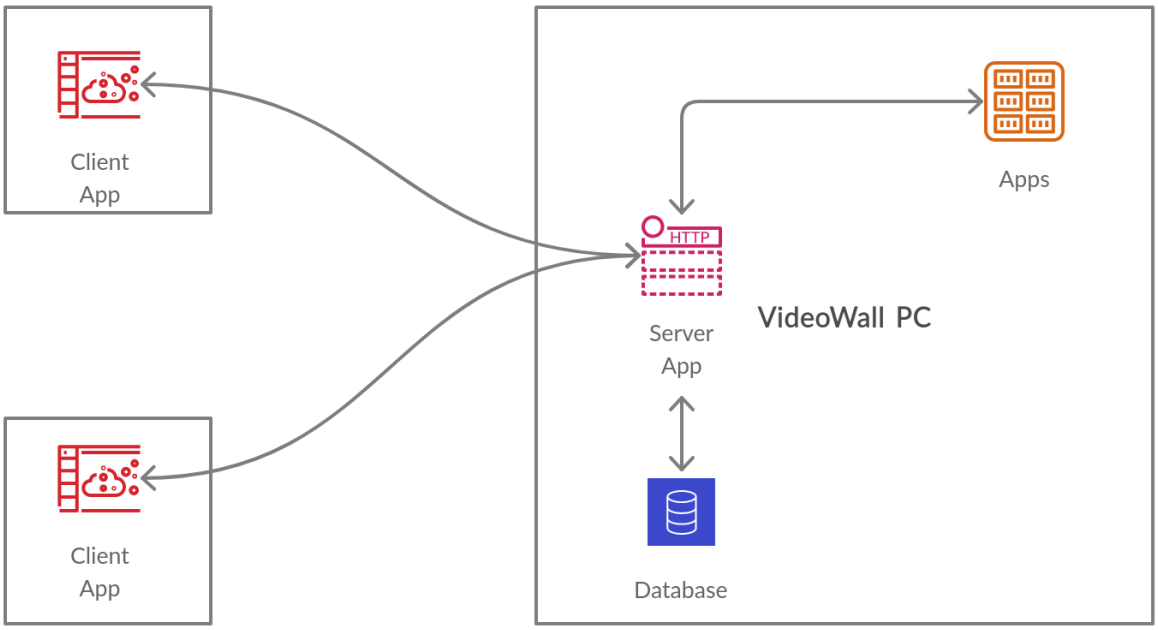


Рисунок 3.5 – Програмна архітектура

3.3 Вибір програмного стека технологій

Перед побудовою програмного додатка потрібно обрати стек технологій які допоможуть реалізувати програмну частину проєкту. Це буде мова програмування, допоміжні бібліотеки, база даних та IDE середовище для побудови програмного коду.

Спочатку розглянемо базовий язик програмування та графічну бібліотеку до нього. Бажано щоб клієнтський та серверний додаток використовували один і той самий стек технологій. Це спростить та прискорить розробку.

Розглянемо найпопулярніші на сьогодні язики програмування, за версією www.northeastern.edu, як бачимо на малюнку 3.6, у першу четвірку входять Python, JS, Java та C# [13].

Рисунок 3.6 – Найпопулярніші мови програмування

Усі вони мають одну чи декілька графічних бібліотек для створення графічного інтерфейсу користувача.

Для java це swing, для C# це WPF, для Python це QT та для JavaScript це Electron. Розглянемо кожний окремо.

Python – доволі проста інтерпретована мова програмування. Один з найкращих інструментів для обробці великих об’ємів даних та швидкого рішення невеликих задач. Але разом з програмою доведеться тягнути з собою інтерпретатор (існують методи його компіляції, але це додатковий рівень розробки, та має проблеми з безпекою). Також він не має вбудованої графічної бібліотеки, тож доведеться тягнути за собою ще й її дистрибутив. Що у кінцевому разі буде потребувати їх постійного оновлення.

JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування. Скриптова мова яка виросла з браузерів, але з появою таких речей як Electron та NodeJS, дозволяє робити повноцінні додатки. Але для проєкту вони не підходять, оскільки не вміють взаємодіяти з WinAPI.

Java – об'єктно-орієнтована мова програмування, використовує для роботи віртуальну машину. Має декілька графічних бібліотек, одна з яких це Swing. Java дозволяє робити додатки любої складності, та має віртуальну машину під різні операційні системи. При використанні цієї мови програмування є декілька недоліків, потрібно постійно слідкувати за оновленням віртуальної машини, потребує дещо більших ресурсів, та використання WinAPI дещо ускладнене.

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET, архітектура якого представлена на рисунку 3.7.

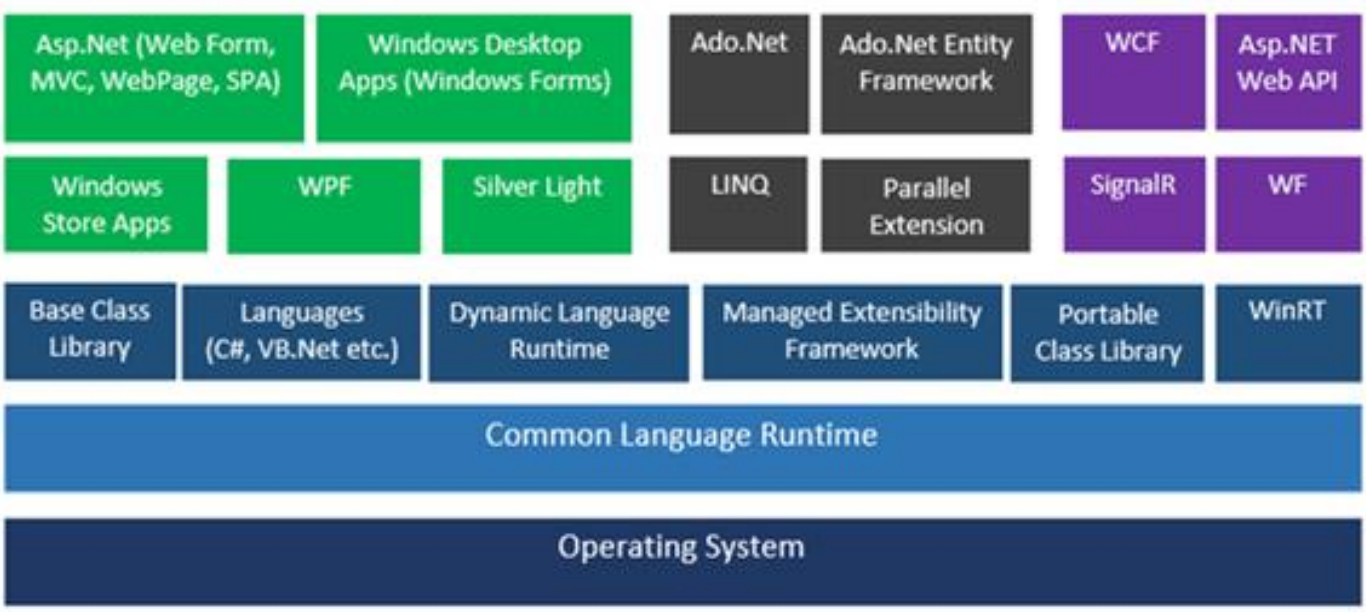


Рисунок 3.7 – Архітектура .NET Framework

.NET Framework – програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків [14]. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Однією з ідей .NET є сумісність служб, написаних різними мовами. У поточний час розвивається паралельно відкритому аналогу .NET Core, який у майбутньому має обладнати обидві платформи.

Має декілька графічних бібліотек, наприклад WPF, архітектуру якого можно побачити на рисунку 3.8, який є аналогом WinForm, але використовує векторну систему візуалізації та мову розмітки XAML [15]. Має високу продуктивність роботи, тому що базується на технології DirectX

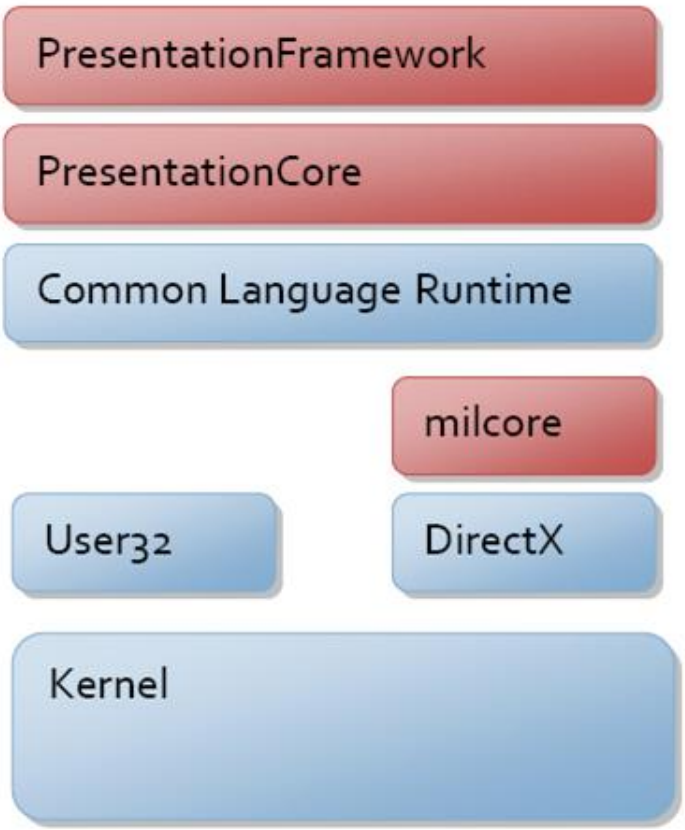


Рисунок 3.8 – Архітектура WPF

Основними перевагами є те, що .Net поставляється разом з дистрибутивом операційної системи, тому непотрібно слідкувати за оновленням. Має можливість зручними способами використовувати WinAPI. Тому для проєкту це буде найкращий вибір.

Для збереження налаштувань найкраще підійде база даних. У поточний момент існує багато відкритих та пропрієтарних варіантів, наприклад, Oracle, MS SQL, Mongo, MariaDB, Postgres та ін. Але у нашому випадку не потребується бази які тримають високе навантаження. Потрібно зберігати лише параметри безпеки та дані додатків. Тому найкращим вибором буде вбудована база даних SQLite, архітектуру

якої можливо побачити на рисунку 3.9. Вона не пострибує окремого встановлення, достатньо використати потрібну бібліотеку.

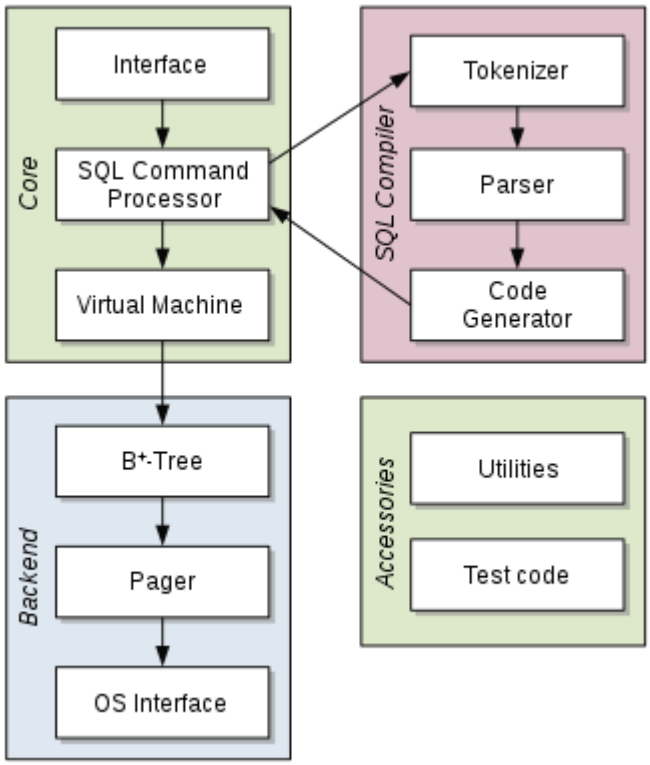


Рисунок 3.9 – Архітектура SQLite

Середовище розробки потрібно вибирати з оглядом на технології та мову програмування яка буде використовуватися. Опираючись на обрану мову C# найкраще підійдуть IDE Visual Studio, Visual Studio Code та JetBrains Rider.

Rider – IDE написано на Java, досить молодий проєкт, але має багатий функціонал. Основна його цінність це вбудований Reshaper, та крос-платформа. Найбільш цінний при розробці під net core в операційній системі Linux. Основний недолік його ціна.

VS Code – легкий та швидкий платформонезалежний редактор, який за допомогою доповнень перетворюється на повноцінну IDE під безліч мов програмування. Основним мінусом виступає невеликий функціонал.

Visual Studio – рідне середовище розробки від Microsoft. Має найбільший функціонал, приклад на рисунку 3.10, та безплатні версії.

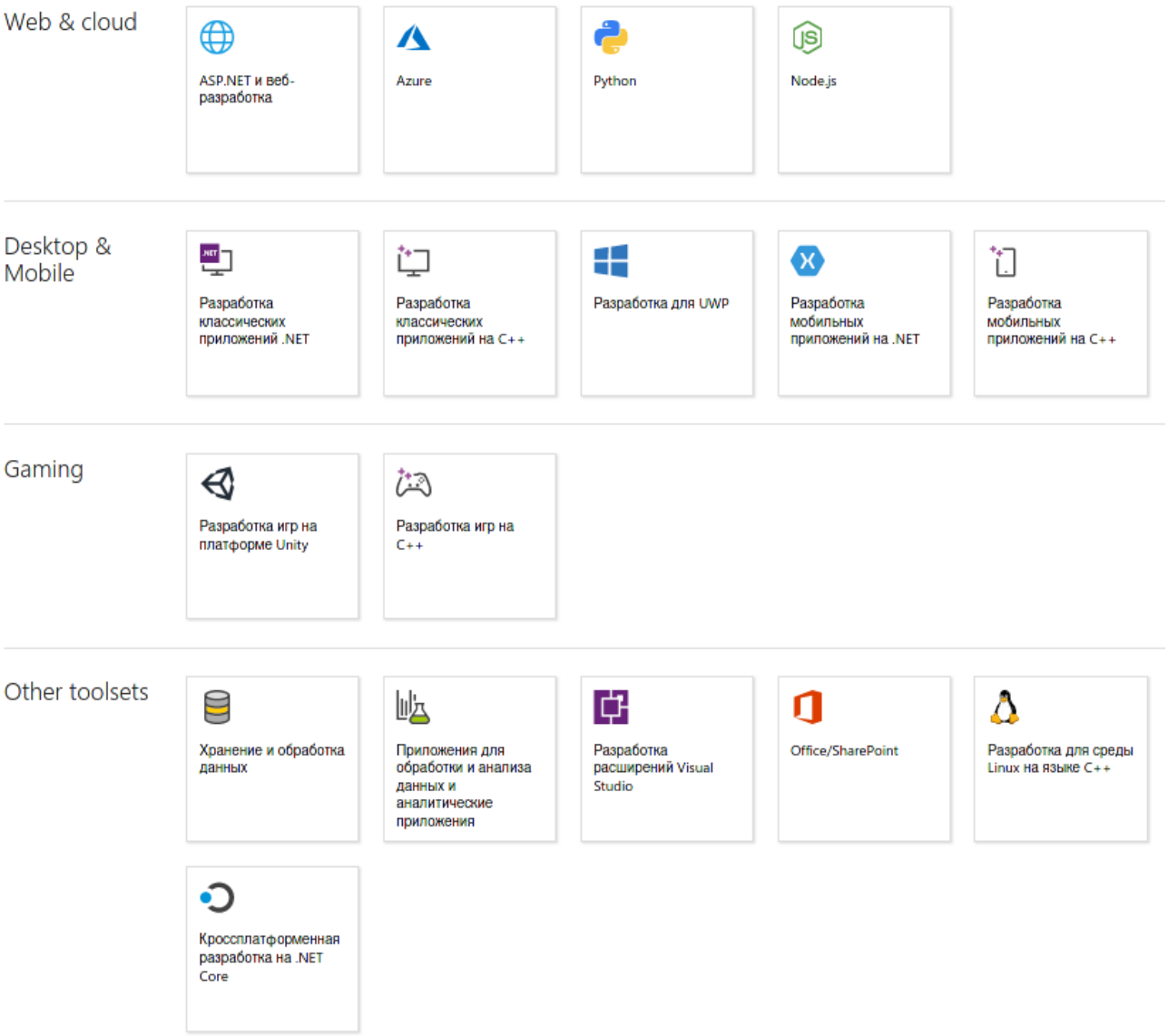


Рисунок 3.10 – Можливості Visual Studio

Найбільш інтегрована під розробку на UWP та WPF. Основним недоліком є складність налаштування. Можна підключати багато доповнень, що дає можливість розширювати функціонал та додавати інші мови програмування.

Отже, найкращим середовищем для розробки проєкту буде Visual Studo.

Оскільки планується сумісність з Windows 7, то бажано використовувати версію вбудованого до системі .Net. Це наддасть можливість не додавати до дистрибутиву інсталяційний пакет. Але у цієї версії доволі складно зробити гарний інтерфейс користувача, стандартні компоненти є малофункціональним, тому спробуємо використати додаткову бібліотеку.

Одна з доволі популярних та частково вільних реалізацій це компоненти з набору Extended WPF Toolkit.

Для вільного користування доступні компоненти зазначені на рисунку 3.11.

AvalonDock	AutoSelectTextBox	BusyIndicator	Calculator
CalculatorUpDown	CheckComboBox	CheckListBox	ChildWindow
CollectionControl	ButtonSpinner	ByteUpDown	CollectionControlDialog
ColorCanvas	ColorPicker	DateTimePicker	DateTimeUpDown
DecimalUpDown	DoubleUpDown	DropDownButton	IconButton
IntegerUpDown	LongUpDown	Magnifier	MaskedTextBox
MessageBox	MultiLineTextEditor	PieChart	PrimitiveTypeCollEditor
PropertyGrid	RangeSlider	RichTextBox	RichTextBoxFormatBar
ShortUpDown	SingleUpDown	SplitButton	2 SwitchPanels
TimelinePanel	TimePicker	TimeSpanUpDown	ValueRangeTextBox
WatermarkPasswordBox	WatermarkTextBox	WatermarkComboBox	WindowContainer
WindowControl	Wizard	Zoombox	Windows 8 Theme

Рисунок 3.11 — Вільні компоненти з пакету Extended WPF Toolkit

Висновки до розділу 3

На основі виконаного аналізу у розділі 3, обрано сімейство операційних систем Microsoft Windows, на базі яких буде працювати система. Система буде базуватися на клієнт-серверної архітектурі. Для збереження даних буде використовуватися база даних SQLite. Мовою програмування зазначений C# на платформі .net framework з графічною бібліотекою WPF. У якості середи розробки найкращою буде Microsoft Visual Studio. Також для прискорення розробки та надання інтерфейсу клієнтської частини інтуїтивного виду, використовується бібліотека Extended WPF Toolkit.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

До початку програмування потрібно підготовити робочу середу. Встановити необхідну IDE, бібліотеки та компоненти.

Для початку встановлюється необхідна для Visual Stusio бібліотека .Net, на поточний це версія 4.6.2. Далі з сайту Microsoft встановлюється Visual Studio з набором опцій, як зображено на рисунку 4.1.

Після встановлення середа готова до початку розробки.

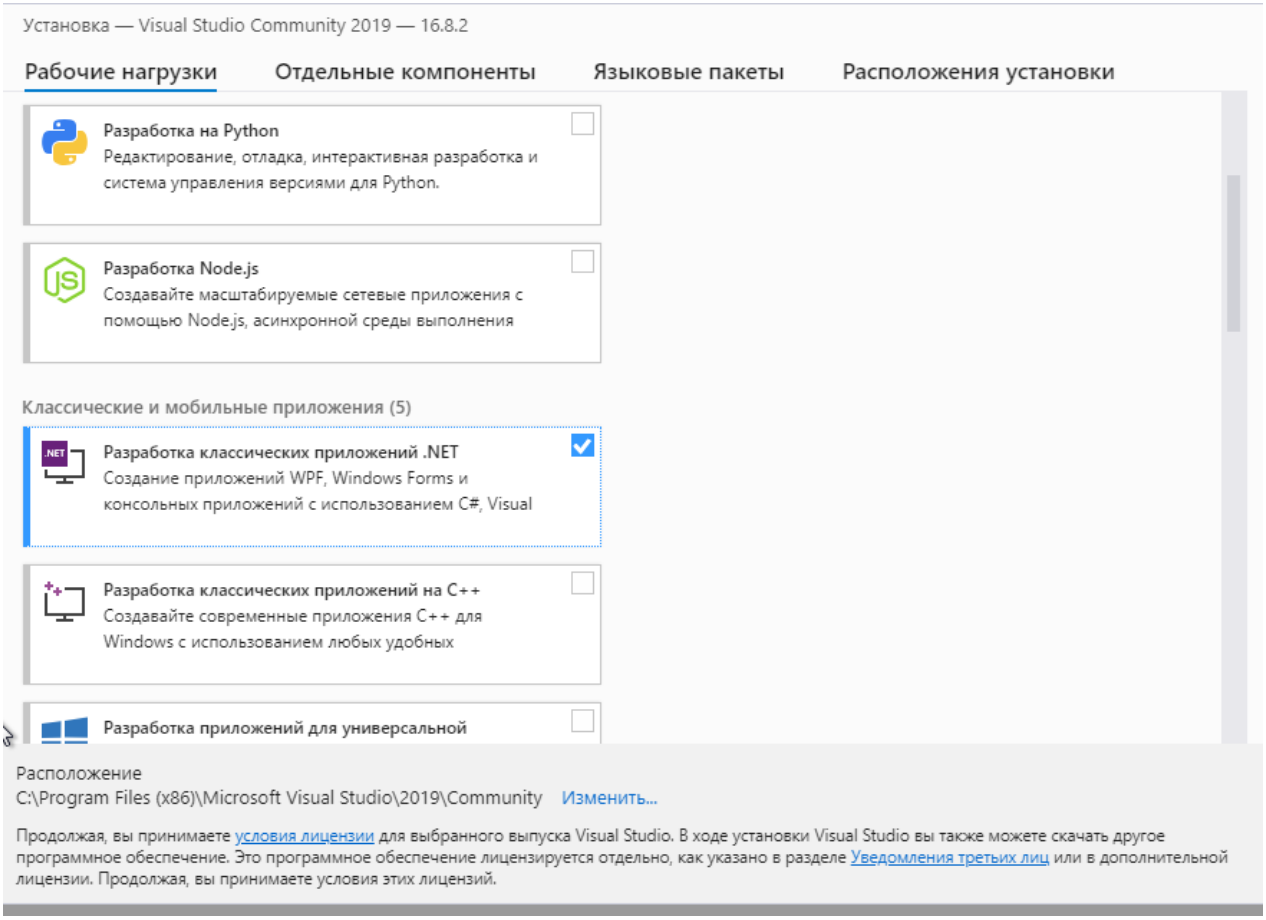


Рисунок 4.1 – Параметри встановлення Visual Stusio

4.1 Серверна частина

Спочатку треба реалізувати серверну частину додатку. Оскільки до серверу можуть звертатися відразу декілька клієнтів, він повинен бути багатопотоковим.

Для реалізації створені класи Server та Client. Клас сервер використовує TcpListener, який за допомогою усього двох параметрів може створити Tcp сервер[16].

На початку утворюється TCP сервер, а далі, як можливо побачити на рисунку 4.2, у безкінцевому циклі при підключенні утворюються потоки які передають обробку до класу Client, та пишуть у журнал кожне звертання.

```
TcpListener Listener;
// start server
ссылка: 1
public Server(int port)
{
    // create listener for port
    Listener = new TcpListener(IPAddress.Any, port);
    Listener.Start();

    while (true)
    {
        ThreadPool.QueueUserWorkItem(new WaitCallback(ClientThread), Listener.AcceptTcpClient());
    }
}

ссылка: 1
static void ClientThread(Object StateInfo)
{
    Program.log.Add(((TcpClient)StateInfo).Client.RemoteEndPoint.ToString());
    new Client((TcpClient)StateInfo);
}
```

Рисунок 4.2 – Старт серверу, та обробка підключень

Клас Client, як бачимо на рисунку 4.3, приймає дані від клієнту та намагається конвертувати їх у строку. У разі невдачі повертає помилку. Якщо отримуємо кінець строки, закінчуємо обробку.

```
while ((Count = Client.GetStream().Read(Buffer, 0, Buffer.Length)) > 0)
{
    // try decode to text
    try
    {
        Request += Encoding.UTF8.GetString(Buffer, 0, Count);
        // drop if \r\n\r\n or > 4kB
        if (Request.IndexOf("\r\n\r\n") >= 0 || Request.Length > 4096)
        {
            Request=Request.Substring(0, Request.Length - 4);
            break;
        }
    }
    catch { Console.WriteLine("bad request"); return; }
}
```

Рисунок 4.3 – Обробка запиту

Також приймання переривається, якщо розмір запиту перевищує чотири кілобайти, бо зазвичай та у проєкті запроси такого розміру не використовуються. Якщо приймати запроси будь якого розміру, то на сервер буде дуже легко провести DDoS атаку [17].

Оскільки для обміну трафіком, за вимогами, використовується шифрування, після отримання строки запиту, треба її розшифрувати.

Для шифрування та декодування створено клас StringCipher, який, як можна побачити на малюнках 4.4 та 4.5, це реалізує за допомогою симетричного ключа[18]. Для роботи він приймає текст, та кодовий ключ, який знаходиться у програмі (його можливо змінити) через окремий параметр при старті серверу, після чого він зберігається у налаштуваннях.

```
public static string Encrypt(string plainText, string passPhrase)
{
    var saltStringBytes = Generate256BitsOfRandomEntropy();
    var ivStringBytes = Generate256BitsOfRandomEntropy();
    var plainTextBytes = Encoding.UTF8.GetBytes(plainText);
    using (var password = new Rfc2898DeriveBytes(passPhrase, saltStringBytes, DerivationIterations))
    {
        var keyBytes = password.GetBytes(Keysize / 8);
        using (var symmetricKey = new RijndaelManaged())
        {
            symmetricKey.BlockSize = 256;
            symmetricKey.Mode = CipherMode.CBC;
            symmetricKey.Padding = PaddingMode.PKCS7;
            using (var encryptor = symmetricKey.CreateEncryptor(keyBytes, ivStringBytes))
            {
                using (var memoryStream = new MemoryStream())
                {
                    using (var cryptoStream = new CryptoStream(memoryStream, encryptor, CryptoStreamMode.Write))
                    {
                        cryptoStream.Write(plainTextBytes, 0, plainTextBytes.Length);
                        cryptoStream.FlushFinalBlock();
                        var cipherTextBytes = saltStringBytes;
                        cipherTextBytes = cipherTextBytes.Concat(ivStringBytes).ToArray();
                        cipherTextBytes = cipherTextBytes.Concat(memoryStream.ToArray()).ToArray();
                        memoryStream.Close();
                        cryptoStream.Close();
                        return Convert.ToBase64String(cipherTextBytes);
                    }
                }
            }
        }
    }
}
```

Рисунок 4.4 – Шифрування тексту

При формуванні шифрованої строки використовується сіль, яка генерується випадково, та значно ускладнює варіанти декодування команд при перехваті трафіку. Завдяки цьому навіть однакові команди будуть виглядати інакше, та зловмиснику буде важко підібрати ключ до шифрування, або підставити свою штучну команду. Сіль передається разом з командою, але це не несе загрози, так як немаючі ключа, розшифрувати команду не вдасться.

```
public static string Decrypt(string cipherText, string passPhrase)
{
    var cipherTextBytesWithSaltAndIv = Convert.FromBase64String(cipherText);
    var saltStringBytes = cipherTextBytesWithSaltAndIv.Take(Keysize / 8).ToArray();
    var ivStringBytes = cipherTextBytesWithSaltAndIv.Skip(Keysize / 8).Take(Keysize / 8).ToArray();
    var cipherTextBytes = cipherTextBytesWithSaltAndIv.Skip((Keysize / 8) * 2).Take(cipherTextBytesWithSaltAndIv.Length
        - ((Keysize / 8) * 2)).ToArray();

    using (var password = new Rfc2898DeriveBytes(passPhrase, saltStringBytes, DerivationIterations))
    {
        var keyBytes = password.GetBytes(Keysize / 8);
        using (var symmetricKey = new RijndaelManaged())
        {
            symmetricKey.BlockSize = 256;
            symmetricKey.Mode = CipherMode.CBC;
            symmetricKey.Padding = PaddingMode.PKCS7;
            using (var decryptor = symmetricKey.CreateDecryptor(keyBytes, ivStringBytes))
            {
                using (var memoryStream = new MemoryStream(cipherTextBytes))
                {
                    using (var cryptoStream = new CryptoStream(memoryStream, decryptor, CryptoStreamMode.Read))
                    {
                        var plainTextBytes = new byte[cipherTextBytes.Length];
                        var decryptedByteCount = cryptoStream.Read(plainTextBytes, 0, plainTextBytes.Length);
                        memoryStream.Close();
                        cryptoStream.Close();
                        return Encoding.UTF8.GetString(plainTextBytes, 0, decryptedByteCount);
                    }
                }
            }
        }
    }
}
```

Рисунок 4.5 – Дешифрування тексту

Далі маючи розшифрований запит можливо його розкласти, як зображено на рисунку 4.6, та у залежності від команди виконати потрібний метод, рисунок 4.7.

Для розкладання використовуються пошук у команді текстових частин за якими визначається команда, далі шукаються її параметри, через спеціальні абревіатури та символи.

```
else if (Extension.IndexOf("CLOSE") == 0)
{
    string mhwnd = "";
    mhwnd = Extension.Substring(Extension.IndexOf("hwnd=") + 5, Extension.IndexOf(";") - Extension.IndexOf("hwnd=") - 5);
    CloseWnd(Convert.ToInt32(mhwnd, 10));
}
```

Рисунок 4.6 – Розкладання запиту

```
ссылка: 1
private void CloseWnd(int mhwnd)
{
    SetForegroundWindow(mhwnd);
    SendMessage(new IntPtr(mhwnd), WM_CLOSE, IntPtr.Zero, IntPtr.Zero);

}
```

Рисунок 4.7 – Метод закриття вікна

Для управління вікнами та процесами використовуються функції WinAPI, наприклад, імпорт функції яка повертає назву вікна, зображено на рисунку 4.8

```
[DllImport("user32.dll", EntryPoint = "GetWindowText",
ExactSpelling = false, CharSet = CharSet.Auto, SetLastError = true)]
Ссылок: 3
public static extern int GetWindowText(IntPtr hWnd, StringBuilder lpWindowText, int nMaxCount);
```

Рисунок 4.8 – Імпорт методу, який повертає назву вікна

У такий спосіб реалізовані усі команди (схему класів надано у додатку Ж):

- GET – повертає дані по усім вікнам, хендли, назву, координати та ін.;

- SET – приймає хендл вікна з координатами та розмірами, після чого змінює його;

- ACT – приймає хендл вікна та активує його, якщо воно було звернуто чи перебувало на задньому плані;

- CLOSE – приймає хендл вікна, та закриває його;

- MIN – мінімізує вікно за хендлом;

- MAX – максимізує вікно за хендлом ;

- RUN – приймає номер додатку з колекції та запускає;

- CLOSALL – завершує усі запущені сервером додатки;

- EXTALL – приймає ідентифікатор сценарію, та завершує його;

- GETSCN – відправляє статус за ідентифікатором сценарію;

- GETP – відправляє дані з бібліотеці додатків, за номером;

- RSPC – видаляє сценарій за номером;

- SPC – додає новий або корегує існуючий сценарій;

- UPDP – додає новий або корегує існуючий додаток;

- MLC – натискає ліву кнопку миші за поточними координатами;

- UMLC – відпускає ліву кнопку миші за поточними координатами;

- MMOV – переміщує курсор миші у задані координати;

- MRC – натискає праву кнопку миші за поточними координатами;

- UMRC – відпускає праву кнопку миші за поточними координатами;

- NRM – нормалізує вікно якщо воно максимізовано;

- MON – передає поточний скріншот всіх екранів;

- SCR – повертає поточну інформацію по дисплеям, позицію, роздрібну здатність та ін.;

- SF – приймає файл, та намагається його відкрити;

- CROP – приймає картинку та транслює її в вікні без рамок на екрані, якщо вікно з картинкою вже є на екрані, оновлює картинку;

- STOPCROP – закриває вікно з демонструванням картинки та припиняє передачу картинки до серверу.

Для захвату зображення, як можна побачити на рисунку 4.9, зчитуються дані з усіх моніторів та перекодуються у формат jpg, за параметрами які вказані в налаштуваннях клієнтської частини [19]. Це не найкращий метод, але даний режим потрібен не часто, лише щоб подивитися, що відображає відеостіна, та скоригувати поведінку додатків.

```
IntPtr hDesk = GetDesktopWindow();
IntPtr hSrce = GetWindowDC(hDesk);
IntPtr hDest = CreateCompatibleDC(hSrce);
IntPtr hBmp = CreateCompatibleBitmap(hSrce, sz.Width, sz.Height);
IntPtr hOldBmp = SelectObject(hDest, hBmp);
bool b = BitBlt(hDest, 0, 0, sz.Width, sz.Height, hSrce, minX, minY, CopyPixelOperation.SourceCopy | CopyPixelOperation.CaptureBlt);
Bitmap bmp = Bitmap.FromHbitmap(hBmp);
bmp = new Bitmap(bmp, new Size(sz.Width / d, sz.Height / d));
SelectObject(hDest, hOldBmp);
DeleteObject(hBmp);
DeleteDC(hDest);
ReleaseDC(hDesk, hSrce);

EncoderParameter qualityParam = new EncoderParameter(System.Drawing.Imaging.Encoder.Quality, (long)q);
EncoderParameters encoderParams = new EncoderParameters(1);
encoderParams.Param[0] = qualityParam;
ImageCodecInfo jpegCodec = GetEncoderInfo("image/jpeg");
MemoryStream jpg = new MemoryStream();
//  bmp.Save(@"d:\temp\test.jpg", jpegCodec, encoderParams);
bmp.Save(  jpg, jpegCodec, encoderParams);
```

Рисунок 4.9 — Захват зображення

Для роботи введені такі класи як:

- додаток (MProcess) – який має у собі назву, код, командну строку, строку параметрів та координаті;

- сценарій (MScenario) – який зберігає назву та перелік кодів додатків, координати та розмір для кожного;

- вікно (MWindow) – зберігає усі властивості вікна.

У такий спосіб серверна частина реалізує усі функціональні вимоги.

4.2 Клієнтська частина

Клієнтська частина реалізована за допомогою технології WPF. Спочатку реалізовано інтерфейс користувача, який зображено на рисунку 4.10

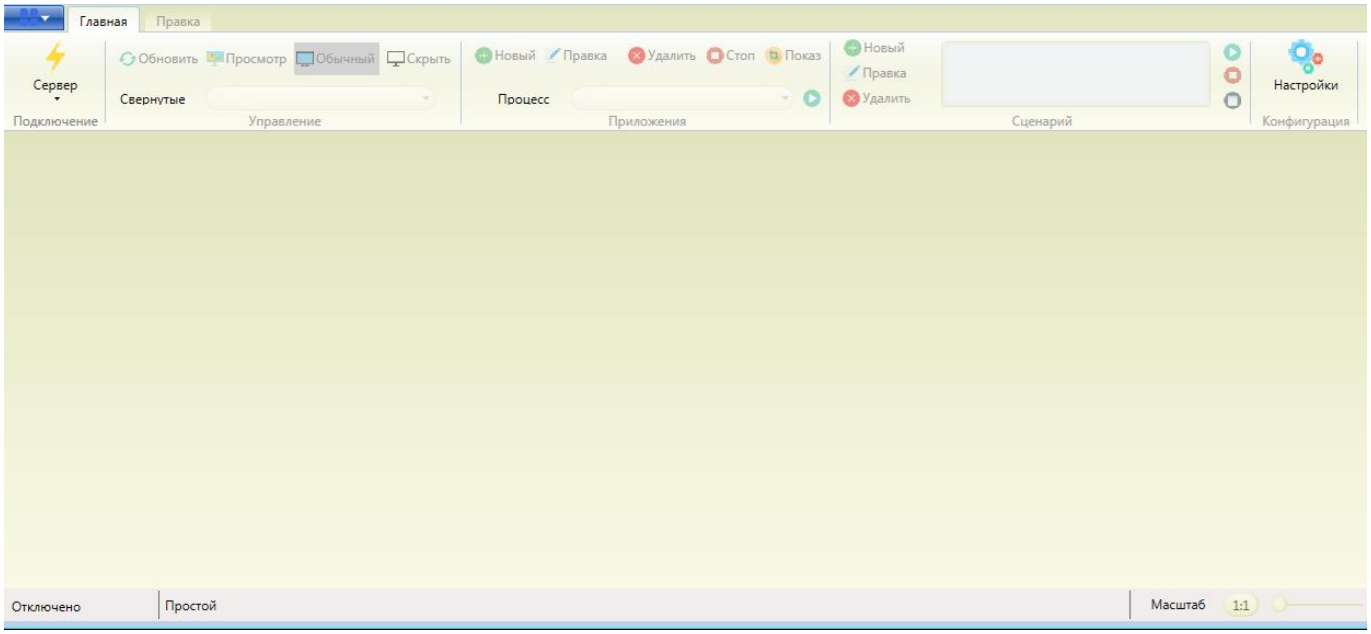


Рисунок 4.10 – Інтерфейс користувача

У додатку можна побачити стрічку (Ribbon) де розміщенні основні функціональні кнопки. Всі вони розділені на п’ять розділів, я яких три недоступні без підключення. Також маємо неактивну вкладку, яка призначена для створення та редагування сценаріїв, та дозволяє розміщувати вікна додатків на потрібних місцях екрану.

Кнопка Connect при натисканні дає можливість підключитися до вже доданих відеостін, як показано на рисунку 4.11, відключитися, чи відкрити вікно з можливістю додавання нових, рисунок 4.12. При додаванні нового підключення є можливість встановити пароль та додати опис підключення.

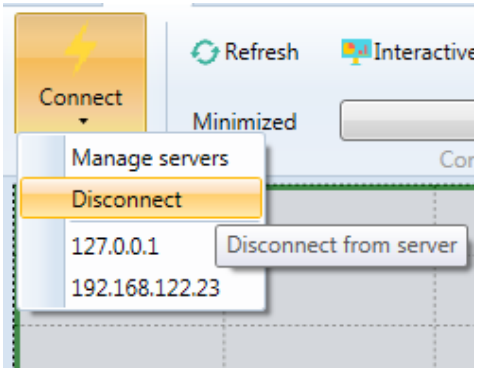


Рисунок 4.11 – Підключення до серверу

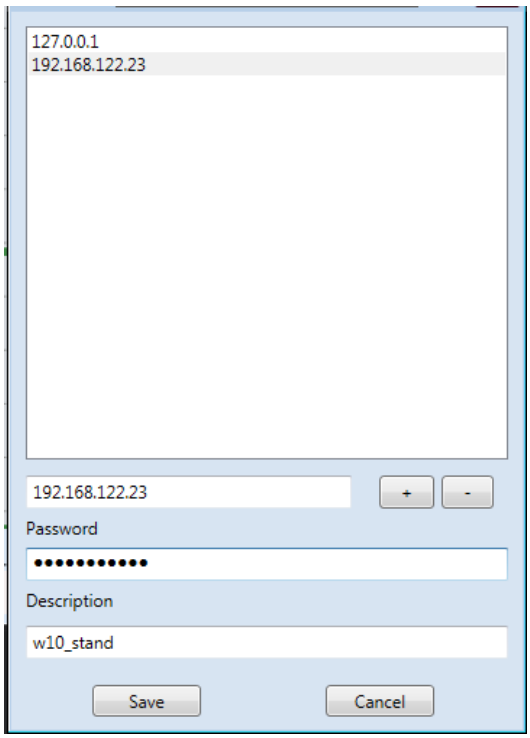


Рисунок 4.12 – Управління підключеннями

Кнопка Options, як зображено на рисунку 4.13, дозволяє налаштувати параметри задані у вимогах, які будуть збережені, та використані у наступному запуску.

У даному вікні є можливість:

- задати частоту оновлення екрану;

- встановити якість та швидкість оновлення зображення у інтерактивному режимі;

- встановити таймаут підключення;

- обрати мову інтерфейсу;

- налаштувати зовнішній вид.

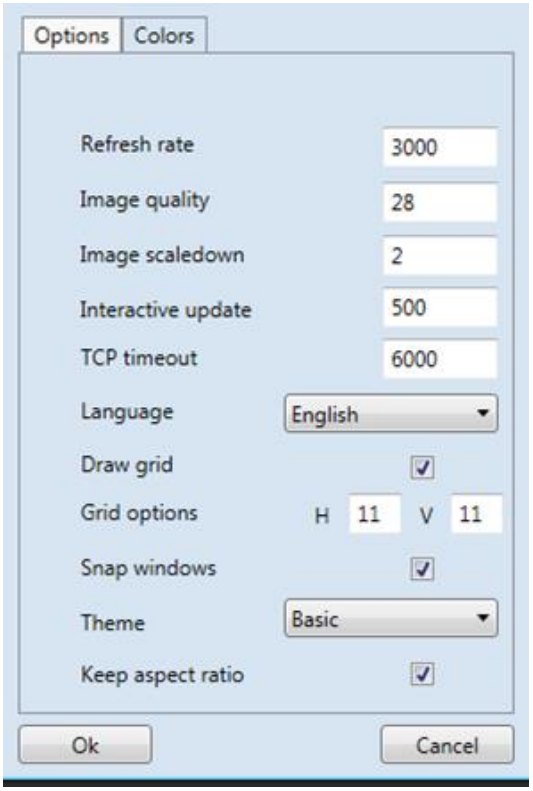


Рисунок 4.13 – Вікно налаштувань

У нижньої частині можна побачити строку статусу та керування масштабом зображення.

Якщо підключитися до серверу, стають доступними усі інші опції. Спочатку розглянемо розділ Control. Там можна оновити статус, кнопка Interctive — переключає схематичний режим до інтерактивного (якщо треба керувати самим додатком, та бачити картинку), Normal повертає у схематичний режим, Hide — приховує усі вікна. Список Minimized відображає усі звернуті вікна, а якщо вибрати вікно з нього, то воно з’явиться на екрані.

Також у робочої області, як можна побачити рисунку 4.14, з’являється схематичне зображення відеостіни (тестовий стенд з чотирма панелями) та можливість керувати вікнами.

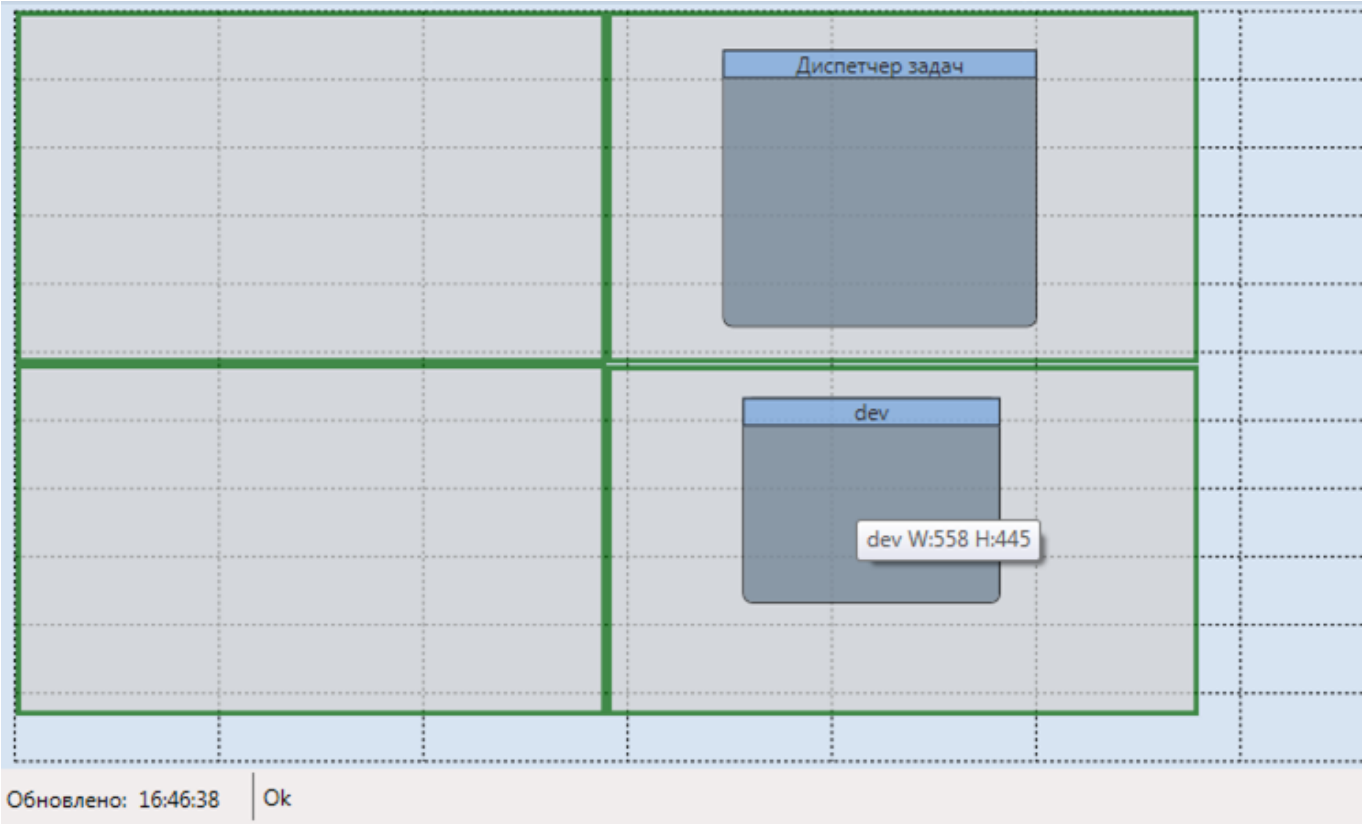


Рисунок 4.14 – Схематичне відображення відеостіни

Якщо спробувати перетягнути на робочу область будь який файл (drag’n’drop), то він буде переданий до серверу та відкритий. Сервер має фільтрацію, згідно контенту файлу, щоб запобігти виконування злоякісного програмного забезпечення. Ця функція потрібна для швидкого відображення документу, чи схеми, яку могли надіслати диспетчеру поштою.

Розділ Applications – дозволяє керувати контентом, який можна відображати на відеостені. Найголовнішим образом це додатки, які можуть бути якими завгодно, наприклад, корпоративне програмне забезпечення, веб контент, відео та ін.

Кнопка New, як можна побачити на рисунку 4.15, дозволяє добавити у систему новий контент. Це може бути звичайний додаток операційної системи, каталог з зображеннями, каталог з відео, або текстовий файл.

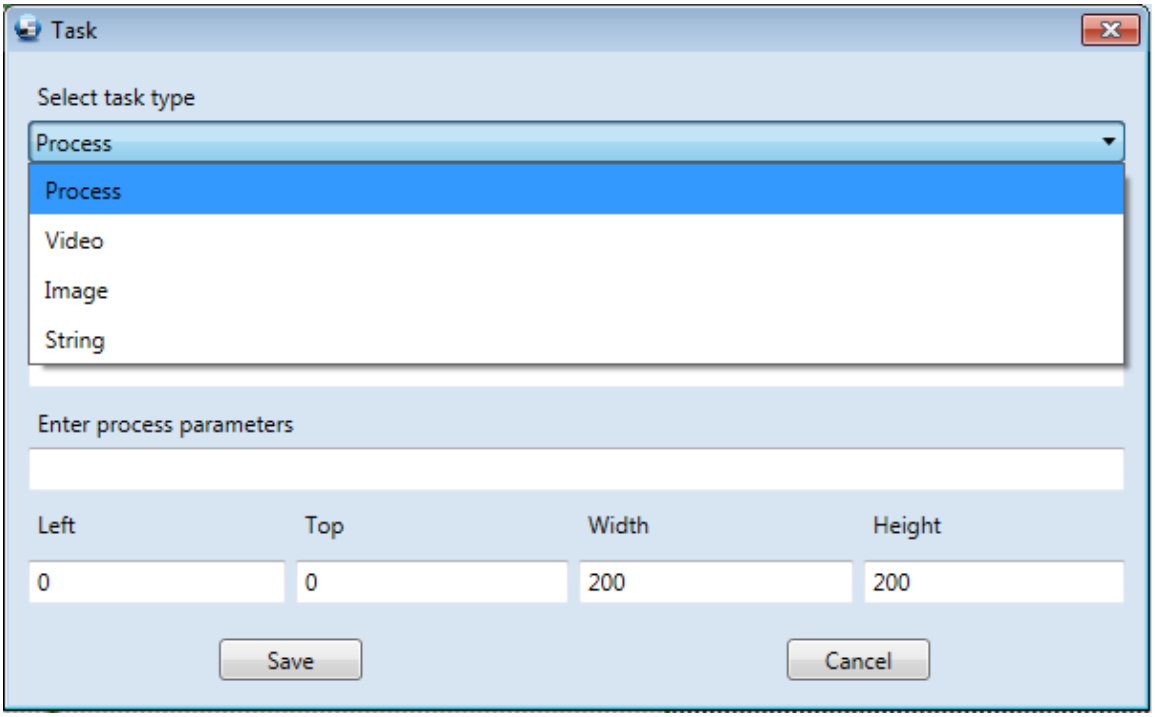


Рисунок 4.15 – Додавання контенту

Тут можливо зазначити основні параметри, та координати виводу за замовченням.

Після додавання, його можливо запустити, редагувати, видалити чи зупинити.

Також у розділі присутня кнопка Crop, яка дозволяє виділити на дисплеї персонального комп'ютера клієнта будь яку область, яка з'явиться на відеостіні. Ця функція дуже важлива, якщо потрібно вивести зображення якогось програмного забезпечення, якого не має на комп'ютері контролера.

Розділ Scenario — потрібен для запуску набору додатків у потрібних містах. Зазвичай на відеостіні у диспетчерських транслюють промисловий процес цілком, але якщо трапилась аварія, потрібно більш детальне зображення аварійного об'єкту. Тому можна зробити сценарії, у яких набір додатків відображає конкретні об'єкти, і натисканням однієї кнопки повністю міняти зображення.

Розділ дозволяє робити нові сценарії, коригувати, видаляти, запускати чи вимикати.

При додаванні нового або коригуванні сценарію, як зображено на рисунку 4.16, додаток перемикається у режим роботи зі сценаріями.

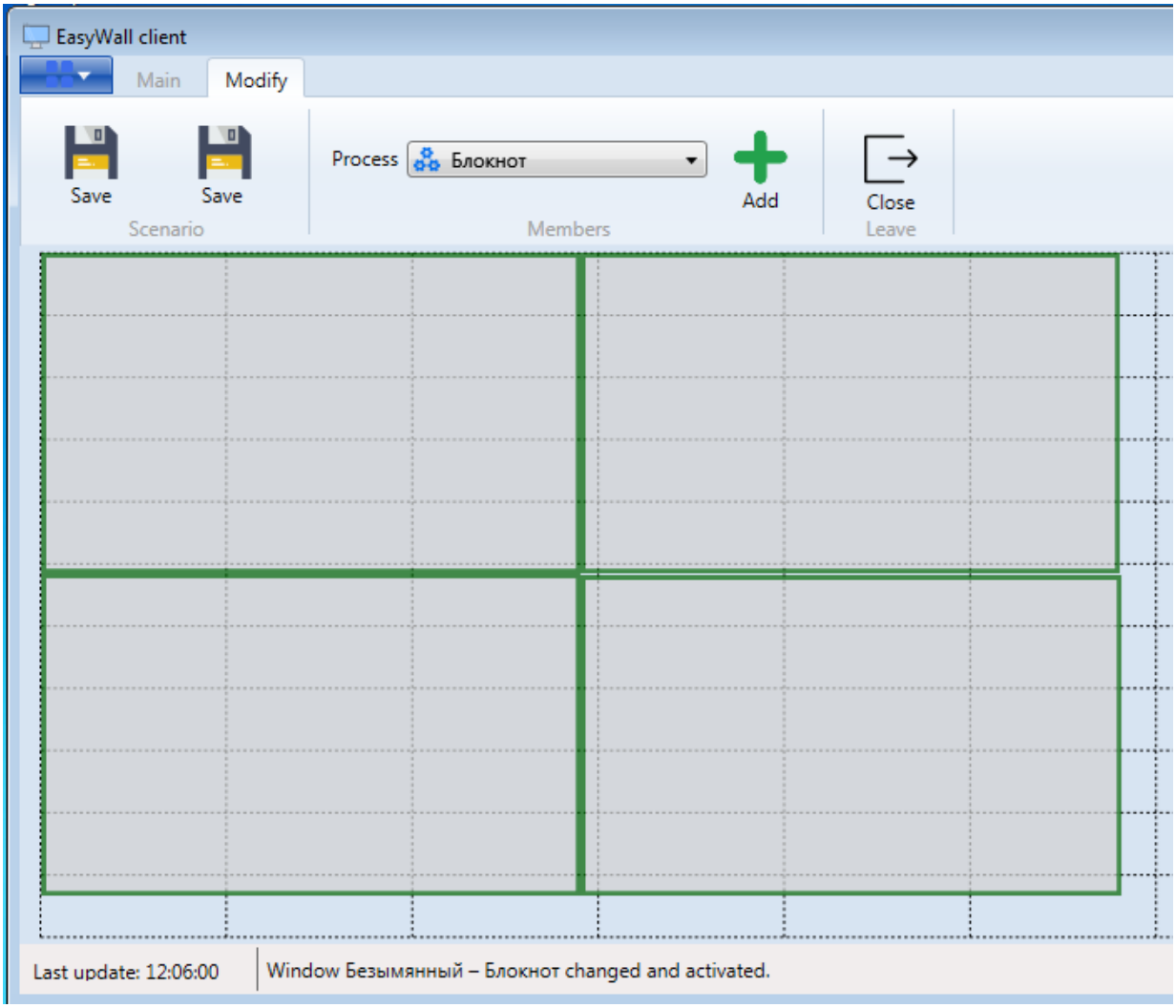


Рисунок 4.16 — Робота зі сценаріями

У цьому режимі можливо вибрати будь який набір доданих раніше додатків, та розмістити їх на потрібному місці відеостіни. Є можливість зберегти скорегований сценарій з новим ім'ям. Сценарій стає доступним усім користувачам, та відразу з'являється у списку. Далі згідно з доданої користувачеві ролі є можливість їм керувати.

Управління вікнами у схематичному режимі працює як зі звичайними, але переміщувати можливо не тільки за заголовок, а за всю площу вікна. При натисканні правої кнопки миші, з’являється контекстне меню, з можливістю мінімізувати, нормалізувати, максимізувати та закрити вікно.

Для реалізації цього функціоналу створено метод sendMessage який дозволяє обмінюватися інформацією із серверною частиною. Метод, як зображено на рисунку

4.17 приймає строку з командою, шифрує та відправляє її на сервер. Детальніше можна побачити у додатку А.

```
private String sendMessage(String cmd)
{
    //  int netTimeout = tcpTimeout;
    string encryptedstring = StringCipher.Encrypt(cmd, CurrentKey);
    encryptedstring += "\r\n\r\n";
    Byte[] data = System.Text.Encoding.ASCII.GetBytes(encryptedstring);
    try
    {
        client = new TcpClient(server, port);
        client.ReceiveTimeout = tcpTimeout;
        client.SendTimeout = tcpTimeout;
        NetworkStream stream = client.GetStream();
        stream.Write(data, 0, data.Length);
        data = new Byte[65536];
        String responseData = String.Empty;
        Int32 bytes;
        while ((bytes = stream.Read(data, 0, data.Length)) > 0)
            responseData += System.Text.Encoding.UTF8.GetString(data, 0, bytes);
        stream.Close();
        client.Close();
        return responseData;
    }
    catch (Exception e)
    {
        return rm.GetString("_e")+": " + e.Message;
    }
}
```

Рисунок 4.17 Обмін даними з сервером

Збереження інформації та взаємодія з вікнами та дисплеями серверу здійснюється за допомогою класів Window та Screen відповідно. Діаграма класу Window зображена на рисунку 4.18. У класах є уся інформація стосовно координат, розміру, назви та доступних методів взаємодії з вікнами та дисплеями серверу. Також реалізоване управління шарами, для цього додано z координату, яка зберігає шар вікна стосовно інших та дозволяє вірно змальовувати їх на зображенні. Детальна схема класів надана у додатку И.

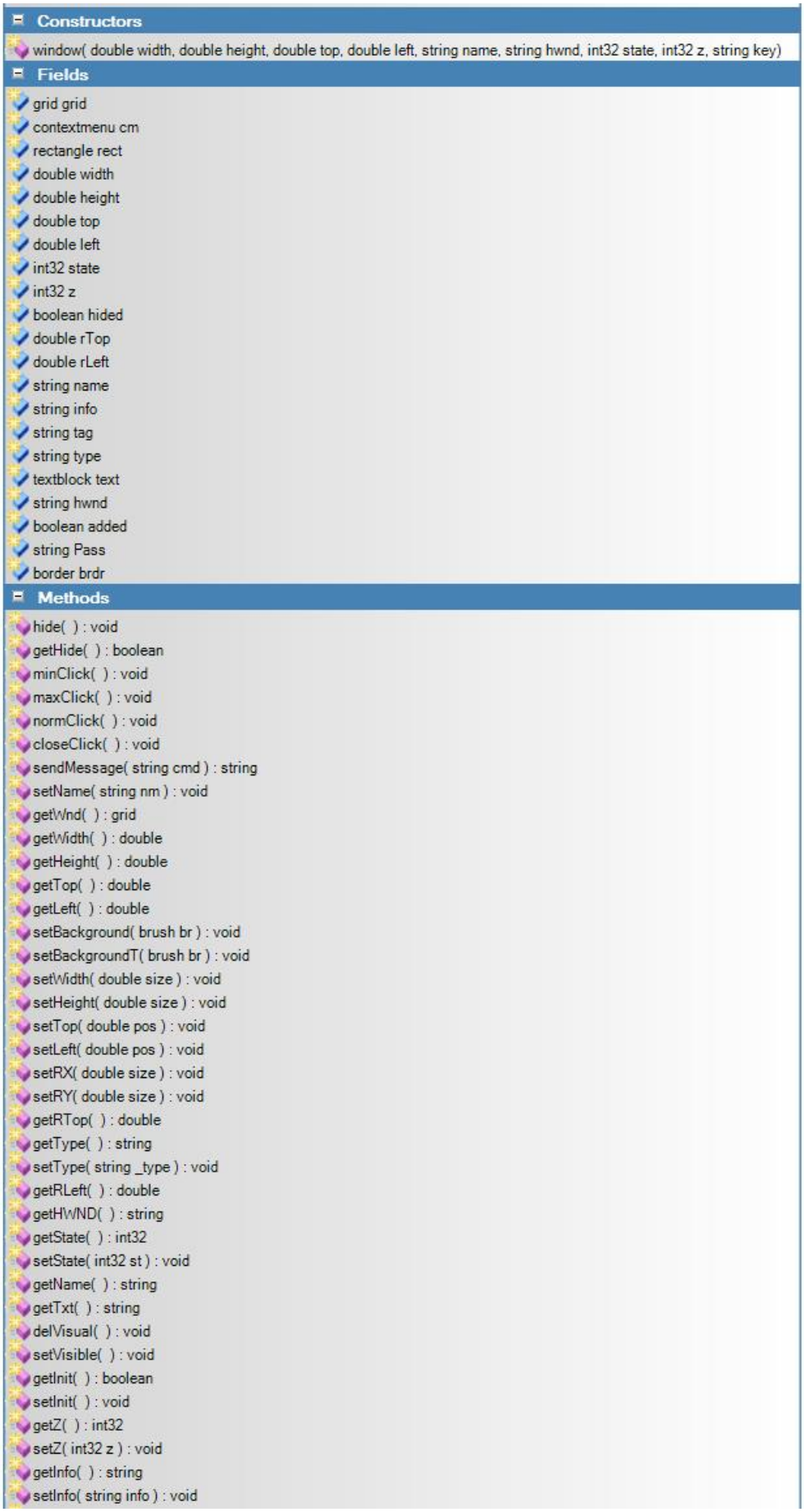


Рисунок 4.18 – Діаграма класу Window

Інформація, як можна побачити на рисунку 4.19, оновлюється згідно з налаштуванням у додатку, 3000 мс за замовчуванням.

```
_tmRefresh = new DispatcherTimer();
_tmRefresh.Interval = TimeSpan.FromSeconds(refreshTimeout / 1000);
_tmRefresh.Tick += refresh;
_tmRefresh.Start();
```

Рисунок 4.19 Автоматичне оновлення

Щоб схематично відобразити вікна та дисплеї серверу, у додатку використовується метод render, який згідно інформації у об’єктах, як зображено на рисунку 4.20, додає їх до канви робочої області. При додаванні вчитуються масштабування та режим роботи. Якщо додаток працює у інтерактивному режими, фон змінюється на прозорий, щоб була можливість побачити робочий стіл серверу.

```
//draw screens
foreach (var screen in screens)
{
    screen.setRX(ratioX);
    screen.setRY(ratioY);
    Canvas.SetLeft(screen.getWnd(), (screen.getX() - minx) / ratioX);
    Canvas.SetTop(screen.getWnd(), (screen.getY() - miny) / ratioY);
    if (pic_mode)
        screen.setBackground(System.Windows.Media.Brushes.Transparent);
    else
        screen.setBackground(new SolidColorBrush(deskColor));
}
```

Рисунок 4.20 Додавання об’єктів

Кожна кнопка чи меню при натисканні, має свій метод, який виконує свою функцію. Якщо потрібен обмін з сервером, формується відповідна команда, яка, за допомогою метода sendMessage відправляється на сервер. Перед відправкою команда шифрується у такий самий спосіб як на сервері з додаванням солі. Відправка зображень, при демонструванні робочого столу або його частини, виконується у окремому потоці. Це дозоляє залишати інтерфейс програми клієнту швидким та своєчасно реагувати на команди користувача. Приклад такої відправки зображено на рисунку 4.21

```
private void runProc_Click(object sender, EventArgs e)
{

    String name = (rbGCProc.SelectedItem as StackPanel).Tag.ToString();
    if (name == null) {
        MessageBox.Show(rm.GetString("_enns"), rm.GetString("_e"), MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }
    Int32 id = ps.FindIndex(delegate (String str) { return (str.IndexOf("<name>" + name + "</name>") > 0); });
    string proc = ps.ElementAt(id);
    string message = "127.0.0.1/RUNNUM=" + proc.Substring(proc.IndexOf("<id>") + 4,
        proc.IndexOf("</id>") - proc.IndexOf("<id>") - 4) + ";\r\n\r\n";
    String responseData = sendMessage(message);
    if (responseData.Contains("Error:"))
    {
        tbStatus.Text = "Run process " + name + " error: " + responseData;
        return;
    }
    tbStatus.Text = name + " "+rm.GetString("_ps");
}
```

Рисунок 4.21 – Приклад процесу запуску

Додатково у клієнтському додатку реалізовані вікна для збереження, додавання процесів та сценаріїв. Приклад діаграми класу Servers зображено на рисунку 4.22.

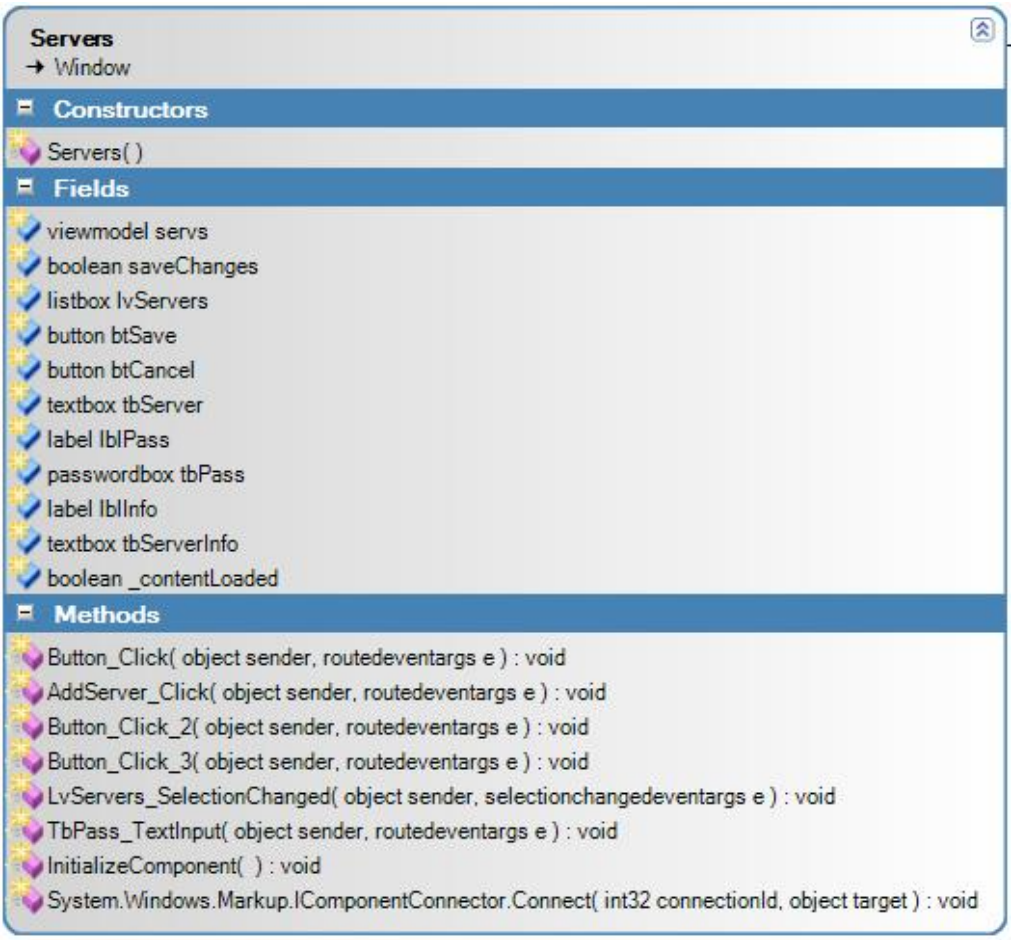


Рисунок 4.22 – Діаграма класу вікна Servers

Частина налаштувань клієнту зберігається на сервері, але такі параметри як мова, перелік серверів та зовнішній вид розташовані у конфігураційному файлі поруч з клієнтським додатком. Для його зчитування використовується метод readConfig. Цей метод, як зображено на рисунку 4.23, намагається зчитати файл у профілі користувача.

```
string line;
StreamReader file;
var pathWithEnv = @"%USERPROFILE%\config.cfg";
var filePath = Environment.ExpandEnvironmentVariables(pathWithEnv);
FileInfo fl = new FileInfo(filePath);
if (fl.Exists)
    file = new System.IO.StreamReader(filePath);
else
{
    firstRun = true;
    saveConfig();
    return;
}
```

Рисунок 4.23 – Зчитування параметрів клієнту

При зчитуванні відбувається перевірка на існування файлу, якщо його не знайдено, то вважається що додаток запущено уперше та запускається формування конфігураційного файлу за замовчуванням.

Якщо файл знайдено, далі робиться його розбір згідно ключових слів, та зчитуються налаштування. Приклад зчитування параметру з переліком збережених серверів надано на рисунку 4.24.

```
if (line.Contains("<server>"))
{
    RibbonMenuItem rbItem = new RibbonMenuItem();
    rbItem.Header = line.Substring(line.IndexOf("<name>") + 6,
        line.IndexOf("</name>") - line.IndexOf("<name>") - 6);
    rbItem.ToolTip = line.Substring(line.IndexOf("<info>") + 6,
        line.IndexOf("</info>") - line.IndexOf("<info>") - 6);
    if (rbItem.ToolTip.ToString().Length <= 0) rbItem.ToolTip = null;
    rbItem.Tag = StringCipher.Decrypt(line.Substring(line.IndexOf("<key>") + 5,
        line.IndexOf("</key>") - line.IndexOf("<key>") - 5), "ABBACAAC");
    rbItem.Click += rbConnect_Click;
    rbConnectMenu.Items.Add(rbItem);
}
```

Рисунок 4.24 – Розбір параметрів сервера

Для додавання контенту реалізовано клас ProcWindow. Клас дозволяє одночасно додавати новий контент чи коригувати існуючий. Для цього у класі

реалізовано два конструктори. Перший не приймає жодного параметру, та використовується для додавання нового додатку, другий, як можна побачити на рисунку 4.25, приймає усі параметри та дозволяє їх зкорегувати.

```
public ProcWindow(String name, String exe, String par,
    String left, String top, String width, String height, bool upd)
{
    InitializeComponent();
    lblPN.Content = App.res.GetString("_lblPN");
    lblPP.Content = App.res.GetString("_lblPP");
    lblL.Content = App.res.GetString("_lblL");
    lblT.Content = App.res.GetString("_lblT");
    lblH.Content = App.res.GetString("_lblHT");
    lblW.Content = App.res.GetString("_lblW");
    lblPC.Content = App.res.GetString("_lblPC");
    btCancel.Content = App.res.GetString("_cnl");
    btSave.Content = App.res.GetString("_btSave");
    this.upd = upd;
    tbName.Text = name;
    tbExe.Text = exe;
    tbPar.Text = par;
    tbLeft.Text = left;
    tbTop.Text = top;
    tbWidth.Text = width;
    tbHeight.Text = height;
}
```

Рисунок 4.24 – Редагування параметрів додатку

Збереження коригувань або додавання нового додатку, при натисканні кнопки у даному вікні, як можна побачити на рисунку 4.25, формує команду для подальшого відсилання на сервер. Інформація о нових додатка оновлюється для усіх користувачів одночасно.

```
TcpClient client = new TcpClient(MainWindow.server, MainWindow.port);
string message = "127.0.0.1/UpdPlot=<n>1</n><state>off</state><name>" +
    tbName.Text + "</name><exe>" + tbExe.Text + "</exe><param>" + tbPar.Text
    + "</param><x>" + tbLeft.Text + "</x><y>" + tbTop.Text + "</y><w>"
    + tbWidth.Text + "</w><h>" + tbHeight.Text + "</h>" + ";\r\n\r\n";
string encryptedstring = EWClient.StringCipher.Encrypt(message, pass);
encryptedstring += "\r\n\r\n";
```

Рисунок 4.25 – Формування команди нового додатку

Для більш інформативного інтерфейсу та можливості користувачеві бачити стан додатку та його стану, у клієнтський додаток додано строку статусу. На якої, як можна

побачити на рисунку 4.26, додається останній час оновлення, та результат виконання останньої команди.

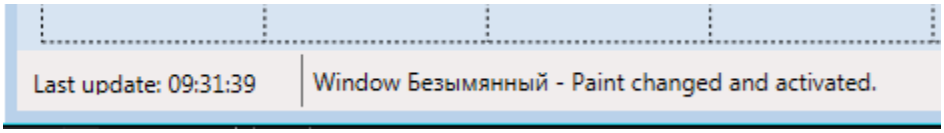


Рисунок 4.26 – Рядок статусу

Рядок статусу має у собі статус підключення чи час оновлення, результат останньої команди та управління збільшенням масштабу. Приклад реалізації на язику XAML можна побачити на рисунку 4.27.

```
</StatusBar.ItemsPanel>
<StatusBarItem>
    <Label x:Name="lbConnect" Width="135" Content="Disconnected" Foreground="Black"/>
</StatusBarItem>
<Separator Grid.Column="1" />
<StatusBarItem Grid.Column="2">
    <TextBlock x:Name="tbStatus" Text="Idle" ToolTip="{Binding Text, ElementName=tbStatus}"/>
</StatusBarItem>
<StatusBarItem Grid.Column="3">
</StatusBarItem>
<Separator Grid.Column="4" />
<StatusBarItem Grid.Column="5">
    <Label Name="lblZoom" Width="77" Content="zoom" Foreground="Black" Margin="0,-2,0,0"
           HorizontalContentAlignment="Center"/>
</StatusBarItem>
<StatusBarItem Grid.Column="6">
    <Button Name="btUnZoom" Width="32" Content="1:1" Click="btUnZoom_Click" ToolTip="Show all displays"/>
</StatusBarItem>
<StatusBarItem Grid.Column="7">
    <Slider x:Name="uiScaleSlider" Width="95" FlowDirection="LeftToRight"
            Minimum="0.7" Maximum="5" Value="0.98" ToolTip="Scale windows"/>
</StatusBarItem>
```

Рисунок 4.27 – Реалізація рядку статусу

Важливою частиною додатку є реалізація масштабування зображення. Розмір та роздільна здатність відеостін дуже велика, тому часто елементи у додатку керування дуже замалі. Завдяки функції збільшування, чи зменшування зображення з'являється можливість чітко керувати вікнами, та у інтерактивному режимі натискати потрібні кнопку у додатках. Функція керування збільшенням зображена на рисунку 4.28.

Рисунок 4.28 – Керування збільшенням

При зміщуванні слайдеру змінюється глобальний параметр Scale, який використовується при формуванні зображення.

Для зображення на сервері робочого столу клієнту використовується функція Crop. При натисканні на кнопку з'являється можливість вибору довільної області екрану, яка буде транслюватися на сервер. При цьому екран стає трохи темішим та курсор змінює форму. Приклад виділення зображено на малюнку 4.29.

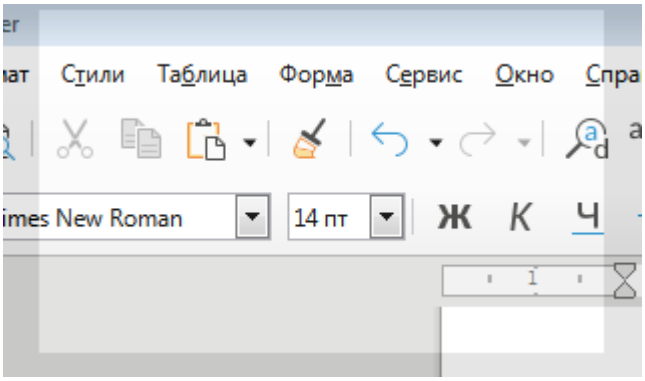


Рисунок 4.29 – Виділення частини екрану для транслювання

Реалізація такого підходу потребує зробити копію екрану, та у новому вікні, яке не має рамок та розміром з робочий стіл клієнта, намалювати цю копію. Дали достатньо при натисканні миші і перетягуванні малювати потрібну область зображення, яка буде мати не затемнений колір. Після цього додаток завдяки координатам області, копіює вміст екрану за координатами, перекодує картинку у графічний формат, згідно з налаштуваннями у клієнтському додатку, та передає їх у окремому потоці на сервер, який зображує картинку у окремому вікні та дозволяє їй керувати. Програмна реалізація зображена на рисунку 4.30.

```
public void SaveScreen(object sender, EventArgs e)
{
    int ix, iy, iw, ih;
    ix = Convert.ToInt32(left + x);
    iy = Convert.ToInt32(top + y);
    iw = Convert.ToInt32(width);
    ih = Convert.ToInt32(height);
    try
    {

        IntPtr hDesk = NativeMethods.GetDesktopWindow();
        IntPtr hSrce = NativeMethods.GetWindowDC(hDesk);
        IntPtr hDest = NativeMethods.CreateCompatibleDC(hSrce);
        IntPtr hBmp = NativeMethods.CreateCompatibleBitmap(hSrce, iw, ih);
        IntPtr hOldBmp = NativeMethods.SelectObject(hDest, hBmp);
        bool b = NativeMethods.BitBlt(hDest, 0, 0, iw, ih, hSrce, ix, iy,
            CopyPixelOperation.SourceCopy | CopyPixelOperation.CaptureBlt);
        Bitmap myImage = Bitmap.FromHbitmap(hBmp);
        NativeMethods.SelectObject(hDest, hOldBmp);
        NativeMethods.DeleteObject(hBmp);
        NativeMethods.DeleteDC(hDest);
        NativeMethods.ReleaseDC(hDesk, hSrce);

        EncoderParameter qualityParam = new EncoderParameter(
            System.Drawing.Imaging.Encoder.Quality, (long)q); //20 - quality
        EncoderParameters encoderParams = new EncoderParameters(1);
        encoderParams.Param[0] = qualityParam;
        ImageCodecInfo jpegCodec = GetEncoderInfo("image/jpeg");
        MemoryStream jpg = new MemoryStream();
        //  bmp.Save(@"d:\temp\test.jpg", jpegCodec, encoderParams);
        myImage.Save( /*@"d:\temp\test.jpg"*/ jpg, jpegCodec, encoderParams);
        jpg.Seek(0, SeekOrigin.Begin);
        sendBmp(jpg);
    }
    catch { }
}
```

Рисунок 4.30 – Обробка виділеної частини екрану

Для кодування використовується формат jpg, що не дозволяє плавно виводити картинку на сервері, але дуже економить ресурси клієнта, так як кодування відео у реальному часі вимагає досить потужного процесору та об'єму оперативної пам'яті, або наявності у системі відеоадаптеру, який вміє апаратно кодувати відеозображення. Але зазвичай ця функція використовується для демонстрування додатків відсутніх на відеостені. Бо деякі з них можуть маті погану сумісність з операційної системою відеосерверу, чи ліцензувати їх встановлювання. Тому швидке оновлення такої інформації не потрібно, достатньо вивести картинку на сервері. Якщо потрібно демонструвати на відеостені відеофайли, то для цього реалізована функція

Drag’n’Drop, яка дозволяє перетягнути відеофайл на додаток, передати його на сервер та запустити там у відеоплеєрі.

Функція Drag’n’Drop при переміщенні файлу на додаток перевіряє його тип та надає можливість перетискувати тільки графічні, відео файли та документи. Це не дасть можливість перетягнути додаток, який може бути запущеним на сервері, та привести до непередбаченої поведінки. Також це правильно з точки зору безпеки, адже серверна частина має привілейовані права і зловмисник таким чином може спробувати обійти рольові обмеження.

Після отримання та обробки інформації з перетягнутого файлу, починається формування необхідної команди до серверу, яка створює на ньому окремій потік для прийому файлу, та починається передача, реалізація якої зображена на малюнку 4.31.

```
static void sendFile(String fname)
{
    TcpClient client = new TcpClient(server, port);
    NetworkStream stream = client.GetStream();
    string message = "127.0.0.1/SFn=" + fname+";\r\n\r\n";
    string encryptedstring = StringCipher.Encrypt(message, CurrentKey);
    encryptedstring += "\r\n\r\n";
    Byte[] data = System.Text.Encoding.UTF8.GetBytes(encryptedstring);
    stream.Write(data, 0, data.Length);
    FileStream fstream = new FileStream(fname, FileMode.Open, FileAccess.Read);
    byte[] block = new byte[2048];
    int bytesRead;
    while ((bytesRead=fstream.Read(block, 0, block.Length)) > 0)
    {
        stream.Write(block, 0, bytesRead);
    }
    stream.Close();
    client.Close();
}
```

Рисунок 3.31 – Передача файлу до серверу

Після передачі файлу, сервер відкриває його згідно зареєстрованих у системі типів файлу у потрібному додатку, що дозволяє керувати їм як іншими вікнами. Головне встановити на сервер додатки для відкриття графічних, відеофайлів чи документів. Наприклад це можуть бути Acrobat Reader та MS Office для відкриття документів, InfranView для графічних файлів та MediaPlayer Classic для демонстрування відео.

Для швидкого розгортання системи додана можливість автоматичного встановлювання клієнтської та серверної частини на комп'ютер. За допомогою Visual Studio Installer Project створені пакети автоматичного розгортання. Вони дозволяють автоматично створити всі необхідні каталоги на жорсткому диску, прописати потрібні параметри до реєстру операційної системи та у процесі встановлення надати можливість користувачеві змінити деякі параметри.

Для розгортання серверу перевіряється попереднє встановлювання, додається можливість обрати необхідний каталог та увімкнути автоматичний запуск серверу при старті операційної системи. Також при встановлюванні до каталогу копіюються усі необхідні бібліотеки, створюються необхідні пункти у системному меню та за бажанням користувача додається посилання на робочому столі. Приклад проєкту зображено на рисунку 3.32.

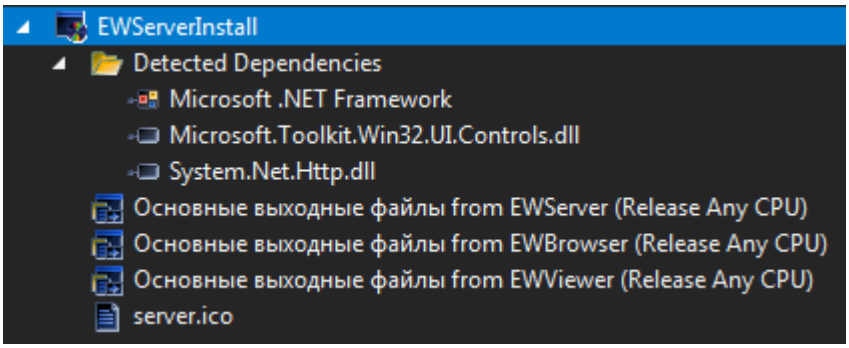


Рисунок 3.32 – Проєкт розгортання серверної частини

Для автоматичного запуску використовується параметр реєстру HCU\Software\Microsoft\Windows\CurrentVersion\Run, у якому створено текстовий параметр зі шляхом до додатку. Це дозволяє при налаштуванні автоматичного входу до операційної системи запускати сервер з правами користувача. Зазвичай використовують права адміністратора, щоб мати можливість використовувати всі функції. Але це не обов'язково, у критичних системах є можливість працювати з правами користувача, єдине обмеження це неможливість керувати вікнами, які виконуються із привілеями адміністратора, або запускати додатки які ці привілеї потребують.

4.3 Допоміжні додатки

Для транслювання веб контенту потрібен браузер, але сучасні браузери мають забагато функціоналу який не потрібен на відеостені, наприклад, кнопки назад, додому, адресна поле, вони лише займають місце. До того ж, сучасними браузерами важко керувати, так як вони роблять окремій процес під кожну вкладку.

Тому у рамках проєкту розроблено простий веб браузер, якій вміє використовувати движки Microsoft Edge чи Internet Explorer.

Браузер вміє розуміти такі параметри як url та координати вікна, що дозволяє їм гнучко керувати.

Як можна побачити на рисунку 4.33, на старті браузер перевіряє версію операційної системи, та залежно від неї обирає компонент для відображення веб сторінок.

```
string releaseId = Registry.GetValue
    (@"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion", "ReleaseId", "").ToString();
if (int.Parse(releaseId)>=1803)
{
    var wvc = new WebView();
    grMain.Children.Add(wvc);
    if (args.Length > 1)
    {
        Uri url = new Uri(args[1]);
        wvc.Navigate(url);
    }
    else wvc.Navigate(new Uri("https://google.com"));
}
else
{
    var wvb = new WebBrowser();
    grMain.Children.Add(wvb);
    if (args.Length > 1)
    {
        Uri url = new Uri(args[1]);
        wvb.Navigate(url);
        //          wbMain.Source = url;
    }
    else wvb.Navigate(new Uri("https://google.com"));
}
```

Рисунок 4.33 – Перевірка версії операційної системи

Компонент WebView використовується у нових версіях операційних систем, та дозволяє використовувати для рендеру сторінок можливості Edge, а компонент WebBrowser використовує можливості Internet Explorer. Ця перевірка надає

можливість використовувати браузер як на операційній системі Windows 7, так і на Windows 10.

Як можна побачити з малюнку 4.34, далі йде обробка параметрів, які визначають координати, розмір та стиль відображення вікна. URL передається у якості першого параметру.

Зміна стилю відображення вікна дозволяє прибрати заголовок вікна, та відображати тільки контент веб сторінки.

У разі помилки, чи невірно переданих параметрів, браузер запускається згідно останніх переданих координат.

```
if (args.Length > 2)  wMain.Left = Convert.ToInt32(args[2]);
if (args.Length > 3) wMain.Top = Convert.ToInt32(args[3]);
if (args.Length > 4) wMain.Width = Convert.ToInt32(args[4]);
if (args.Length > 5) wMain.Height = Convert.ToInt32(args[5]);
if (args.Length > 6)
    {
        if( Convert.ToInt32(args[6])==1) wMain.WindowStyle=WindowStyle.None;
    }
}
catch (Exception e) {
```

Рисунок 4.34 – Обробка аргументів

Усі обробки параметрів командної строки огорнуті функціоналом try() catch(), якій у разі спроби передачі строки замість координат, обробить помилку, та виведе на екран потрібне повідомлення про помилку.

На рисунку 4.35 зображено приклад запуску браузеру з наданням координат та без формування рамок вікна, що дозволяє більш щільно розмістити контент, та прибрати границі вікна.

У разі потреби у інтерактивному режимі є можливість повноцінно використовувати браузер. Наприклад, при використанні браузера для демонстрування логістичних потреб, буде можливість керувати картою або перемикати режими.

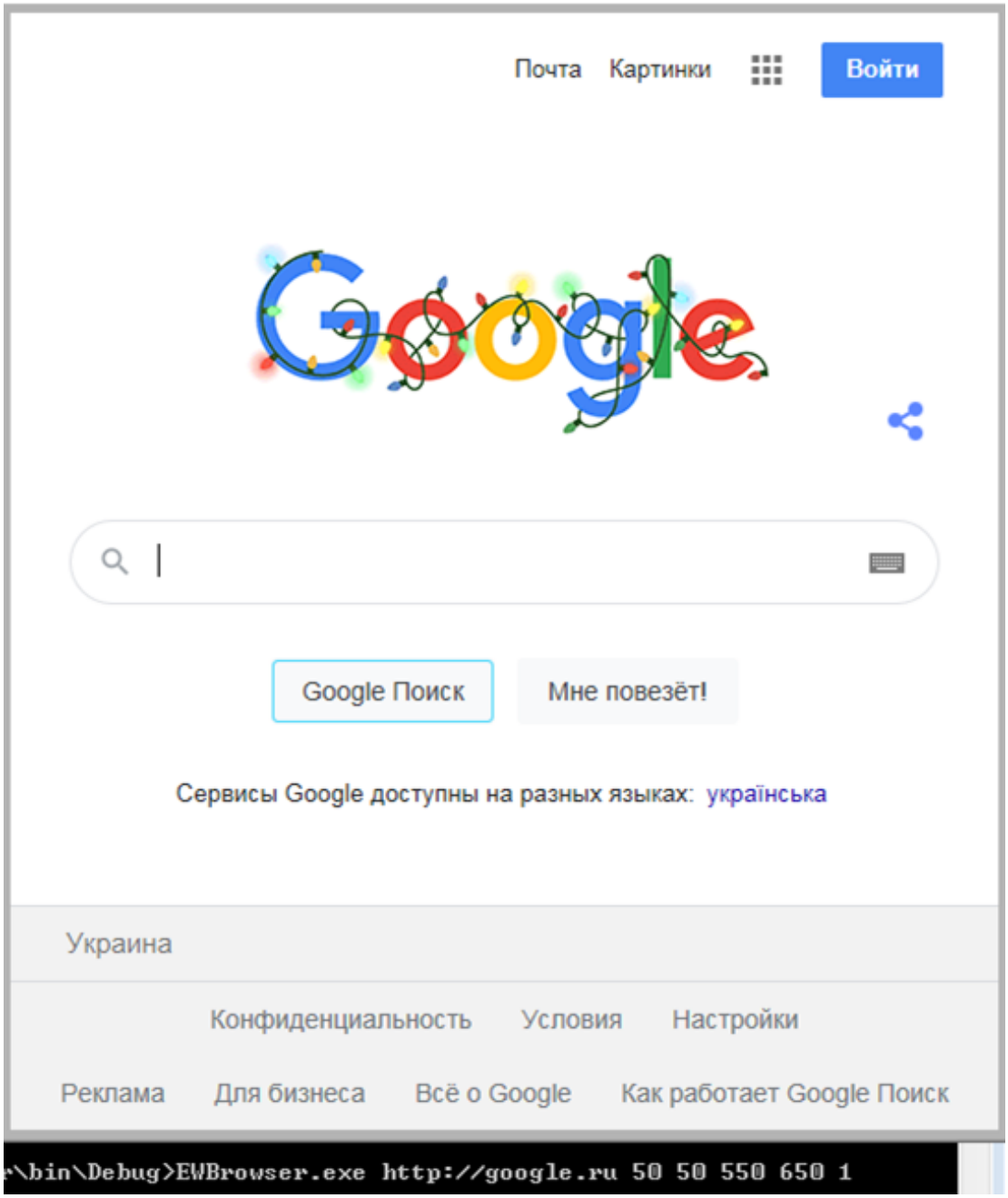


Рисунок 4.35 – Запуск браузеру без заголовку вікна

Також для транслювання відео чи графічної інформації розроблено додаток для перегляду. Він по аналогії з браузером має можливість обробляти параметри з координатами.

Додаток має можливість відображати слайд шоу, відеоконтент та малювати рухомий рядок, який може виводити інформацію з текстового файлу, або з інтернету за допомогою RSS.

На рисунку 4.36 зображена обробка параметрів командної строки, що може приймати до одинадцяти аргументів.

```
if (args.Length > 1) url = args[1];
if (args.Length > 2) type = Convert.ToInt32(args[2]);
if (args.Length > 3) timeout = Convert.ToInt32(args[3]);
if (args.Length > 4) this.Left = Convert.ToInt32(args[4]);
if (args.Length > 5) this.Top = Convert.ToInt32(args[5]);
if (args.Length > 6) this.Width = Convert.ToInt32(args[6]);
if (args.Length > 7) this.Height = Convert.ToInt32(args[7]);
if (args.Length > 8) fontSize = Convert.ToInt32(args[8]);
if (args.Length > 9) spaceSize = Convert.ToInt32(args[9]);
if (args.Length > 10) fontColor = (Color)ColorConverter.ConvertFromString(args[10]);
if (args.Length > 11) backColor = (Color)ColorConverter.ConvertFromString(args[11]);
```

Рисунок 4.36 – Обробка параметрів додатку перегляду

Є можливість визначити URL, тип контенту, паузу, координати, розмір шрифту, відстань, колір шрифту та фону.

Як можна побачити на малюнку 4.37, для обробки RSS контенту використовується XMLReader, який обробляє отриману з сайту інформацію, та перетворює її на строку. Що дає можливість транслювати на відеостені новини чи іншу інформацію, яка може динамічна змінюватися.

```
if (url.Contains("http://") || url.Contains("https://"))
{
    XmlReader reader = XmlReader.Create(url);
    SyndicationFeed feed = SyndicationFeed.Load(reader);
    reader.Close();

    foreach (SyndicationItem item in feed.Items)
    {
        subject = item.Title.Text;
        summary = item.Summary.Text;
        filtered = System.Text.RegularExpressions.Regex.Replace(subject + " - " + summary, "<.*?>", "");
        filtered = filtered.Replace(" ", " ");
        filtered = filtered.Replace("\n", "  ");
        str += filtered;
    }
}
```

Рисунок 4.37 – Обробка RSS

4.4 База даних

Для збереження даних у системі обрана база даних SQLite. Оскільки система не передбачає великого об’єму даних, та висока швидкість обміну не потрібна, цієї бази даних буда достатньо [20].

Функціювання системи, як можна побачити на рисунку 4.38, забезпечує тринадцять таблиць які мають між собою зв’язки.

Рисунок 4.38 – Перелік таблиць у базі даних

За інформацією таблиці мають наступне призначення:

- «config» зберігає налаштування клієнту кожного з користувачів. Зчитується при підключенні, та загружає відповідні налаштування – якість зображення, швидкість оновлення та інше;

- «lastState» зберігає останні запущені на сервері процеси та сценарії. Використовується при запуску серверної частини для відновлення останнього стану після перезавантаження комп’ютеру;

- «roles» зберігає налаштовані на сервері ролі доступу, що дозволяє розподіляти права між користувачами;

- «serverConfig» зберігає налаштування серверу такі як шифрування, автоматичний перезапуск, малювання іконки в системному треї та ін;

- «slides» зберігає перелік доданого до системи графічних файлів та параметри його запуску;

- «strings» зберігає перелік доданого до системи інформаційного контенту та параметри його запуску;

- «videos» зберігає перелік доданого до системи відео контенту та параметри його запуску;

- «users» зберігає дані безпеки користувачів, роль у системі, логін та пароль;

- «userDetails» зберігає детальну інформацію по користувачам;

- «logs» зберігає журнал усіх команд, які були передані до серверу;

- «program» зберігає дані про усі додатки додані до системи, та їх параметри;

- «scenarios» зберігає код сценарію, та його опис;

- «scenarioProgs» зберігає перелік посилань на додатки та їх параметри у сценарії.

Спрощену схему БД зображено на рисунку 4.39, детальна надана у додатку Е.

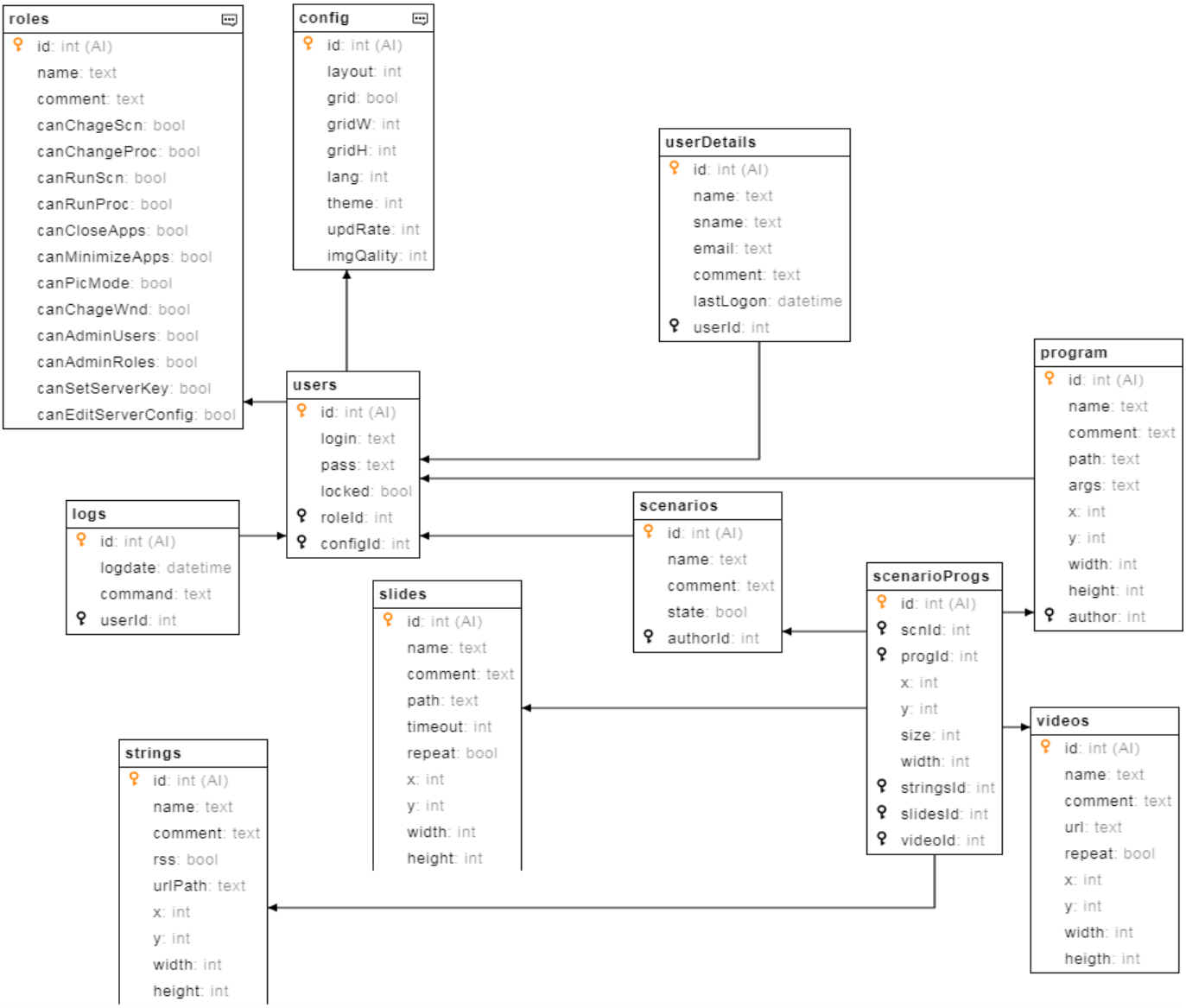


Рисунок 4.39 – Спрощена схема бази даних

4.5 Системні вимоги

Для визначення апаратних вимог можна взяти вимоги до операційних систем та добавити у них потреби нашого програмного забезпечення.

Запустимо обидва на системі з одним Full HD монітором, зробимо підключення, та спробуємо запустити інтерактивний режим, з низькою та високою якістю картинки. Виконання додатків прив'яжемо до одного ядра процесору. Запустимо диспетчер задач, та поглянемо на використання ресурсів.

Для низької якості картинки це буде 59 Мб, 2% CPU для клієнта та 27 Мб, 1% CPU для серверу, як зображено на рисунку 4.40.

Имя образа	Пользо...	ЦП	Память (...
EWClient.exe	stim	02	58 892 КБ
EWServer.exe	stim	01	27 144 КБ

Рисунок 4.40 – Використання ресурсів з низькою якістю

Для високої якості картинки це буде 64 Мб, 2% CPU для клієнта та 46 Мб, 1% CPU для серверу, як зображено на рисунку 4.41.

Имя образа	Пользо...	ЦП	Память (...
EWClient.exe	stim	02	64 288 КБ
EWServer.exe	stim	01	46 456 КБ

Рисунок 4.41 – Використання ресурсів з високою якістю

Тепер добавимо до стенду ще один монітор, та повторимо тест. Цього разу виходить що система потребує трохи більше ресурсів.

Для низької якості картинки це буде 65 Мб, 2% CPU для клієнта та 35 Мб, 1% CPU для серверу. Для високої якості картинки це буде 71 Мб, 3% CPU для клієнта та 52 Мб, 2% CPU для серверу.

Далі визначаємо скільки місця займають обидва додатка на жорсткому диску. Для клієнта це буде 4 Мб, для серверу 5 Мб. Для серверної частини також враховуємо необхідність вільного місця під базу даних.

Для того щоб визначити потреби для процесору, візьмемо частоту тестового стенду та підрахуємо процент використання від робочої частоти. Додамо до неї 20% (середнє підвищення швидкості на ядро, за останні 5 поколінь процесорів) та мінімальні потреби операційної системи.

Отже, потреби першого експерименту дадуть мінімальні та бажані апаратні вимоги до проєкту. Далі підраховуємо різницю між першим та другим експериментом та маємо потреби на кожний додатковий монітор.

Підсумкові апаратні вимоги надані у таблиці 4.1. Опираючись на них можливо розрахувати потреби під відеостіну будь якого розміру. Також потрібно враховувати потреби додатків, які будуть виконуватися на сервері.

Таблиця 4.1 – Апаратні вимоги

Клієнтська частина	Мінімальні вимоги	Бажані вимоги
Процесор	1.2 GHz	1.3 GHz
Оперативна пам’ять	60 Mb free RAM + 6 Mb per monitor	65 Mb free RAM + 7 Mb per monitor
Жорсткий диск	4 Mb free HDD space	4 Mb free HDD space
Серверна частина		
Процесор	1.1 GHz	1.2 GHz
Оперативна пам’ять	30 Mb free RAM + 5 Mb per monitor	46 Mb free RAM + 6 Mb per monitor

Жорсткий диск	10 Mb free HDD space	10 Mb free HDD space

Висновки до розділу 4

У даному розділі надана реалізація програмної частини системи. Показані можливості багатопотокового серверу, який за допомогою WinAPI виконує усі вимоги до системи. Приведена реалізація клієнтської та серверної частини системи, та зазначені її можливості. Наведені приклади та надано опис реалізації кожної функції системи.

Зазначена реалізація та обґрунтована потреба у допоміжних додатках, які дозволяють демонструвати веб-сторінки, інформативні рядки, відео та графічні файли.

У кінці розділу, експериментальним шляхом визначені апаратні вимоги до системи, з можливістю розрахунків при масштабуванні.

Я бачимо за допомогою цього програмного засобу можливо збудувати відеостіну майже під будь які потреби, та працювати вона буде на любому сумісному з Microsoft Windows обладнання, включаючи обладнання споживчого сегменту.

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї

5.1.1 Зміст ідеї

Спочатку зробимо опис ідеї стартап проєкту. Для цього заповнимо таблицю 5.1 з ідеями, напрямками застосування та вигодами для користувачам.

Таблиця 5.1 – Опис ідеї стартап проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	Покращення вигляду приміщень	Покращення зовнішнього стану, розміщення додаткової інформації з планом приміщень та сервісів.
	Відображення корпоративної інформації	Зображення всіх технологічних процесів в одному місці, швидке реагування на аварії, зменшення простоїв
	Відображення рекламних оголошень	Можливість показувати динамічний контент та таргетовану рекламу

Основною ідеєю проєкту є створення системи, яка працює з апаратними компонентами споживчого сегменту, та не має прив'язки до конкретного вендору. Це значно здешевіє систему, та робить її конкурентоспроможною. Для побудови відеостін зі статичним контентом, достатньо навіть бюджетного персонального комп’ютера та необхідної кількості засобів відображення, у ролі яких можуть виступати будь які пристрої, що працюють у операційній системі як дисплей.

5.1.2 Аналіз техніко-економічних переваг ідеї

Системи керування відеостінами представлені на ринку великою кількістю конкурентів. Розглянуті декілька найбільш популярних компаній та у таблиці 5.2 наведено порівняння з моїм проєктом – контролери Christie, VuWall, AMC.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W	N	S
		Мій проєкт	Конкурент 1	Конкурент 2	Конкурент 3	слабка сторона	нейтральна сторона	сильна сторона
1	Прив'язка до апаратних компонентів	відсутня	присутня	присутня	присутня			+
2	Функціональність програмного забезпечення	висока	висока	висока	низька		+	
3	Підтримка різних пристроїв відображення	+	+	+	-		+	
4	Вартість впровадження	низька	висока	висока	середня			+
5	Технології розробки	нові	застарі-ли	нові	застарі		+	
6	Рівень технічної підтримки	низький	високий	високий	середній	+		
7	Підтримка одночасного керування	+	+	+	-		+	
8	Складність заміни обладнання	низька	висока	висока	середня			+

Згідно з таблицею 5.2, бачимо, що ідея проєкту має значні переваги над конкурентами, що робить проєкт перспективним. Треба звернути увагу на слабкі сторони, бо це може бути крупним недоліком, що наддасть значну перевагу конкурентам.

5.2 Технологічний аудит ідеї проєкту

Потрібно зробити технологічний аудит ідеї проєкту, щоб визначитися чи існують на даний момент потрібні нам технології та перевірити їх доступність.

Таблиця 5.3 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології реалізації	Наявність технології	Доступ-ність технології
		Можливість використання декількох відеокарт	Так	Так
		Наявність достатності ліній PCIe	Так	Так
		Можливість виносу відеокарт за межі системного блоку	Так	Так
		Мова програмування C#	Так	Так
		Середовище розробки Visual Studio	Так	Так
		Бібліотеки мови програмування C# для створювання інтерфейсів користувача WPF	Так	Так

Роблячи висновок з таблиці 5.3, можливо побачити, що існуючих технологій достатньо як за для розробки апаратної частини, так і для створення програмного забезпечення.

5.3 Аналіз ринкових можливостей запуску стартап-проєкту

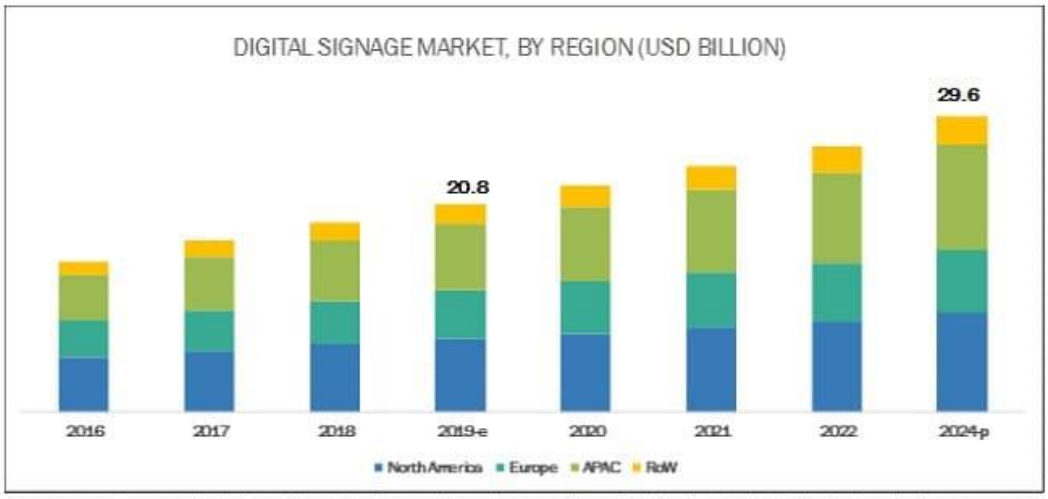
Провівши аналіз ринку на наявність попиту та обсяг продаж у галузі відеостін, заповнимо таблицю 5.4 з характеристиками ринку.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	10
2	Загальний обсяг продаж, грн/ум.од	29600 тис $
3	Динаміка ринку (якісна оцінка)	стагнує
4	Наявність обмежень для входу (вказати характер обмежень)	Увага на функціональні патенти
5	Специфічні вимоги до стандартизації та сертифікації	відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	19

Рентабельність, яка перевищує банківський відсоток, та щорічний зріст ринку у галузі, демонструє, що владати гроші у даний проєкт вигідніше ніж у інший.

Згідно з рисунком 5.1 бачимо, що ринок зростає з року у рік, тобто вкладання у сферу відеостін є привабливим. Також згідно з прогнозом зростання буде подовжуватися у наступні роки та збільшить ринок майже до тридцяти мільярдів доларів у 2024 році. Якщо подивитися на малюнок то також можливо побачити зростання обсягу майже удвічі з 2016 року.



Source: Industry Experts, Secondary Research, and MarketsandMarkets Analysis

Рисунок 5.1 – Зростання об’єму ринку

Проаналізувавши потенційних покупців нашого проєкту, бачимо результат у таблиці 5.5, у якої зазначена цільова аудиторія, та вимоги споживачів. Як бачимо обидві зазначені потреби є дуже привабливими на ринку.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних цільових груп клієнтів	Вимоги споживачів до товару
1	Рекламні панелі	Рекламні агентства Торгові центри Великі компанії	Ціна Розміри панелей місця установки Електроживлення	Робота в реальному часі керування з одного місця
2	Диспетчерські	Корпоративний та державний сегмент	Ціна розміри панелей функції підтримка	Робота в реальному часі інтеграція з корпоративним и сервісами

Також треба мати на увазі основні фактори загроз та можливостей які надані у таблиці 5.6 та таблиці 5.7 відповідно, щоб забезпечити відповідну реакцію компанії. Це надасть можливість заздалегідь проаналізувати можливу поведінку у разі загроз, чи навпаки завдяки можливостям продукту.

Таблиця 5.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкурент	Вихід на ринок великого конкурента	Випуск ранньої версії з обмеженим функціоналом
2	Патенти	Виявлення патентів, що заважають реалізації	Відмова від частини функціоналу, реалізація з урахуванням патентів конкурентів.
3	Санкції	Заборона покупок в певних станах	Вибір нейтральної країни для реєстрації

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Ранній вихід на ринок	Наявність готового прототипу дозволить раніше вийти на ринок	Продаж раннього доступу з націнкою
2	Якість продукції	Якісні коплектуючі, дозволяють конкурувати на ринку	При необхідності здешевлення продукту, можна перейти на більш дешеві аналоги
3	Ціна	Можливе використання будь яких комплектуючих	Створення списку тестованих рішень

Отже, за наданими результатами бачимо яку реакцію необхідно приймати компанії у тому чи іншому випадку

Результати ступеневого аналізу конкуренції на ринку та аналізу за М.Портером надані у таблиці 5.8 та таблиці 5.9 відповідно.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції – монополія	Галузь в основному є конкурентною. Ціна може зростати без втрачання усієї кількості торгових угод.	При малому обсязі, можуть демпінгувати конкуренти. Основний напрямок розвиток функціоналу, відсутнього в інших.
Рівень конкурентної боротьби – національний	Можливість міжнародних продажів, немає прив'язки до певної культури	Просувати товар на ринках різних країн, через партнерів
Галузева ознака – міжгалузева	Використовуються як досягнення в електроніці, так і в ІІ	Використання найбільш сучасних технологій
Конкуренція за видами товарів: – товарно-родова	Конкуруємо з звичайними дорогими системами	Основний опір на демонстрацію переваг над існуючими
За характером конкурентних переваг – нецінова	Основний опір на функції, яких немає у конкурентів	Розвивати функціонал, якого немає у конкурентів
За інтенсивністю – не марочна	Прив'язка до бренду відсутня, новий продукт	Розвивати ім'я бренду

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Навести перелік прямих конкурентів	Визначити бар'єри входження в ринок	Визначити фактори сили постачальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників
Висновки:	WuWall Christie Сильна боротьба. При рівній якості обладнання, основна боротьба буде за функціонал. Ситуація спрощується повільним розвитком функціоналу у великих компаній.	AMC. Основна проблема з якістю обладнання. Використовуються дешеві китайські аналоги. Основна сила конкуренції - цінова боротьба.	Постачальники, довгий час працюють з великими фірмами можуть диктувати умови і вимагати знижки	Можлива проблема з клієнтами консерватори, які вважають за краще брендовы аппаратні рішення.	Обмежень практично немає, серце системи - ПО Апаратна складова легко змінюється.

Аналіз факторів конкурентоспроможності наданий у таблиці 5.10, демонструє можливість конкурувати з потенційними конкурентами.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	якість	Можуть використовуватись брендові комплектуючи, підтримка за допомогою ОС.
2	функціонал	Більше функціоналу, ніж у конкурентів, корпоративні функції, безпека, USB GPU та ін.
3	Ціна	Можливість використання будь якого обладнання, яке підтримує ОС
4	Підтримка	Можливість використання обладнання конкурентів

Порівняльний аналіз сильних та слабких сторін, наведений у таблиці 5.11, демонструє, що найбільш сильною стороною є фінішна ціна проєкту.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін стартап-проєкту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з стартап-проєктом						
			–3	–2	–1	0	+1	+2	+3
1	якість	15					*		
2	функціонал	15					*		
3	ціна	18							*
4	підтримка	18						*	

Результати SWOT- аналізу, з демонструванням сильних, слабких сторін проєкту, а також можливостями та загрозами наданий у таблиці 5.12

Таблиця 5.12 – SWOT- аналіз стартап-проєкту

Сильні сторони: якість функціонал ціна	Слабкі сторони: бренд
Можливості: Ранній вихід на ринок	Загрози: конкуренти патенти санкції

За результатами альтернативи ринкового впровадження проєкту, наведеними у таблиці 5.13, найкращим варіантом є пункт 1

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Випуск ранньої версії з обмеженим функціоналом	висока	3-4 мес
2	Відмова від частини функціоналу, реалізація з урахуванням патентів конкурентів. Реєстрація патентів на свої функціонал	середня	6-8 мес

5.4 Розроблення ринкової стратегії проєкту

Визначивши потенційних споживачів, дані о яких надано у таблиці 5.14, можна починати утворювати ринкову стратегію.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Диспетчерські 55%	Висока, дуже багато потреби у корпоративному сегменті	Висока, збільшення застарілого обладнання без підтримки	Середня. Багато вендорів, але завелика ціна. Обладнання вендор-специфічне	Середнє, багато потенційних клієнтів
2	Торгові центри 40%	Висока, йде збільшення відеореклами на панелях	Висока, кількість центрів постійно зростає	Середня, присутні конкуренти	Середнє
3	Кінцеві споживачі 5%	Низька, для одиничних користувачів нецікаво	Низка, доволі дорого для домашнього використання	Мале, немає поклику у сегменті	Середнє

Згідно з аналізом базової стратегії розвитку, наданому у таблиці 5.15, обираємо стратегію диференціації. Ця стратегія більш підходе так як робити функціонал

ширший за конкурентів буде складно, а споживчі комплектуючи значно здешевлять систему, що дозволить конкурувати навіть з таки самим функціоналом як у конкурентів.

Таблиця 5.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розвиток підтримки споживчих комплектуючих	власна або залучена система збуту	Низька ціна, легка кастомізація	Стратегія диференціації
2	Розвиток функціоналу	власна або залучена система збуту	Крос-платформа, веб-додаток	Стратегія спеціалізації

Аналіз конкурентної поведінки, наданий у таблиці 5.16 показує найкращим вибір стратегії заняття конкурентної ніші. Так як на ринку немає систем здатних працювати з обладнанням споживчого сегменту, а наслідування лідеру буде заскладним, тому що конкуренти мають дуже розвинений функціонал. Також стратегія диференціації наддасть можливість швидко масштабувати систему, та зменшить час ремонту, так як усі деталі системі є у вільному продажу.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№	Чи є проєкт	Чи буде	Чи буде компанія	Стратегія

п/п	першопроходцем на ринку?	компанія шукати нових споживачів, або забирати існуючих у конкурентів?	копіювати основні характеристики товару конкурента, і які?	конкурентної поведінки
1	Так. Розвиток підтримки комплектуючих споживчого сегменту	Пошук нових споживачів	Ні	Стратегія заняття конкурентної ніші
2	Ні. Розвиток функціоналу відеостін	Конкурентна боротьба	Так, управління у реальному часі, та інші функції систем конкурентів	Стратегія наслідування лідера

Проаналізувавши вище надану інформацію з обраних сегментів та обравши стратегію конкурентної поведінки, визначаємо найкращі стратегії позиціювання, надані у таблиці 5.17.

Стратегія позиціювання дозволить утримувати конкурентоспроможність та мати лояльних споживачів. Потрібно обрати основні стратегії на ринку, які будуть підходити до нового бренду, щоб мати можливість вийти на ринок та зайняти свою частку. Це дозволить визначити унікальність бренду та наддасть підтримку споживачів.

Таблиця 5.17 – Визначення стратегії позиціювання

№ п/п	Вимоги до товару	Базова стратегія	Ключові конкурентоспр	Вибір асоціацій, які мають сформувати

	цільової аудиторії	розвитку	оможні позиції власного стартап-проєкту	комплексну позицію власного проєкту (три ключових)
1	Використання звичайних GPU	Позиціювання по споживачу	Непотрібні рідкі вендор-спеціфічні комплектуючи	Ціна Гнучкість
2	Крос-платформа	Стратегія диференціації	Можливість використання багатьох платформ	вибір Opensource
3	Функціонал	Стратегія спеціалізації	Реалізація функціоналу конкурентів	сценарії інтеграція
4	Обладнання	Стратегія спеціалізації	Підтримка застарілих систем	Підтримка Період використання

5.5 Розроблення маркетингової програми стартап-проєкту

Спочатку треба проаналізувати продукт, та визначити його переваги, які надані у таблиці 5.18

У нашому випадку ключовими перевагами будуть ціна, та доступність. Використання комплектуючих споживчого сегменту робить систему дешевою, та надає можливість зменшити час на її ремонт або модернізацію.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує	Ключові переваги перед

п/п		товар	конкурентами, (існуючи або такі, що потрібно створити)
1	Дешевий аналог	Ті ж самі функції за меншою ціною	Підтримка усіх видів додатків доступних до запуску у операційній системі. Інтеграція з корпоративними системами. Використання шифрування у протоколах обміну. Зручне керування у реальному часі. Функціональне керування. Масштабування. Підтримка сценаріїв. Підтримка розкладу.
2	Звичайні комплектуючи	Використання всіх GPU на ринку, які підтримує операційна система	Система дозволяє не використовувати дорогі контролери, та замінити їх на звичайні GPU. Швидка заміна. Легка модернізація. Просте масштабування.

Більш детальний опис та виводи надані у таблиці 5.19, після аналізу трьох рівнів моделі.

Отже, пропонується створити дві версії системи, та заробляти на продажу додаткових опцій, які найчастіше зустрічаються у корпоративному сегменту.

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові

<table>
<tr><td>І. Товар за задумом</td><td colspan="3">ПЗ для управління відеостіною. Система базується на ПЗ, класичному сервері / ПК і відео-панелях. По суті це звичайний ПК з операційною системою, в який встановлено кілька відеокарт з підключеними моніторами і програмне забезпечення, яке управляє контентом. Додатково можна встановити ТВ тюнер і подібні пристрої для відео-захоплення. Система дозволяє запускати на відео стіни такі додатки як карти, системи відеоспостереження, SCADA клієнти, ПО власної розробки та ін..</td></tr>
<tr><td rowspan="6"></td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Панелі LG GTFJ83374 4k
2. ПК
3. ПЗ, річна підтримка
4. Пакування</td><td>М
М
Нм
М</td><td>Тх/Е/ВР
Тх/Е/ВР
Вв/Е/Ор
Тх/Е/ВР</td></tr>
<tr><td colspan="3">Якість: ISO9000,вибіркове тестування. Тестування комплектуючих виробником</td></tr>
<tr><td colspan="3">Пакування картон коробка, ребра дерево</td></tr>
<tr><td colspan="3">Марка: EasyWall</td></tr>
<tr><td colspan="3">До продажу — Продаж опцій: інтеграція з корпоративними сервісами, підтримка безпеки, розширені функції</td></tr>
<tr><td></td><td colspan="3">Після продажу — продаж підтримки, інтеграція з новими сервісами</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: Основна цінність — професійна версія програмного забезпечення, яке захищене ліцензійним ключем, що визначає набір додаткових опцій, та підтримка яка захищена ліцензійною угодою .</td></tr>
</table>

Визначення меж встановлення ціни надане у таблиці 5.20, на основі аналізу ринку.

Таблиця 5.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	-	5000$	3000000 грн	30000-50000 грн

Далі потрібно вибрати оптимальні системи збуту, надані у таблиці 5.21, на основі специфіки закупівельної поведінки клієнтів.

Таблиця 5.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Великі замовлення від споживачів	Прямий продаж споживачеві. Продаж ПЗ через інтернет	Канал першого рівня	Торгівля через виробника
2	Одиничні замовлення, замовлення вимагають інтеграції та систем “під ключ”	Прямий продаж	Канал другого рівня	Прямий продаж

Останньою частиною має бути концепція маркетингових комунікацій, надана у таблиці 5.22, яка залежить від специфіки поведінки цільових клієнтів.

Таблиця 5.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціювання	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Власники магазинів	Email SMM Контент-маркетинг	Кінцеві рішення з панелями	Демонстрація можливостей, ціна, підтримка	Відеореклама
2	Рекламні агентства	Email Контент-маркетинг	Інтерактивні рекламні щити, кінцеві рішення	Керування багатьма щитами з однієї точки	Приклади одночасної швидкої зміни контенту по всьому місту
3	Корпоративні підприємства диспетчерські	Контент-маркетинг Публікації Виставки	інфо панелі, ПЗ, підтримка	Демонстрація готових рішень, приклади відновлення використання застарілих рішень	Приклади диспетчерських, та інтеграція з корпоративними сервісами

Висновки до розділу 5

На основі проведеного аналізу у розділі 5, можна визначити, що до проєкту є попит, динаміка ринку гарна, тобто є можливість ринкової капіталізації проєкту.

З огляду на потенційні групи клієнтів, середні бар’єри входження, стан конкуренції та конкурентоспроможність проєкту, можливо зробити висновок гарних перспектив впровадження проєкту. Найкращим варіантом альтернативою, може бути випуск ранньої версії, з обмеженим функціоналом.

Подальша імплементація проєкту є доцільною, у рамках розробки додаткового функціоналу, з огляду потреб корпоративних підприємств та інтеграцією з основними сервісами.

ВИСНОВКИ

У ході виконання магістерської дисертації проведено огляд існуючих на ринку рішень для побудови систем керування відеостінами. Визначені ключові недоліки таких систем та розглянута можливість їх уникнути. Була розроблена система яка дозволяє побудувати відеостіну та керувати нею за допомогою обладнання споживчого сегменту.

Було виконано наступні завдання:

- проаналізовані обмеження обладнання споживчого сегменту;

- визначені методи використання великої кількості відеокарт на материнських платах з обмеженою кількістю PCIe слотів;

- проведено аналіз існуючих пристроїв виводу зображення та їх кріплення;

- розроблене програмне забезпечення для керування відеостіною;

- розроблені допоміжні додатки.

Результатом дослідження є обґрунтування можливості використання обладнання споживчого сегменту у побудові відеостін великого розміру та з достатнім функціоналом. Було отримане гнучке рішення, яке дозволяє значно здешевити кінцеву вартість системи, прискорити можливість заміни обладнання, ремонту, модернізації та масштабуванню системи.

Функціонал розробленого програмного забезпечення відповідає усім сучасним потребам, та є аналогом існуючих на ринку рішень. Програмний продукт представляє собою клієнт-серверну архітектуру з допоміжними додатками. Має можливість безпечного обміну інформацією завдяки шифруванню. Програмні додатки розроблені за допомогою технологій .Net Framework, WPF та бази даних SQLite.

Завдяки дослідженню та розробці маємо можливість використовувати комплектуючи споживчого сегменту у реальних системах, що підтверджується впровадженням на великому промислову підприємстві.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Куделин П. Видеостены и многоэкранные инсталляции / Куделин П., Старов А./ Ridero, 2018. – 202 с.

2. Lars-Ingemar L. Digital Signage Broadcasting / Lars-Ingemar Lundstrom / Taylor & Francis, 2013. – 352 с.

3. Тюнин Н.А. ЖК мониторы. / Тюнин Н.А. / Никея, 2006. – 108 с.

4. Родин А. В. Бюджетные ЖК мониторы. / ред. Родин А. В., Тюнин Н. А. / М.: Солон-Пресс, 2016. – 136 с.

5. Все о мультимедіа проєкторах [Електронний ресурс] – Режим доступу до ресурсу: https://www.allprojectors.ru/

6. Світлодіодні екрани [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Светодиодный_графический_экран

7. Digital signage [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Digital_Signage

8. Контролери AMC [Електронний ресурс] – Режим доступу до ресурсу: https://www.prosoft.ru/products/types/501472/501476.html

9. Christitie Digital [Електронний ресурс] – Режим доступу до ресурсу: https://www.christiedigital.com/emea/video-walls

10. Контролери VuWall [Електронний ресурс] – Режим доступу до ресурсу: https://vuwall.com/products/

11. Аналіз вимог [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Аналіз_вимог

12. Statcounter Global Stats [Електронний ресурс] – Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share

13. The Northeastern University System [Електронний ресурс] – Режим доступу до ресурсу: https://www.northeastern.edu/graduate/blog/most-popular-programming-languages/

14. Svetlin N. Fundamentals Computer Programming with C# / Svetlin N, Vaselin K / Faber, 2013. – 1122 с.

15. Matthew M. Pro WPF 4.5 in C# 5 Windows Presentation Foundation in .NET 4.5 / Matthew MacDonald 4 изд. / Вильямс, 2013. – 1024 с.

16. Joseph A. C# 3 in a Nutshell / Joseph A., Ben A. / OReilly Media, 2007 - 864 с.

17. Мережеве програмування у C# та .Net [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/net/

18. Шифрування тексту [Електронний ресурс] – Режим доступу до ресурсу: https://vscode.ru/tag/shifrovanie-teksta

19. John M. Compressed Image File Formats/ John M / Addison-Wesley Professional, 1999. – 264 с.

20. Paul S. SQLite Forensics / Paul Sanderson / Amazon, 2018. – 315 с.