Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи Digital Signage для реалізації корпоративного телебачення (IP TV)”

КБПЗ_2025

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-МБ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Ісаєнко А.О.
«____» _____ 2025 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*“Інформаційні технології”*_____
Спеціальність *123 “Комп’ютерна інженерія”*
Освітньо-професійна (освітньо-наукова) програма _*“Комп’ютерна інженерія”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Ісаєнку Артему Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи Digital Signage для реалізації корпоративного телебачення (IP TV)*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 48-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи Digital Signage для реалізації корпоративного телебачення (IP TV)*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

КБПЗ-2025

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

КБПЗ_2025

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Ісаєнко А.О.
(прізвище та ініціали)

АНОТАЦІЯ

Ісаєнко А.О. Програмне забезпечення системи Digital Signage для реалізації корпоративного телебачення (IP TV). 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи Digital Signage для реалізації корпоративного телебачення (IP TV).

Метою розробки є програмне забезпечення системи Digital Signage для реалізації корпоративного телебачення (IP TV).

Результат роботи – програмна реалізація системи Digital Signage для реалізації корпоративного телебачення (IP TV).

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, Digital Signage

КБПЗ_2025

ABSTRACT

Isayenko A.O. Software for the Digital Signage system for the implementation of corporate television (IP TV). 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for the Digital Signage system for the implementation of corporate television (IP TV).

The purpose of the development is the software for the Digital Signage system for the implementation of corporate television (IP TV).

The result of the work is the software implementation of the Digital Signage system for the implementation of corporate television (IP TV).

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with OS Windows 10/11.

The program was developed in the Python environment.

Keywords: computer engineering, Digital Signage

КБП3_2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи 5
1.2 Область застосування 6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 8
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти 8
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 23
2.3 Розгорнута постановка завдання 28
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 30
3.1 Опис функціонування системи 30
3.2 Розробка структурної схеми 35
3.3 Розробка функціональної схеми 50
3.4 Розробка діаграми процесів 61
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 63
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 63
4.2 Захист розробленого програмного забезпечення 76
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 78
6 ОСНОВНІ ВИСНОВКИ 85
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 87

КБПЗ_2025

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ВКЗ	–	відеоконференцзв’язок
ПЗ	–	програмне забезпечення
CMP	–	Codian Management Platform
DNS	–	Domain Name System
GMS	–	Global Management System
GRE	–	Generic Routing Encapsulation
ISDN	–	Integrated Services Digital Network
MCU	–	Multipoint Control Unit
MPLS	–	Multiprotocol Label Switching
MXM	–	Media eXchange Manager
SSL	–	Secure Socket Layer
TLS	–	Transport Layer Security
TMS	–	Tandberg Management Suite
VPN	–	Virtual Private Network
USENET		User Network

К6ПЗ_2025

ВСТУП

Актуальність теми. Цифрові інформаційно-рекламні вивіски, більше відомі фахівцям під англомовним терміном Digital Signage, знаходять всі нові застосування в різних галузях.

Прабатьками сучасних систем Digital Signage можна вважати неонові вивіски для реклами товарів, які з'явилися в 20-і роки минулого століття. За минулі майже сто років технології кардинально змінилися, на зміну аналоговим прийшли цифрові рішення, а в останнє десятиліття бурхливому розвитку рішень Digital Signage сприяло застосування сучасних інформаційних технологій. Якщо, будучи у великому місті, ви вийдете з Інтернету, відірвавшись від екрана смартфона або планшета, то відразу потрапите в мир інформаційно-рекламних вивісок. Вони оточують нас усюди: на вулицях і в транспорті, у банках і музеях, у медичних установах і на спортивних об'єктах, ну й звичайно, у торгових центрах.

Консалтингова компанія invidis consulting регулярно відслідковує стан ринку Digital Signage в Україні. Опитуючи ведучих гравців ринку, її експерти обчислюють відповідний індекс бізнес-клімату (Digital Signage Business Climate Index). Якщо ще на початку 2024 року цей показник перебував на негативному рівні (-3,56), то під кінець року він істотно покращився – до рівня 42,30. Як відзначається в дослідженні invidis consulting, «поточна ситуація на ринку оцінюється гравцями, що прийняли участь в опитуванні, як «дуже гарна», протягом наступних шести місяців на ринку рішень і послуг Digital Signage очікується стабільність».

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи Digital Signage для реалізації корпоративного телебачення (IP TV).

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

КБПЗ-2025

– Огляд існуючих систем Digital Signage для реалізації корпоративного телебачення (IP TV).

– Дослідження системи Digital Signage для реалізації корпоративного телебачення (IP TV).

– Програмна реалізація системи Digital Signage для реалізації корпоративного телебачення (IP TV).

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі Digital Signage для реалізації корпоративного телебачення (IP TV).

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи Digital Signage для реалізації корпоративного телебачення (IP TV), є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

К6ПЗ_2025

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Головною тенденцією на українському ринку Digital Signage в 2024 році експерти називають посилення конкуренції. Це цілком природний наслідок поліпшення бізнес-клімату в даній області. Підвищення рівня конкуренції зв'язано й з виходом на цей ринок великої кількості гравців з інших сегментів. Традиційно системами Digital Signage займалися постачальники професійних дисплеїв і компанії, що спеціалізуються на аудіовізуальних технологіях (АВ). Однак у міру конвергенції АВ і IT, на них звернули увагу й великі IT-інтегратори.

Як показало дослідження invidis consulting, що зростає попитом на ринку Digital Signage в 2024 році будуть користуватися бюджетні рішення. Це й зрозуміло з обліком не найкращої економічної ситуації в країні. Основну тенденцію, що виділила наша компанія в області Digital Signage, можна охарактеризувати одним словом – економія. Ми бачили, як різко скорочувалися бюджети на проекти, устаткування замінялося на більше доступні за ціною аналоги, скорочувалася кількість закуповуваних ліцензій, а багато великих проектів були відкладені до «кращих часів».

Але, незважаючи на економічні складності, що пройшов 2024 рік вселив оптимізм більшості експертів, що працюють у сфері Digital Signage. Останнім часом клієнти усе більше усвідомлюють потенціал Digital Signage, що виражається в підвищенні попиту. Так, незважаючи на непросту ринкову ситуацію, продажі по даному напрямку показують упевнену динаміку (в 2024 році ріст приблизно на 30% у порівнянні з 2024-м, у порівнянні з 2025 роком він ще більше). Топ-лист вертикальних ринків – споживачів систем Digital Signage, по даним Auvix, залишився незмінним: це корпоративні комунікації, рітейл і транспортна галузь.

КБПЗ – 2025

Відзначимо ріст числа інсталяцій систем Digital Signage на виробничих площадках, де інформаційні панелі використовуються для відображення KPI, статусів, графіків і таблиць, а також підвищення уваги до інтеграції Digital Signage з іншими IT-системами (. Крім того, він указує на те, що усе більше великих проектів орієнтуються на використання убудованих плеєрів. Якщо два роки тому убудовані медіаплеєри мали примітивний функціонал, то сьогодні вони являють собою потужну платформу, на базі якої можна створювати складні рішення.

1.2 Область застосування

Системи Digital Signage затребувані компаніями самих різних видів діяльності: скрізь, де коштує завдання оперативного повідомлення до аудиторії актуального інформаційного й рекламного контенту.

По даним IMS, основними споживачами рішень Digital Signage в 2024 році стали музеї, ресторани швидкого харчування, транспортні компанії, банки. У музеях ці рішення використовуються для організації яскравих експозицій і інсталяцій, у ресторанах швидкого харчування – для переходу на цифрові меню, у банках – для формування реклами, що запам'ятовується, фінансових продуктів і послуг на дисплеях і в інтерактивних кіосках, для інформування й розваги клієнтів, організації зручної цифрової черги.

Транспортні компанії встановлюють відеостіни, дисплеї й кіоски на авіа- і ж/д вокзалах, відеопанелі – усередині рухливих составів. Як потенційні клієнти привабливими залишаються метро Києва й Харкова. Бюрократичні перепони гальмують розвиток проектів, але рано або пізно дані об'єкти однаково будуть оснащені системами сучасної цифрової реклами.

Системи Digital Signage починають усе активніше застосовуватися для інформування вболівальників на спортивних об'єктах. Рішення забезпечує трансляцію матчів і рекламних повідомлень на 300 цифрових екранах стадіону.

КБПЗ_2025

Необхідний контент виводиться оператором в один клич на інформаційні панелі з урахуванням їх місцеположення: VIP-ложа, магазин сувенірів, фуд-корт і пр.

Почали виявляти цікавість до систем Digital Signage і виробничі компанії. Ключова мета замовника складалася в оперативному контролі виробничих показників. На екранах, які встановлені на виробничих лініях на фабриках, у режимі реального часу відображаються показники функціонування устаткування, результати роботи кожної групи в зміні та ін. Це відповідає концепції так званого ощадливого виробництва.

Існує також проект який охоплює адміністративні й виробничі будинки підприємства. Завданнями корпоративної системи Digital Signage є інформування співробітників і гостей підприємства про важливі корпоративні події, висвітлення різних заходів (виступу керівників підприємства, матеріали про підприємство в ЗМІ й т.д.), а також зворотний зв'язок зі співробітниками, здійснювана за допомогою інтерактивних засобів. До складу рішення входять РК-панелі (з діагоналями від 55 до 82”) з убудованими програвачами, інтерактивні кіоски, а також сервер керування. Всі вони підключені до спеціально організованого сегмента локальної мережі, а керування ними здійснюється як єдиною системою.

Систему Digital Signage планується використовувати як свій рід планувальника виробничих завдань, також на ній повинні відбиватися досягнуті результати робіт, різні показники та ін.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи Digital Signage для реалізації корпоративного телебачення (IP TV), є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ_2025

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Система відеотрансляції TNTv Digital Signage

Подібне рішення ми вже вивчали два роки тому. Так, до звичних медіаплеєрів або інтерактивним приставкам дане устаткування не віднесеш. І серед домашнього парку техніки такий апарат навряд чи зустрінеш. Але от на рекламному фронті, у магазині, у різних установах і організаціях, а також у транспортній сфері такий пристрій нині є чи ледве не обов'язковим атрибутом бізнесу й будь-якої іншої діяльності, як державної, так і суспільної.

Може здатися, що в рамках роздягнула, де вивчається техніка для рядового споживача, огляд такого пристрою виглядає недоречно. Але як же допитливість? Невже не цікаво, за допомогою чого організується суспільне відеовіщання на вуличних екранах, у готелях, барах, магазинах, у навчальних і інших установах? Скласти уявлення про виворіт суспільної відеотрансляції нам допоможе контент-плеєр **TNT SX 325 M-W**.

Комплектність, конструкція

Корпус апарата по суті є одним більшим радіатором, він виготовлений з металу – ні найменшого натяку на пластик. Зрозумілий хід, адже пристрій призначений для постійної роботи в режимі 24/7 і 365 днів у році. Зовсім невідомо, у яких умовах виявиться плеєр – жара або холод – отже, тут потрібна гарантований захист від перегріву. А що може бути кращою гарантією від згубного впливу високих температур, крім металевого корпуса з більшою кількістю ребер, що відводять тепло?

КБПЗ – 2025

У комплекті з контент-плеєром, наданим для тестування, були Wi-Fi-антена, блок живлення з кабелями, і монтажні металеві планки для кріплення пристрою в стійку. Тут же, у комплекті, виявився компакт-диск, де втримується контент і шаблони для створення користувальницьких сценаріїв трансляцій.

Лицьова панель контент-плеєра містить тільки два порти USB 2.0, ці порти призначені для підключення периферійних пристроїв. Наприклад, що здобувається окремо GPS-приймача. Ніяких індикаторів, що показують поточний стан пристрою, немає. Це з'ясовно: даний апарат не повинен перебувати на очах. Йому уготовано бути захованим у якій-небудь підсобці, технічному приміщенні.

На задній панелі, ліворуч, можна бачити заглушку, що прикриває отвір з написом Tuner. Цей отвір призначений для високочастотного входу ТБ-тюнера, що є в іншій моделі плеєра, з літерним позначенням T. Ми ж розглядаємо модель W, що не має убудованого тюнера.

Крім цієї заглушки, тут є наступні працюючі інтерфейси:

– кнопка включення/відключення живлення;

– VGA- і HDMI-виходи на монітор, телевізор та інші пристрої відображення;

– два порти USB 2.0;

– LAN-порт;

– вхід мережного адаптера живлення;

– аудіовхід і вихід (Aux);

– рознімання для приєднання прикладеної Wi-Fi-антени.

Чимала частина маси плеєра доводиться на металеву пластину, з'єднану з металевою же кришкою. Ця пластина збирає тепло, що піднімається від нагрітих електронних компонентів плеєра, і рівномірно розподіляє його по кришці, форма якої, як уже говорилося, нагадує радіатор. Великий вільний обсяг внутрішнього простору також знижує ризик перегріву – після декількох доби безперервного відтворення сценарію корпус контент-плеєра нагрівся лише до 35°C (температура приміщення становила 20-22°C).

КБПЗ_2025

Налаштування, продуктивність, експлуатація

Як і у випадку з будь-якою професійної вузькоспеціалізованою технікою, спілкування з розглянутим пристроєм – справа непросте. Тут немає ніяких «дружніх, інтуїтивно зрозумілих і т.д.» інтерфейсів, властивій електроніці споживчого класу. Так, керування даним контент-плеєром відбувається тільки за допомогою мережних протоколів, а попросту – через локальну мережу в браузері. Але є й другий спосіб. Цей контент-плеєр може експлуатуватися як окремий пристрій, так і в складі єдиної мережної системи TNTv Digital Signage. Керування такою системою відбувається вже за допомогою апаратури іншого класу: CMS-сервера TNT SX (CMS – Content Management System, система керування контентом сайту).

Оскільки такого сервера в нашім розпорядженні не є, розглянемо процедуру керування пристроєм через браузер. Але колись потрібно зробити перше підключення плеєра. Це нескладно: до відеовиходів підключаються монітори, а порт LAN з'єднується з маршрутизатором або хабом. Включення пристрою займає близько 30 секунд, за це завантажується операційна система Ubuntu і веб-сервер Apache2, ініціалізується підключений накопичувач, відбувається мережне підключення, і, нарешті, автоматично стартує завантажений раніше сценарій. Якщо ж активних сценаріїв у розкладі не є, то плеєр відобразить екранну заставку, поверх якої буде написана поточний IP адреса апарата в локальній мережі.

Тепер у браузері на ПК, що входить у цю же локальну мережу, досить увести дану адресу, і у формі, що з'явилася, авторизації ввести необхідну «замолчувальну» привітальну **пару** administrator**:**welcome.

Варто мати на увазі: для повноцінної роботи браузер повинен бути оснащений розширеннями HTML5 Video Player 1.0 і вище, а також Windows Media Player Extension for HTML5 1.0 і вище. У нашім випадку ідеально підійшов браузер Internet Explorer 11, з іншими браузерами чомусь не зложилося.

КБПЗ_2025

Стартова веб-сторінка пристрою, вхід у яке був тільки що виконаний, являє собою нескладний інтерфейс із фіксованим розміром вікна, і декількома керуючими вкладками. Претензій до перекладу інтерфейсу на українську мову в нас не найшлося.

У першу чергу заглянемо в налаштування, щоб одержати подання про можливості пристрою. Ці налаштування розбиті на кілька категорій. При цьому, якби контент-плеєр був оснащений доповненнями, такими, як GPS приймач або ТБ-тюнер, то в даному меню з'явилися б відповідні пункти, призначені для налаштування ціх складових.

У наступних розділах *Керування* є можливість переглянути стан системи: завантаження мережі, процесора й пам'яті, список підключеного устаткування, а також виконати одне з наступних дій:

Наступні пункти можуть здатися нудними обивателеві, однак, вага їх важко переоцінити при повсякденній професійній експлуатації устаткування. Так, наприклад, плеєр фіксує кожну трансляцію в спеціальному журналі відтворення, що доступний будь-якому користувачеві, що має відповідні права. Таким чином, рекламодавець може в будь-який момент переконатися в правильності (або неправильності) рахунків, що виставляються йому, за трансляцію його реклами. Те ж ставиться й до системи екстрених повідомлень, що дозволяє використовувати монітори як система оповіщення. Нарешті, функція самодіагностики й оповіщення про системні події дозволить адміністраторові тримати руку на пульсі, бути завжди в курсі про поточний стан устаткування (а в комерційному використанні його несправність загрожує серйозними збитками).

Так, з налаштуваннями, здається, все зрозуміло. Багатими їх не назвеш, але й скромними – теж мова не повертається. Вслід на ціми загальними налаштуваннями перейдемо до ознайомлення з основним інструментом, без якого експлуатація пристрою неможлива в принципі.

Інструмент цей – редактор. Він складається із двох головних модулів: *Manage Content* і *Presentation Design*.

КБПЗ_2025

Перший пункт, *Контент*, призначений для перегляду наявних на диску плеєра медіаматериалов, поповнення бібліотеки й керування їй. Головна незручність тут – всі файли, наявні на диску плеєра, «звалені» в одну купу, а не рознесені по папках. Втім, будь вони й рознесені – однаково отут не є файлового браузера зі звичною навігацією. Однак пошук і сортування матеріалу полегшує така функція, як присвоєння файлам користувальницьких міток. Мітки ці бажано додавати ще на етапі імпорту контенту, оскільки згодом пошук потрібного матеріалу може виявитися скрутним. Таким чином, різні клієнти або групи клієнтів можуть легко відсортувати «свої» файли від «чужих», так само як і файли, що відносяться до одного проекту, від іншого. У підсумку необхідність використання структури папок зникає.

Як матеріал, придатного для імпорту на HDD контент-плеєра, можуть виступати будь-які файли, підтримувані системою, включаючи навіть PDF, а також посилання на веб-сторінки, і навіть на контент, що зберігається на FTP сервері (у цьому випадку до ім'я файлу варто додати логин і пароль для входу).

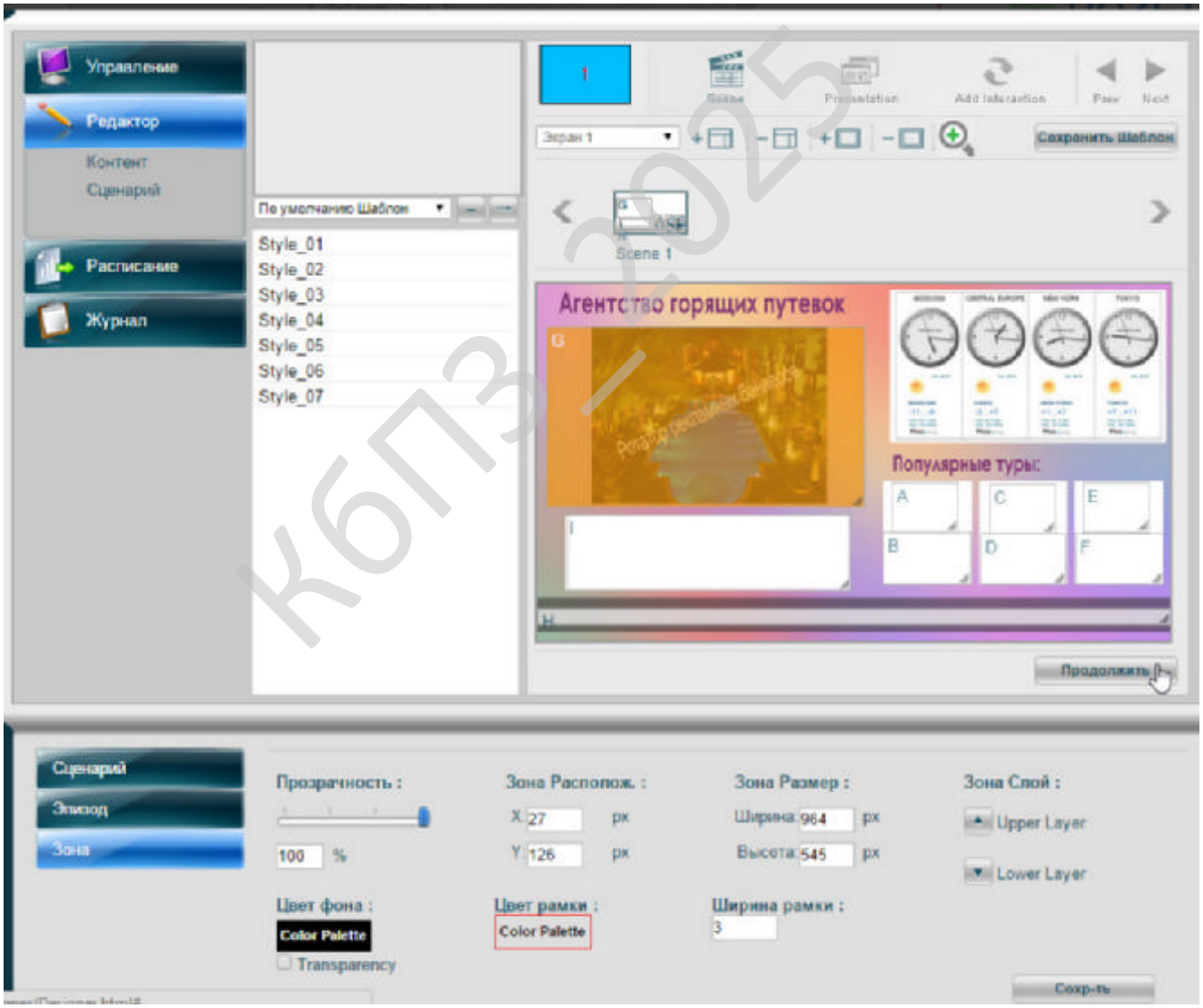
Весь цей матеріал придасться при створенні майбутнього шаблона, що передбачається транслювати за допомогою розглянутого плеєра. Тут такий шаблон називається *сценарієм*. Ці сценарії, у свою чергу, створюються в однойменному модулі. Покроковий майстер створення сценарію дозволить вибрати кількість дисплеїв, на які буде йти віщання, режим їхньої роботи (клон, незалежні або об'єднані екрани), орієнтацію й дозвіл екранів. Для візуального контролю в правій частині вікна відображається поточний результат налаштувань у вигляді мініатюр екранів.

Будь-який сценарій, створюваний тут, складається з *епізодів* і *зон*. Епізод у цьому випадку являє собою аналог однієї презентації, що може складатися з будь-якої кількості об'єктів. Зона ж – це складова частина епізоду, властиво демонструємий об'єкт, розміщений в епізоді: рисунок, анімація, що біжить рядок, відеоролик та інше. Однак, через обмеженість інтерфейсу редактори, занадто велика кількість об'єктів, що входять в один епізод, приводить до підвищеної

КБПЗ_2025

складності редагування, і чревате плутаниною. Щоб мінімізувати ризик помилок, розроблювачем були уведені такі епізоди (у нелокалізованій версії ПЗ вони називаються Scene, тобто, сценами). Кожний епізод має певну тривалість відтворення, по закінченні першого епізоду плеєр запустить другий, і так далі.

Редагування сценарію відбувається в невеликому віконці, де мишкою дозволяється змінити розміри й положення кожної зони. Додаткові параметри зон налаштовуються інструментами, розташованими в нижній частині вікна редагування: розмір, тривалість відтворення (відображення), ефект появи, параметри синхронізації з іншими об'єктами (зонами).



КбПЗ – 2025

Рисунок 2.1 – Інтерфейс користувача TNTv Digital Signage

Але поки ці зони – усього лише порожні прямокутники, а наповнення їх контентом відбувається в наступному кроці, у вікні, що нагадує тайм-лінію звичайного відеоредактора.

Тут користувач має можливість перетягнути мишкою обраний файл, завантажений раніше, і розмістити його в потрібнім місці на потрібній доріжці (зоні). Нагадуванням користувачеві служить розташована вгорі мініатюра епізоду із вказівкою розташування зон і букв латинського алфавіту, привласнених їм.

До речі, от одна з хитростей, до яких доходиш самостійно, у процесі експлуатації: якщо якийсь епізод потрібно оснастити фоновим звуком (музичний супровід фотослайдшоу, або голос диктора, що повідомляє про що-небудь), треба для цього створити зону будь-якого розміру, а щоб її не було видно на екрані, потрібно привласнити цій зоні повну прозорість, або помістити даний шар у саму нижню позицію. Після цього в наступному кроці треба лише перетягнути на відповідну доріжку імпортований раніше аудіофайл. Тепер епізод має звуковий супровід!

Збережений сценарій з'являється в загальному списку сценаріїв, і готовий до відтворення. Але про відтворення – пізніше.

Поки ж хочеться поділитися враженнями від проробленої роботи. Сказати «незручно» – не сказати нічого. Завдання, що може стояти перед користувачем, вимагає, здавалося б, набагато більше просунутих інструментів. Однак істина криється в наступному твердженні: обмеженість інтерфейсу, незручність його використання, неможливість навіть змінити розміри вікон, вся ця вбогість і стислість усього лише нагадує нам про сугубо професійне призначення розглянутого апарата. Так уже повелося – з погляду звичайного споживача, будь-яка професійна техніка має незручні інтерфейси, немов би навмисно ускладнені способи підключення й налаштування, і некомфортна в експлуатації, вимагаючи обов'язкового навчання й практики. Чому так повелося – питання окремий. Можливо, відіграє роль менталітет самих розроблювачів, які створюють продукт «для себе», не обертаючи особливої уваги на відсутність елементарних

КБПЗ-2025

споживчих якостей, але вкладаючи всі сили в підвищення точності, надійності й відказостійкості.

Тепер настав час глянути, як же все-таки запускається відтворення готового сценарію. Ні, не подвійним клацанням по іконці або рядку списку, як це було б реалізоване в техніку «для мас». Тут підхід набагато більш серйозний: перш, ніж запустити програвання якогось сценарію, варто створити розклад, тривалість якого може становити цілий місяць. Цей розклад зберігається в пам'яті пристрою, а відтворення запуститься автоматично відразу після завантаження апарата.

Властиво відтворення почнеться автоматично, як тільки призначений час програвання запланованого сценарію в розкладі збіжиться з поточним часом системи. Як цей процес виглядає в реальній експлуатації, покаже наступний ролик:

Для перегляду в більшому розмірі натисніть кнопку Оригінальний розмір.

Можна бачити, як контент-плеєр жваво реагує на зміни в розкладі, на поява в ньому нового сценарію, відразу, без довгих пауз, запускаючи його відтворення. Кількість і характер об'єктів, що входять до складу сценарію, не позначаються на часі формування фінальної картинки. Втім, перевантажувати епізод великоваговими відеороликами не рекомендується: такий тип контенту є самим вимогливим до ресурсів. До речі, раз вуж ми згадали про відтворення відео, чому б не вивчити ці можливості контент-плеєра більш заглиблено? Адже в більшості сценаріїв, які плеєр повинен програвати, напевно є присутнім відео самих різних форматів. І якщо якийсь ролик раптом відмовиться відтворюватися, залишивши замість себе чорний прямокутник – це буде виглядати, м'яко говорячи, непрофесійно.

Для початку перевіримо здатність апарата коректно відтворювати відео найпоширеніших сьогодні форматів. Для цього використовуємо наявну в нас скарбничку відеофайлів, а результати відтворення приведемо в окремій табличці.

КБПЗ-2025

Цього потрібно було очікувати: проблем з форматами, схоже, не передбачається. Але як є справа із правильністю відтворення? Нагадаємо: розглянутий апарат не має налаштувань частоти відеовиходу, ця частота встановлюється автоматично, по максимуму, що показує EDID приєднаного пристрою відображення. Не виключений варіант, що плеєр може дублювати або пропускати кадри з метою оптимальної витрати ресурсів. Що в деяких випадках приводить до помітного зниження плавності зображення, ривкам, стробу. Зрозуміло, у випадку з вуличним відеоекраном або вокзальним інформаційним табло навряд чи хто стане звертати увагу на якість відтворення. Однак, це не означає, що ми обійдемо стороною дане питання.

Вивчити плавність програвання контенту допоможе нескладний тест. Для перевірки коректності відображення відеофайлів з різною частотою кадрів ми традиційно скористаємося набором тестових роликів з переміщаються на один розподіл за кадр стрілкою й прямокутником. При тестуванні використовувалося відео з розміром кадру 1920?1080, але різними частотами: від 24 до 60 кадр/с.

Розглянутий плеєр підключався до Full HD-телевізора в тому самому, максимальному для даного ТБ, режимі – 1080 60p. Із представлених фотознімків видно, що при відтворенні Full HD-відео із частотою кадрів вище 30 розглянутий контент-плеєр нездатний видати щирий FPS, роблячи випадкові відкидання й дублювання кадрів. Звідси можна зробити вивід, що при створенні сценаріїв не потрібно використовувати подібний відеоконтент, обмежившись відеороликами з розміром кадру 1280?720 і частотою 30 кадрів у секунду.

Наш контент-плеєр не має налаштувань формування відеозображення (так просто тому, що це не відеоплеєр), отже, можна припустити, що тут можуть перебувати якісь інструменти, що трансформують відео самостійно, «без попиту». Перевірити дане припущення допоможе порівняння двох стоп-кадрів, захоплених з відеовиходу плеєра.

Читач, що «у темі», відразу вкаже на стоп-кадр із відеоролика, той, що праворуч – у наявності явний деінтерлейс. Так, так і є. У плеєрі завжди

КБПЗ-2025

активований примусовий деінтерлейс будь-якого відео, поза залежністю від його формату й стандарту. Навіть якщо відео має прогресивне розгорнення – як у нашім випадку – до нього однаково буде застосоване злиття рядків. Для чого це зроблено? Здається, з метою розвантажити користувача, щоб він не витрачав свою увагу й час ще й на селекцію підходящих форматів і стандартів. Ну а страждаюча від цього деталізація – чи не так вона важлива, якщо врахувати сферу застосування розглянутого пристрою?

Виводи

З недоліків розглянутого пристрою слід зазначити типово споживчі: обмеженість інтерфейсу інструмента редагування сценаріїв і неможливість імпорту контенту крім як по мережі. Плюси апарата не менш очевидні: можливість вилученого контролю й керування, розширення функціональності за допомогою підключення додаткових модулів, гарантія від перегріву, і, звичайно, загальна програмно-апаратна надійність. Також варто згадати про наявність у комплекті із плеєром великої бази шаблонів, які створені професійним дизайнером – ці шаблони, розраховані буквально на всі випадки, допоможуть оперативно створити потрібний сценарій.

Під час знайомства із плеєром раз у раз пригадувалися різні випадки злому й перехоплення керування подібними пристроями, коли зловмисник (або просто жартівник) віддалено, через інтернет, змінював трансльований на вуличних екранах контент. Що отут скажеш. Можна лише дати єдину пораду – не залишати в даному пристрої пароль за замовчуванням, що так і просить: *welcome*.

Незважаючи на достаток виробників систем керування контентов в Digital Signage, на українському ринку їх реально продає не дуже багато. На ринку світовому виробників набагато більше й розподіл місць інше.

Забираємо з розгляду системи, зібрані «на коліні», залишаємо тільки системи з мінімальним набором функціональних можливостей для побудови рішень Digital Signage:

КБПЗ_2025

– можливість роботи з динамічним контентом, тікери (RSS, індикатори валют і погоди);

– можливість підключати периферію. Наприклад, хоча б RS-232 порт для керування зовнішнім устаткуванням або одержання інформації із зовнішньої системи;

– робота з відео в різних форматах, зображеннями, їхнє компонування, створення ефектів;

– централізоване керування показами, створення розкладів;

– інтеграція із зовнішніми базами даних, бажано.

І бачимо в підсумку, що дуже небагато систем і виробників, які забезпечують цей, базовий, функціонал. По парі слів про кожне з них, з посиланнями на виробників.

Пам'ятаємо також, що рішення Digital Signage діляться на програмні й апаратно-програмні. У випадку чисто програмних рішень вам знадобиться апаратна основа, яку розглянули нижче.

1. BroadSign

Сайт виробника: http://broadsign.com/

У лінійці продуктів є й апаратно-програмне рішення й чисто програмне. Апаратно-програмне цікавіше. Перебуває в середньому ціновому діапазоні в порівнянні з іншими брендами. Для повноцінного керування контентом, одержання всіх можливостей потрібно купувати SaaS ліцензію на основний софт по керуванню системою. У безкоштовному базовій версії доступно далеко не все.

Одні з останніх версій плеєрів ми інтенсивно дивилися. Якщо цікаво, то от огляд ціх пристроїв.

На мій погляд, цікавий з погляду проектів середнього рівня складності. Складні речі я б на ньому не став піднімати. Софт не занадто зручний, але це як завжди ціна/якість.

КБПЗ_2025

2. C-nario

Сайт виробника: http://www.ycdmultimedia.com/

Донедавна продавався добре й використовувався великими українськими інтеграторами для різних проектів. Це програмна платформа, тобто потрібна апаратна база, куди буде встановлений софт від C-nario.

Останнім часом бренд почуває себе погано, дуже схоже, що незабаром закриється або буде кимсь куплений. Але проектів на ньому створено досить багато, тому згадати його точно поки треба.

Зробив для себе вивід про те, що система складна й з великою кількістю «проблем». Але при правильних інженерних руках основні речі робить.

3. DISE

Сайт виробника: https://www.dise.com/

Аналогічно попереднє, це програмне забезпечення. Ставимо на апаратні плеєри й одержуємо плеєр Digital Signage. Купуємо основну централізовану систему керування контентом і одержуємо всього функціонала. У нашій країні продається завдяки зусиллям компанії Моушен В'ю. По цій же причині левова частина проектів реалізована в рітейлі й фастфудах. Мені особисто система подобається. Є нюанси використання, але вони всі відомі й акуратно обходяться.

4. PADS4

Сайт виробника: http://www.netdisplaysystems.com/

Бренду в українських проектах кілька років, активно розвивається. Особисто софт руками не мацав, докладно не можу описати. Так, рішення програмне, знову ж потрібна апаратна платформа для установки. У мене поки що відчуття, що статей про PADS4 на порядок більше, ніж реальних інсталяцій, але можу помилятися.

5. Scala

Сайт виробника: http://scala.com/

Один з найстарших виробників систем Digital Signage. Програмна платформа, потрібна апаратна основа для установки. У принципі, у будь-якому

КБПЗ-2025

проекті Ви з нею зштовхнетеся у вигляді конкурента. Самі представники працюють активно й добре. Дуже вдала лінійка продуктів з погляду клієнта. Неодноразово чув подібні відкликання. Налаштовується досить легко й зрозуміло. Весь функціонал підтримує, усе можна зробити. Але що додати – і правда, один з основних брендів, на яких в Україні робляться проекти.

Можна будувати великі й більші мережі Digital Signage, підключати безліч плеєрів, робити складні алгоритми показу контенту.

6. Spinetix

Сайт виробника: http://www.spinetix.com/

Це апаратно-програмне рішення. Лінійка продуктів маленька, але більше й не потрібно. Рішення одне з найбільш дорогих, але й найбільш продуктивних/функціональних. Точно варто звернути увагу на цей бренд, якщо у вас завдання середньої або високої складності. Дуже цікаве ПЗ для підготовки контенту, напевно, найбільш потужне із всіх тут представлених.

Можна будувати великі й більші мережі Digital Signage, підключати безліч плеєрів, робити складні алгоритми показу контенту. Є безкоштовна версія софта з обмеженим функціоналом для простих завдань і платна, якщо завдання складна й вам необхідно зробити реально серйозну інсталяцію.

7. Triple Play

Сайт виробника: http://www.tripleplay-services.com/

Бренд підходить не для всіх проектів. Це програмна основа для трансляції контенту по IP. Це значить, що будувати територіально розподілені мережі на ньому не вийде. Основне застосування – системи корпоративного ТБ, Digital Signage на більших об'єктах з величезною кількістю дисплеїв, але там, де можна побудувати єдину мережу й управляти всіма потоками в рамках єдиної локальної мережі.

КбПЗ-2025

Ще декілька виробників, які намагаються продавати на українському ринку, але поки безуспішно:

ComQi

Компанія народилася після об'єднання EnQii і Minicom (яка у свою чергу належить Kramer Electronics). У нашій країні погано продається.

Cisco Digital Signs

Smartsign (http://www.smartsign.se)

Winmate і інші китайські бренди, яких мільйон...

Частка їхнього ринку мала, незважаючи на сильні позиції деяких на світовому ринку.

Ще, якщо цікаво, проводили відкрите тестування плеєрів (Spinetix, Aopen+Scala, Winmate і інші). Отут результати цього тесту.

Апаратна частина рішень

Здебільшого системи Digital Signage є рішеннями по керуванню контентом, тобто вони мають потребу в апаратній частині. Простіше говорячи, це софт, якому необхідно встановити на промисловий комп'ютер, після якого готовий пристрій є повноцінним плеєром Digital Signage.

Крім плеєра в системі присутня серверна частина, що займається плануванням і розподілом контенту, збором і аналізом інформації.

У лінійці виробників апаратних платформ багато рішень. Найпростішому – це плеєр з одним відеовиходом HDMI/DVI, на який установлюється операційна система (донедавна Windows 7, але зараз незабаром піде Win10) і потім установлюється клієнт однієї із систем керування контентом: Scala та інші.

Більше складні апаратні платформи можуть відразу мати кілька відеовиходів. Це зручно для підключення двох дисплеїв з виводом на них різної інформації або створення відеостіни розміром 2*1.

Ще більш складний варіант – додавання в плеєр модуля WiFi для синхронізації із сервером. Багато інсталяцій реалізовані так: плеєр з WiFi у

КБПЗ-2025

кафі/на вуличній відеостіні. Коли потрібно обновити контент, під'їжджає машина. Оператор, не виходячи з машини, підключається до WiFi мережі плеєра й завантажує відновлення контенту.

Апаратна частина для майбутніх плеєрів Digital Signage будується на основі лише декількох виробників.

1. IAdea

Сайт виробника: http://www.iadea.com/

У лінійці їсти можливість реалізувати все варіанта – від простих одноканальних рішень, до більше складних. Плюс є готові платформи з екраном для побудови інформаційних кіосків. З недавнього часу з'явилося кілька пристроїв на базі вже передвстановленого Android. Останнім часом з'являються SaaS рішення із клієнтами на Android і, видимо, це відповідь на дану тенденцію.

2. AOpen

Сайт виробника: http://www.aopen.com/

По правді говорячи, є все, що й у попереднього бренда. Плеєри різних варіантів, готові моноблоки й інше.

Багато моделей, у тому числі моноблоків, роблять із захистом по IP65, тобто не бояться пилу й вологи. Це важливо, тому що часто апаратну частину заганяють за стелю, ховають у самі вилучені частини. Часте устаткування працює в складних умовах, якщо це кафе або ресторан – вологість, бруд, масло та інше.

Це були два основних виробники.

Є й інші. Коротко їх перелічимо.

3. Foxconn

Сайт виробника: http://www.foxconnchannel.com/

У нього є менш дорогі, чим в IAdea і AOpen рішення, їх можна використовувати, коли бюджет урізаний. Вони не такі функціональні, оскільки Foxconn не робить рішення спеціально під Digital Signage.

4. Advantech

Сайт виробника: http://www.advantech.com/

КБПЗ-2025

Дуже популярний бренд в області промислової автоматизації, але і їхнього рішення також іноді використовують. За ціною не сильно дешевше перших двох брендів, але менш відомий для Digital Signage.

5. OPS

Ну й останнє. Рідко, але іноді проекти будують на базі зв'язування система керування контентом + OPS (технології Open Pluggable Specification). Платформа OPS являє собою апаратні платформи, що вбудовуються. Вбудовуються вони відразу ж у дисплеї, у яких є відповідний слот. Наприклад, в NEC.

OPS рішення виробляються рядом великих виробників, у тому числі, до речі й Advantech, у силу ряду причин поширення в Україні дуже невелике.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – динамічна інтерпретована об’єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Офіційний сайт мови програмування Python https://www.python.org/. Python – багатоцільова мова програмування, яка дозволяє писати код, що добре читається. Відносний лаконізм мови Python дозволяє створити програму, яка буде набагато коротше свого аналога, написаного на іншій мові. Python – багатоплатформова мова програмування. Це означає, що програми на Python можна запускати в різних операційних системах без будь-яких змін.

Ще однією перевагою Python є його стандартна бібліотека, яка встановлюється разом з Python і містить готові інструменти для роботи з операційною системою, веб-сторінками, базами даних, різними форматами даних, для побудови графічного інтерфейсу програм тощо. Програми, написані на мові програмування Python, можуть бути як невеликими скриптами, так і складними системами. Python абсолютно безкоштовний.

КБПЗ_2025

Швидкість виконання коду Python

Один з можливих недоліків Python – швидкість виконання коду. Python не є компільованою мовою. Код на Python спочатку компілюється у внутрішній байт-код, який потім виконується інтерпретатором Python. У більшості випадків при використанні Python виходять програми повільніші в порівнянні з такими мовами, як C.

Втім, сучасні комп'ютери мають таку обчислювальну потужність, що для більшості застосунків швидкість розробки важливіша швидкості виконання, а програми на Python зазвичай пишуться набагато швидше.

Окрім того, Python легко розширюється модулями, написаними на C або C++. Такі модулі можуть використовуватися для виконання частин програми, що створюють інтенсивне навантаження на процесор.

Використання Python

Python використовується для різних цілей: для створення ігор і веб-застосунків, розробки внутрішніх інструментів для різноманітніх проектів. Мова також широко застосовується в науковій області для досліджень і розв'язування прикладних завдань.

Застосування мови програмування Python:

1. BitTorrent – протокол для обміну даними.

2. Ubuntu Software Center – вільне програмне забезпечення для пошуку, установки і видалення пакунків в системі Ubuntu Linux.

3. Blender – програма для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, вимальовування, пост-обробки відео, а також створення відеоігор.

4. GIMP – растровий графічний редактор, із підтримкою векторної графіки.

5. World of Tanks.

6. Вільна енциклопедія Вікіпедія.

7. Пошукова система Google.

КБПЗ_2025

8. DropBox – файловий хостинг, що включає персональне хмарне сховище, синхронізацію файлів і програму-клієнт.

9. YouTube – популярне відеосховище.

Версії Python

Мови програмування з часом змінюються – розробники додають в них нові можливості, а також виправляють помилки. Так з'являються різні версії мови. Наприклад, код написаний на Python 2 у більшості випадків не буде працювати у версії Python 3 без внесення додаткових змін.

Процесор є найважливішим компонентом в комп'ютері. Одна з основних функцій процесора – це обробка даних згідно комп'ютерної програми, яка є списком інструкцій, шляхом виконання арифметичних і логічних операцій над фрагментами даних.

Кожна інструкція в програмі – це команда, яка «повідомляє» процесору, яку операцію він повинен виконати. Процесор комп'ютера може розуміти лише ті інструкції, які написані на машинній мові. Машинна мова – це штучна мова, створена для передачі команд комп'ютеру. За допомогою машинної мови створюються ефективні програми, оскільки розробник отримує доступ до всіх можливостей процесора. Машинна мова – мова низького рівня.

Інструкція машинної мови існує для кожної операції, яку процесор здатний виконати – є інструкція для додавання чисел, є інструкція для віднімання чисел і т.д. Увесь набір інструкцій, який центральний процесор може виконати, відомий як набір інструкцій процесора.

Наприклад, у вас є певна програма, яка зберігається на диску вашого комп'ютера. Для виконання програми, ви здійснюєте подвійний клік на значку програми. Це змушує програму копіюватися з диска в оперативну пам'ять, після чого процесор комп'ютера виконує копію програми, яка знаходиться в оперативній пам'яті.

Коли процесор виконує інструкції програми, він бере участь у процесі, який є відомим як цикл fetch – decode – execute (отримати – декодувати –

КБПЗ_2025

виконати). Цей цикл виконується для кожної інструкції у програмі і складається з трьох кроків:

Отримати

Програма – це послідовність інструкцій на машинній мові. Першим кроком циклу є завантаження (отримання) наступної інструкції з пам'яті в процесор.

Декодувати

Інструкція машинної мови – це двійкове число, яке представляє команду, що повідомляє процесору виконати певну операцію. На цьому кроці процесор декодує інструкцію, яку було «витягнуто» з пам'яті, для визначення того, яка операція повинна виконуватись.

Виконати

Останній крок циклу – виконати операцію.

Хоча процесор комп'ютера розуміє тільки машинну мову, людині непрактично писати програми на машинній мові. Така програма може мати тисячі або навіть мільйони бінарних інструкцій, і написання такої програми буде дуже обтяжливим процесом.

З цієї причини була створена мова асемблера як альтернатива машинній мові. Замість використання двійкових чисел для написання інструкцій, мова асемблера використовує короткі слова, відомі як мнемокоди.

Незважаючи на те, що мова асемблера не вимагає двійкових інструкцій, як у випадку машинної мови, проте вона вимагає високих знань про процесор. Використовуючи мову асемблера, навіть для найпростішої програми, необхідно написати велику кількість інструкцій.

Мова програмування високого рівня дозволяє створювати складні програми, не знаючи, як працює процесор, і не записуючи великої кількості інструкцій низького рівня. Крім того, більшість мов програмування високого рівня використовують слова, які легко зрозуміти.

КБПЗ_2025

Python – одна із популярних сучасних мов програмування високого рівня. Python – інтерпретована мова програмування. Python – це високорівнева інтерпретована мова програмування, на відміну від C++, яка є прикладом компільованої мови програмування. Назва Python відноситься як до мови програмування, так і до інтерпретатора – комп'ютерної програми, яка зчитує початковий код (написаний на Python) і виконує інструкції (команди).

Для перекладу мови високого рівня на машинну мову доступні два типи програм:

1. Компілятор.
2. Інтерпретатор.

Завантаження Python

Версії інтерпретатора Python для різних операційних систем доступні для безкоштовного завантаження за адресою https://www.python.org/downloads.

Середовище програмування для Python

Для написання програм використовують текстові редактори або інтегровані середовища розробки, які включають в себе різні інструменти для роботи з кодом: засіб для написання коду (текстовий редактор), інтерактивний інтерпретатор, відлагоджувач тощо.

Текстові редактори та інтегровані середовища програмування для Python:

– IDLE – стандартний редактор Python. Встановлюється разом з Python для користувачів Windows, окремим пакунком для користувачів Linux.

– Notepad++ – безкоштовний текстовий редактор початкового коду, який підтримує велику кількість мов, в тому числі і Python. Лише для користувачів Windows.

– Visual Studio Code – це легкий, але потужний редактор початкового коду, який розповсюджується безкоштовно і доступний у версіях для платформ Linux, Windows і macOS.

– PyScripter – інтегроване середовище розробки для мови програмування Python. Для користувачів Windows. Поширюється безкоштовно.

КБПЗ_2025

– Wing IDE 101 – вільне інтегроване середовище для Python, розроблене для навчання програмістів-початківців. Для користувачів Linux, Windows і macOS. Поширюється безкоштовно.

– Geany – вільний текстовий редактор з базовими елементами інтегрованого середовища розробки, доступний для операційних систем Linux, Windows і macOS.

– PyCharm – інтегроване середовище розробки для мови програмування Python. PyCharm є власницьким програмним забезпеченням. Наявна безкоштовна версія Community з усіченим набором можливостей. Для користувачів Linux, Windows і macOS.

– Thonny – IDE для вивчення програмування мовою Python. Для користувачів Linux, Windows і macOS.

– Mu – редактор коду Python для програмістів-початківців. Для користувачів Linux, Windows і macOS.

КБПЗ_2025

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи Digital Signage для реалізації корпоративного телебачення (IP TV).

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати

поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

К6ПЗ_2025

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Тенденцію росту інтересу до інтеграції Digital Signage з іншими IT-системами проілюструємо на прикладі проекту, реалізованого у флагманському магазині, де єдина система Digital Signage на базі PADS4 інтегрована із внутрішньою системою SAP. Крім звичних рекламних і промороликів у торговельних залах, на екранах Digital Signage відображаються черга й статуси замовлень, які клієнти оформляють самостійно через термінали. Крім того, у системі передбачено багато інших можливостей, таких як відображення рекомендованих аксесуарів, що формується автоматично по поточному замовленню.

Інший проект, що ілюструє цю тенденцію, реалізований для одного з аеропортів (у проекті використані ПЗ PADS4 і панелі Samsung з убудованими плеєрами). Це складне рішення, інтегроване з іншими системами: на панелях відображаються відео, графічна й текстова інформація, що міняється автоматично залежно від поточних рейсів і інших відомостей, що витягаються з бази даних SQL.

За рівнем розвитку ринку систем Digital Signage Україна відстає від США і Європи на 8-10 років. «Останні роки ми спостерігаємо збільшення попиту на технології Digital Signage, характерне для початку XXI століття на Заході. У минулому році наші замовники – як інтегратори, так і кінцеві користувачі – цікавилися системами Digital Signage набагато більше, ніж в 2023-му.

Основними споживачами рішень Digital Signage стали музеї, транспортні компанії, банки. Магазини, активні покупці технологій Digital Signage у минулі роки, в 2024-м небагато зменшили темпи. Дане явище зв'язуємо насамперед із занепадом самого рітейла: рік був не самий удалий через зменшення споживчого

КБПЗ-2025

попиту. При цьому нові магазини міжнародних брендів, які відкриваються в Києві, однаково прагнуть до використання Digital Signage у тім або іншому форматі.

Системи Digital Signage одержують поширення не тільки в сфері реклами, але й для інформування людей у таких галузях, як спорт, охорона здоров'я, утворення. Компанії також стали активніше використовувати Digital Signage для своїх внутрішніх потреб (інформування співробітників і гостей про внутрікорпоративну й іншу важливу події).

Перераховуючи галузі, де останнім часом активно впроваджуються системи Digital Signage, крім банківського й медичного секторів, назвемо автомобільний. Наприклад, компанія KIA ухвалила рішення щодо того, що вся дилерська мережа повинна бути оснащена системами Digital Signage, а не тільки салони в містах-мільйониках.

Багато експертів відзначають, що все більша частина запитів замовників, пов'язаних з Digital Signage, стосується впровадження корпоративного телебачення (IP TV). Даний IT-інструмент гарний тим, що дозволяє вибудувати ефективний канал комунікації усередині компанії – оперативно й одноразово транслювати, наприклад, у територіально розподілених офісах актуальну інформацію. Малоймовірно, що всі співробітники щодня вивчають новини на сайті своєї компанії або із завидною регулярністю читають внутрішні бюлетені, а от від екранів з Digital Signage нікуди не дінешся, якщо вони грамотно розташовані в місцях підвищеної прохідності – там, де ока легко «зачепитися за контент», – на вході в офіс, у ліфтових холах і коридорах, у переговірних кімнатах, у їдальнях і кафі. Більше того, переклад корпоративних комунікацій у цифровий формат дозволяє істотно заощадити на паперових носіях: листівках, плакатах, флаерах та ін.

Щоб донести актуальну інформацію до співробітників, служби внутрішніх комунікацій використовують Digital Signage, у тому числі для показу персоналізованої інформації. Масово впроваджується автоматизація подання

КБПЗ_2025

бізнес-показників на екранах: KPI, продуктивність зміни, виконання плану, показники виробітку щодо інших змін, кращі працівники, дні народження й т.д.

Засоби відображення

На різних виставках уже широко демонструються дисплеї з дозволом 8K (порядку 8 тис. пікселів по горизонталі). Однак висока вартість поки залишається непереборною перешкодою на шляху впровадження таких продуктів. Разом з тим ціновий бар'єр для дисплеїв 4K (4 тис. пікселів по горизонталі), схоже, уже переборений, і вони користуються все більшою популярністю. Коли такі дисплеї з'явилися на ринку, їхня ціна теж була дуже високої, однак за минулі два роки вони істотно подешевіли, тоді як число й розмаїтість пропозицій на ринку виросли. Власне кажучи, в області побутової техніки телевізори Ultra HD (з дозволом 4K) уже домінують на полках у магазинах. Очевидно, що техніку з таким дозволом усе активніше будуть використовувати й у професійних проектах Digital Signage, особливо в ті з них, де важливе якість зображення.

Згідно із прогнозом, протягом двох-трьох років 4K повністю витисне формат HD. У цілому, говорячи про розвиток засобів відображення, експерт із IMS відзначає підвищення яскравості, збільшення діагоналі (у рамках одного із проектів фахівцями IMS недавно був установлений екран 98”), звуження рамок дисплеїв (стандартом міжпанельного шва сьогодні вважається 1,8 мм). Частка сенсорних панелей у всьому різноманітті дисплеїв становить приблизно 10-20%, при цьому вони стають дешевше, продуктивніше й надійніше.

Дисплеї 3D також активно вдосконалюються, і на ринку вже кілька років присутні автостереоскопічні екрани, тривимірне зображення на які можна бачити без спеціальних окулярів. Однак у реальних проектах вони використовуються вкрай рідко. Головні проблеми – труднощів створення контенту 3D і висока вартість рішень. Проте при усуненні ціх перешкод 3 D-системи мають непогану перспективу для застосування в системах Digital Signage.

В останні роки досягнуть великий прогрес у технологіях світлодіодних (LED) екранів: істотно зменшений крок пікселя – до 0,9 мм для плоских і до 1,5

КБПЗ_2025

мм для вигнутих дисплеїв. При цьому вартість рішень також знижується. Тут відразу необхідно внести пояснення: у побутовій техніці LED-дисплеями часто називають звичайні РК-дисплеї з LED-підсвічуванням. Коректніше вживати термін «LED-екран» відносно пристроїв відображення, у яких світлодіоди використовуються в якості пікселів для передачі візуальної інформації. Саме в такому змісті цей термін уживається в даній статті.

Переваг у світлодіодних екранів багато: більше висока яскравість, мала товщина (на ринку представлені моделі глибиною кілька міліметрів), енергоефективність, тривалий термін служби й т.д. Такі екрани можуть бути різного розміру й форми, у тому числі більших розмірів (до сотень метрів завширшки й висоту), а також з нестандартним співвідношенням ширина/висота. Оскільки яскравість окремих світлодіодів можна індивідуально регулювати, що відповідають екрани мають більше високий рівень контрастності. Ще один важливий аспект – гарна ремонтопридатність: навіть при ушкодженні частини пікселів екран можна нормально використовувати, а при можливості – замінити тільки ушкоджена ділянка.

Не дивно, що експерти високо оцінюють перспективи усе більше широкого застосування LED у системах Digital Signage. Світлодіодні екрани захоплюють ринок, міняючи підхід до оформлення простору. Екрани стають гнучкими й можуть приймати практично будь-яку форму й вид: куля, куб, зиґзаґ. Нестандартні гнучкі поверхні особливо актуальні для рітейла, для створення вау-ефекту в боротьбі за увагу покупців.

Експерти також високо оцінюють потенціал росту попиту на відеостіни. До таким перевіреного роками перевагам ціх рішень, як ефектність і ефективність, додалися також простота й доступність технологій організації відеостін. Навіть незважаючи на високу ціну, відеостіна, що складається з декількох засобів відображення, звичайно, обійдеться дешевше, ніж дисплей розміром зі стіну, і при цьому пропонує більше гнучкі можливості по відтворенню контенту.

КБПЗ_2025

Відеостіни стають обов'язковим (must have) інструментом для телестудій, ситуаційних центрів і багатьох інших областей. Установлені на ресепшен, вони є ефектним рішенням для інформування співробітників і ділових партнерів, зміцнення власного бренда, створення унікальної атмосфери. Промислові підприємства також звертаються за відеостінами – для них цей зручний і наочний засіб відображення інженерних схем.

Коли відеостіни з'явилися в 80-х роках минулого століття, серйозні технічні складності представляла синхронізація контенту, відтвореного на складову стіну дисплеях. По суті, було два варіанти побудови відеостіни:

– Перший – використання дуже дорогих тоді комп'ютерів, здатних розділяти й синхронізувати контент.

– Другий – попередній поділ контенту й відтворення його на окремих плеєрах, кожний з яких підключався до конкретного дисплея. Обидва варіанти – складні й дорогі.

Сьогодні, у тому числі завдяки здешевленню й підвищенню потужності обчислювальної техніки, поділ контенту між безліччю складову відеостіну дисплеїв – завдання нескладна. Існує безліч варіантів побудови відеостін, у тому числі цілком бюджетні. Так, як контролер можна використовувати звичайний сервер, побудований на архітектурі Intel, але з додатковими платами захвата й виводу зображення. Більше надійне й продуктивне (але й більше дороге) рішення – спеціалізований апаратно-програмний комплекс. Крім централізованого контролера, можна використовувати й розподілену архітектуру, коли кожному дисплею виділяються по одному пристрої захвата й плеєру. При цьому для зв'язку між компонентами можна задіяти звичайну ЛОМ, що дозволяє знизити загальну вартість проекту.

Широкий і вибір варіантів для відображення інформації на відеостіні. Мабуть, найбільшу популярність придбали відеостіни з LCD-панелей, чому в чималому ступені сприяло зниження їхньої вартості. Інший варіант – відеостіни із проекційних відеокубів, які звичайно застосовують у тих ситуаціях, коли до системи відображення пред'являються найвищі вимоги. Перспективним є й побудова відеостін зі світлодіодних (LED) екранів.

КбПЗ_2025

3.2 Розробка структурної схеми

Хмарні моделі

Які є можливості економії при створенні систем Digital Signage? Самі очевидні – це пошук більше дешевих продуктів, у першу чергу в каталогах китайських виробників, а також урізування функціональності рішення й відмова від «надмірностей» (див. урізання «Як можна заощадити при створенні системи Digital Signage»). Ще одна можливість скоротити витрати – використання вже згаданих вище інтегрованих плеєрів класу «система на чипі» (System on Chip, So). У випадку побудови відеостін можливість економії забезпечать уже згадані контролери на основі стандартних серверів.

Як можна заощадити при створенні системи Digital Signage:

– Знайти більше дешеві продукти й/або урізати функціональність.

– Використовувати плеєри, інтегровані в дисплеї.

– Для відеостін застосовувати контролери на основі стандартних серверів.

– Задіяти хмарні моделі.

– Для доставки відео на об'єкті використовувати стандартну ЛОМ.

Істотно знизити капітальні витрати при організації системи Digital Signage допоможе застосування хмарних моделей, добре відомих у світі IT. Тут можуть бути використані різні варіанти. Наприклад, замість власних систем зберігання контенту можна задіяти хмарне сховище по моделі «інфраструктура як сервіс» (IaaS). Це дозволить без праці масштабувати обсяг збережених даних, платячи тільки за ті ресурси СЗД, які реально необхідні в цей момент. Віртуальну інфраструктуру IaaS можна задіяти й для розгортання ПЗ керування контентом. Інший варіант – використання із хмари системи керування контентом по моделі «ПЗ як сервіс» (SaaS). При переході на хмарні моделі необхідно взяти до уваги безліч факторів, включаючи питання безпеки і якості каналів зв'язку з Інтернетом. Крім того, варто пам'ятати, що зекономлені капітальні інвестиції (CAPEX) будуть витрачатися при експлуатації систем (OPEX), тому важливо

КБПЗ - 2025

зрівняти загальну вартість володіння (TCO) виходячи з розрахункового обрію планування бізнес-моделі використання системи Digital Signage.

Відзначимо зрослий інтерес замовників до хмарних сервісів Digital Signage, коли, наприклад, вони встановлюють у себе засобу відображення інформації, а послугу керування контентом одержують по підписці, тобто за щомісячну абонентську плату. Це зручніше й дешевше в короткостроковій перспективі, уважає фахівець інтегратора.

Опишемо наступні можливі сценарії застосування різних рішень по керуванню контентом, включаючи хмарні. Якщо завдання – оснастити один магазин, то при бажанні досить придбати один плеєр і просто міняти SD-карти, при цьому мережа ніяк не задіється. Якщо плеєрів уже з десяток і вони перебувають в одному ТЦ, контентом можна управляти через локальну мережу. Якщо в 20 магазинах по 10 плеєрів, то зручніше використовувати хмару, оскільки так простіше систематизувати пристрою (розподілити їх по групах: наприклад, у групу «Магазин 1» увійдуть десять плеєрів, у групу «Магазин 2» – 3 плеєра й т.д.) і здійснювати моніторинг їхньої роботи. «При цьому якщо плеєрів менш 30, то вигідніше оформити підписку, а якщо їх 30 і більше, те має сенс придбати хмарне ПЗ, установити його на свій сервер і більше ніколи не платити за підписку, як надійшли великі компанії.

Але хмарні моделі в цілому в Україні поки не дуже популярні. На сучасний момент всі наші клієнти воліють здобувати програмне забезпечення для керування контентом у власність. Для банків хмарні рішення не підходять по причинам, пов'язаним з безпекою, тоді як великі компанії воліють мати продукт у себе на балансі й не залежати від зовнішнього контрагента.

Мережна інфраструктура

Ще один потенційний спосіб економії – використовувати для зв'язку між елементами системи Digital Signage звичайну ЛОМ, що може бути вже розгорнута на об'єкті. Відзначимо тенденцію активного перекладу засобів трансляції контенту в системах Digital Signage на мережні технології. При цьому,

КБПЗ_2025

по його досвіду спілкування із клієнтами, навіть фахівці не завжди розуміють різницю між різними рішеннями, застосовуваними в системах трансляції відеосигналу. Найчастіше розподіл контенту від джерела до пристроїв відображення по кручений парі сприймається як мережне рішення з передачею по IP, хоча насправді тут можлива безліч різних варіантів.

Виділимо три основних підходи до передачі сигналів по «кручений парі»:

– Перший – аналогова передача з використанням кабелів із крученими парами як фізичного носія. У цьому випадку задіється тільки фізичне середовище (СКС), але ніяк не технологія Ethernet. Переваги аналогової передачі – у її низької вартості при досить великій дальності: можлива доставка відео на відстані до 300 м, але при збільшенні дальності підтримуваний дозвіл відео, звичайно, знижується. Недоліків у цього варіанта набагато більше: це й чутливість до зовнішніх перешкод (при використанні неекранованої (UTP) проводки), і можливість передачі по одному кабелі тільки одного АВ-потоку, і підтримка тільки топології «точка – точка» (для підключення двох і більше моніторів потрібні повторювачі сигналу) і т.д.

– Другий підхід – використання технології HDBase-T. Це вже цифрова передача відео по проводці UTP/FTP. Дана технологія реалізується, якщо так можна виразитися, «паралельно» Ethernet. Справа в тому, що при використанні HDBase-T, крім незжатого відео високого дозволу, по тій же кабельній проводці можна передавати інші сигнали, включаючи: аудіо; кадри Ethernet зі швидкістю 100 Мбіт/с; керуючі сигнали RS-232 і інфрачервоні; USB 2.0, а також електроживлення (до 100 Вт).HDBase-T також функціонує по каналах «точка – точка» і через стандартні комутатори Ethernet «не проходить».

– Третій підхід – це властиво трансляція по мережі Ethernet. Тут також можливі різні варіанти. Одними з перших трансляцію АВ-сигналів по IP запропонували виробники IP KVM. У цьому випадку використовується стандартна мережа Gigabit Ethernet з досить широкою смугою пропущення. Однак системи KVM орієнтовані на передачу статичних зображень і

КБПЗ - 2025

забезпечення гарантованого каналу керування, а тому вони не дуже добре підходять для доставки динамічної картинки.

Інший варіант – використання пропрієтарних рішень, спеціально розроблених для доставки відео. Як приклад такого рішення Ігор Калінін приводить систему TNTv. Це рішення, що працює по локальній мережі Gigabit Ethernet, підтримує практично необмежена кількість джерел і пристроїв відображення, довільну комутацію сигналів, а також транскодовані відеопотоку (скажемо, інтерфейс джерела сигналу може бути аналоговим, а пристрою відображення – цифровим, або навпаки). Головний недолік такого рішення – пропрієтарний (закритий) протокол передачі даних.

Існує й стандартизований варіант рішення по передачі мультимедіа по мережі Ethernet. Це технологія AVB (Audio Video Bridging), про яку докладно розказане в статті. Для доставки відео по локальній мережі можна використовувати й стандартні технології H.264 і H.265, оптимізовані для територіально розподілених мереж (WAN) і працюючі поверх IP. Кодеки H.264 і H.265 розроблялися в першу чергу для не самих швидкісних WAN-каналів, тому в них реалізоване сильний стиск відео, на відміну від рішень для локальних мереж, у яких відео часто передається взагалі незжатим.

Технологія AVB (Audio Video Bridging)

Audio Video Bridging дозволяє усунути потенційні проблеми при перекладі професійного АВ-контенту в мережу Ethernet.

Процес переходу на комп'ютерні технології торкнути практично весь медіаринок. За останні кілька десятиліть від традиційної телефонії майже нічого не залишилося. Лише в деяких будинках ще збереглася проводка у вигляді «локшини», але на всіх магістральних каналах провайдери якщо не перейшли, то активно здійснюють перехід на VoIP. Аналогічний процес спостерігається й у сфері телебачення – всі частіше ця послуга надається як IP-TV через Інтернет, по мережах Ethernet і IP, що відкриває доступ до більше різноманітного контенту.

КБПЗ_2025

Те ж відбувається й з відеоспостереженням. Колись для цього будувалися окремі мережі й інфраструктура. Тепер повсюдно використовуються IP-відеокамери, а сигнал від них передається по комп'ютерній мережі на записуючий і відтворюючий пристрій. Чому ж настільки популярний перехід із пропрієтарних, закритих технологій на Ethernet? Причина в уніфікації. А що дає уніфікація? Відповідь проста – економію.

Одна перспективна область, де перехід на використання комп'ютерної мережі ще не відбувся, – це професійні аудіовідеосистеми. У них застосовуються досить закриті технології, власні протоколи й, природно, спеціалізована інфраструктура. Крім студій звукозапису, ці системи розгортаються на більших об'єктах, де необхідно встановити устаткування для звукового оповіщення, доставки контенту, показу відео. Мова йде про стадіони, концертні комплекси, конференц-зали, парках, готелях, аеропортах і студіях віщання.

Професійні аудіовідеосистеми – це зовсім особливий мир. Мало хто з фахівців з IT розбирається в AB настільки ж добре, як, наприклад, у мережах. У ціх рішень досить багато нюансів. Перша й сама головний – якість доставки аудіо- і відеоконтенту. Технологія Audio Video Bridging (AVB) дозволяє усунути потенційні проблеми, які виникають при перекладі професійного AB-контенту в мережу Ethernet: забезпечити синхронізацію, резервування смуги, рівномірний потік даних і т.д. По оцінках аналітиків, у цього ринку великий потенціал – близько 2 млрд доларів (при поточному обсязі 100 млн доларів).

Особливості аудіо-відео

Професійні аудіовідеосистеми (Audio Video System, AVS) повинні забезпечувати студійна якість звуку й відео, при цьому вони висувають високі вимоги до рівня затримки, що не повинна перевищувати 2 мс. Найчастіше використовується кодек без стиску, для кожного каналу виділяється окремий кабель, з'єднання організується за принципом «точка – точка», зв'язок однобічна, напівдуплексна. При цьому процес комутації підключень досить трудомісткий,

КБПЗ_2025

тому що кожний виробник реалізує в устаткуванні, що випускається їм, свої рознімання й свої стандарти передачі (SDI для відео й S/PDIF для звуку).

У результаті при побудові систем AVS на базі традиційних технологій доводиться використовувати велику кількість пристроїв. Всі вони зв'язані між собою безліччю кабелів, а для комутації потрібно досить складне й дороге устаткування. Причому це постійна комутація, тобто з того самого мікрофона звук виводиться на ті ж самі колонки – устаткування не можна набудовувати динамічно.

Як зміниться картина при використанні мережі Ethernet? Кількість необхідних кабелів і устаткування різко скоротиться. Так що ж заважає перевести АВ-сигнал у мережу IP? Адже давно вже існують технології передачі аудіовідеоконтенту по мережі: VoIP, IP-TV, різні методи кодування, оцифровки й розбивки на кадри. Однак не можна просто взяти Ethernet-комутатори й поставити їх у мережу АВ: стандартна технологія Ethernet не дозволяє забезпечити ні припустиму (не більше 2 мс) затримку при проходженні сигналу через всю мережу, ні синхронізацію за часом і рівномірний потік даних, ні гарантовану пропускну здатність і автоматичне виявлення пристроїв. Саме для рішення ціх проблем, без чого неможливо перевести весь АВ-трафік на Ethernet, і був розроблений стандарт AVB.

Основні стандарти й терміни

Насправді AVB – це ціла група стандартів IEEE:

– 802.1BA описує загальну архітектуру системи AVB, у тому числі автоматичне виявлення й завдання параметрів пристроїв;

– 802.1AS відповідає за синхронізацію;

– 801.Qav розширює 802.1Q у частині обслуговування чутливих вчасно потоків;

– 801.Qat доповнює 802.1Q протоколом резервування потоків.

Для початку познайомимося з використовуваними термінами. Всі кінцеві пристрої підрозділяються на Talker, Listener і Endpoint. Talker, або мовний

КБПЗ_2025

пристрій, передає контент у мережу. Listener, або слухач, цей контент одержує, або слухає. Endpoint, або властиво кінцева точка, являє собою будь-який пристрій, що може приймати й/або передавати трафік AVB. Таким чином, кінцева точка може бути як передавачем, так і приймачем або обома одночасно. Крім цього, уводиться поняття потоку, під яким мається на увазі потік медіаданних від передавача й до приймача. Стандарт AVB передбачає можливість віщання з одного мікрофона на кілька відтворюючих пристроїв.

Як приклад розглянемо просту мережу, до якої підключено два джерела звуку (гітара й барабани) і два споживачі звуку (2 колонки). Будь-які музичні пристрої – і гітара, і барабани – видають аналогові сигнали. Щоб аналоговий сигнал потрапив у мережу, його потрібно перевести в цифровий вид. А для передачі оцифрованого сигналу по мережі спеціальні пристрої – експандери й конвертери – повинні інкапсулювати його до протоколу AVB. Відповідно, кожний аналоговий пристрій постачений експандером, що саме й конвертує звук у цифровий вид. На виході встановлені інші два експандера, що конвертують цифровий звук назад в аналоговий формат.

Щоб експандери «знали», на який кінцевий пристрій направляти сигнал, необхідний сервер керування, у даному прикладі це BIAMP Tesira. З його допомогою оператор системи АВ робить налаштування підключених до мережі пристроїв. Крім того, система керування відповідає за виявлення в мережі підключених пристроїв.

Синхронізація

У великому приміщенні, наприклад у концертному залі, колонки, що відтворюють звук, перебувають у різних його частинах, і тому виникає проблема синхронізації. Необхідно зробити так, щоб звук у ці колонки надходив у те саме час, інакше та, що перебуває ближче до джерела, буде відтворювати його раніше, ніж розташована, скажемо, за трьома комутаторами, кожний з яких вносить певну затримку при проходженні сигналу. Одночасність відтворення забезпечує протокол синхронізації.

КБПЗ_2025

Як він працює? По-перше, всі пристрої обмінюються певними пакетами, перевіряючи, що кожне з них підтримує протокол 802.1AS (це стосується всіх пристроїв, сертифікованих по стандарті AVB). Далі, автоматично або шляхом налаштування вручну, у мережі вибирається так званий гранд-майстер, що стає єдиним і самим головним еталоном часу в мережі. Це може бути будь-який мережний пристрій, у цьому випадку ми вибрали Biamp Tesira. Воно може «брати» час як з мережі (з будь-якого мережного пристрою по протоколі NTP), так і від якогось зовнішнього джерела (наприклад, від установлених рядом атомних годинників).

Цей гранд-майстер повідомляє всі інші пристрої, скільки зараз часу, при цьому протокол ураховує, що пакети, що йдуть до конкретного експандера, передаються по конкретному шляху. Кожний пристрій AVB на шляху проходження пакета записує в пакеті 802.1AS внесену їм затримку: умовно говорячи, комутатор указує, що на вхідному інтерфейсі 100 Мбіт/с така-те затримка, при проходженні трафіку між двома його портами затримка така-те, на вихідному гігабітном інтерфейсі, відповідно, затримка таке-те. Всі затримки підсумуються й відбиваються в пакеті 802.1AS. Таким чином, у пакеті, що надійшов на кінцевий пристрій, утримується фактична затримка при передачі по мережі.

Як бачимо, пристрою не тільки синхронізують годинники й одержують інформацію про точний час, але й визначають затримку, внесену будь-якими мережними вузлами. Знаючи точний час і величину затримки на кожному етапі, можна розрахувати так званий презентаційний час, коли повинен бути відтворений сигнал на всіх одержувачах. Воно визначається виходячи із самого гіршого варіанта проходження сигналу від джерела до пристрою відтворення. Відповідно, звук з гітари відправляється на далеку колонку раніше, ніж на ближню, тому на обидві колонки він приходить одночасно.

Це одне із самих головних досягнень групи стандартів AVB: забезпечення синхронізації за часом у мережі Ethernet при передбачуваних затримках, завдяки чому досягається синхронне відтворення звуку й відео.

КБПЗ_2025

Резервування потоків

Що відбувається після визначення затримок у мережі? Коли оператор АВ-системи вказує, що звук від гітари й барабанів повинен надходити на обидві колонки, сервер керування інформує відповідні експандери про те, що їм потрібно передавати звук такому-те й такому-то кінцевому устаткуванню (його експандерам). Потім у дію вступає протокол резервування потоків (Stream Reservation Protocol, SRP).

Відповідно до протоколу, кожне джерело звуку надсилає запит, що називається Talker Advertise – дослівно «оголошення мовця». У відповідь експандери, яким ці потоки призначаються, відправляють повідомлення про готовність Listener ready. Воно знову проходить весь шлях по мережі до первісних експандерів, звідки береться звук. Мережні комутатори відслідковують всі чинені дії й при зворотному проходженні пакетів з інформацією про готовність слухачів кожний з них резервує необхідну пропускну здатність для даного конкретного потоку.

Це одне з найважливіших додавань у стандарт, тому що воно дозволяє гарантувати смугу для трафіку. Причому мова йде не про гарантію засобами Qo, а про тверде виділення необхідної пропускної здатності. До 75% пропускної здатності інтерфейсу може бути віддане під AVB, а 25% залишено для іншого трафіку, щоб AVB не займав весь канал.

Якщо смуги не вистачає, наприклад, комутатор уже й так обслуговує досить багато потоків AVB на певному інтерфейсі, то цей комутатор пошле спеціальне повідомлення в обидва боки про неможливість організувати потік.

У даному прикладі аудіопотоки йдуть прямо від джерел до пристроїв відтворення, але на практиці вони, швидше за все, будуть проходити через систему мікшування.

Зіткнення пріоритетів

Якщо який-небудь користувач починає завантажувати з Інтернету файл великого обсягу й завантаження займає значну пропускну здатність якого-небудь

КБПЗ_2025

інтерфейсу, то смугу, виділену для потоків AVB, цей трафік не зможе використовувати. Однак при перегляді, наприклад, потокового відео або у випадку проведення ресурсномісткої телеконференції конкуруючий за смугу трафік теж має високий пріоритет.

Відповідно, комутатор з підтримкою DiffServ буде намагатися відправити й потік AVB, і потік IP-TV якнайшвидше, з тим самим пріоритетом. Відсутність розподілу відправлення пакета в часі може привести до переповнення буферів. Щоб цього не відбувалося, у рамках групи стандартів AVB прийнятий ще один стандарт – FQTSS.

Він забезпечує свого роду формування трафіку (shaping), але не зовсім звичайне – на основі кредитних буферів. FQTSS розподіляє пакети за часом, тобто забезпечує рівномірний, плавний потік пакетів AVB, що виключає виникнення різниці в затримці, дозволяючи уникнути джиттера.

Налаштування AVB на комутаторах

AVB споконвічно створювався таким чином, щоб для його підтримки було потрібно внести лише мінімальні зміни в стандарт Ethernet, у пристрої, що беруть участь в організації мережі Ethernet, і в налаштування устаткування.

Формально, для того щоб настроїти AVB на комутаторах, необхідно виконати налаштування наступних компонентів:

– VLAN;
– протоколу резервування смуги;
– шейпингу;
– синхронізації й т.д.

На перший погляд, це складно, але на комутаторах Extreme Networks протокол AVB налаштовується однією командою enable AVB ports all, за якої ховається багато внутрішніх команд, тобто це якась автоналаштування, що дозволяє комутатору самостійно здійснювати синхронізацію, резервування пропускної здатності й інші дії.

КБПЗ_2025

Щоб комутатор Ethernet міг підтримувати AVB, у ньому повинна бути реалізована апаратна й програмна підтримка. У комутаторах Extreme апаратна підтримка забезпечується встановленими в них чип-сетами, а програмна – операційною системою. У цей час Extreme Networks – єдина компанія, що випускає комутатори корпоративного рівня з підтримкою AVB і одночасно POE і POE+. Для використання AVB ніяких додаткових ліцензій здобувати не потрібно.

POE й POE+ потрібні для живлення експандерів, тому що колонки, мікрофони й інші системи можуть розташовуватися де завгодно. У величезному концертному залі колонка може перебувати досить далеко від розетки, і її підключення до мережі Ethernet через експандер вимагає підведення до нього електрики. Щоб спростити інфраструктуру, у тому числі електричну, для живлення таких пристроїв краще задіяти POE.

У цей час AVB підтримується фіксованими комутаторами серій 440 (комутатори початкового рівня для доступу), 460 (комутатор рівня агрегації) і X-670 (48-портовий комутатор з можливістю установки портів 40 Gigabit Ethernet). Найближчим часом буде реалізована підтримка AVB у комутаторах Black Diamond X-8.

Планування мережі

Що входить до складу готового рішення? Насамперед потрібні комутатори з підтримкою AVB і пристрою для конвертації звуку (для цифрового або аналогового форматів) в Ethernet з підтримкою AVB. При цьому протоколи AVB повинні підтримуватися кожним задіяним пристроєм, тобто потрібна наскрізна підтримка AVB. У кожному разі потрібне грамотне планування всієї мережі.

Залежно від розв'язуваних завдань необхідно розрахувати кількість потоків і необхідну пропускну здатність. Так, на один звуковий канал (моно) потрібно близько 6 Мбіт/с, якщо це незжатий звук. Знаючи кількість потоків, можна визначити, які з'єднання будуть потрібні (100 Мбіт/с, 1, 10, 40 Гбіт/с).

КБПЗ_2025

Якщо, наприклад, у мережі є 100 пристроїв, на яких необхідно відтворювати той самий контент, і на кожний пристрій направляти окремий потік, то ці потоки займуть всю доступну ємність. Однак, задіявши багатоадресне розсилання, можна на один потік підключити навіть 500 пристроїв.

Крім того, мережа Ethernet накладає обмеження на дальність передачі. У випадку мідної проводки вона становить 100 м, а щоб затримка не перевищувала 2 мс, у мережі не повинне бути більше 7 транзитних вузлів (для каналів 100 Мбіт/с). Коли в мережі використовуються гігабітні з'єднання, при тих же 7 транзитних вузлах затримка буде становити менш 1 мс.

Якщо говорити про доставку відео по територіально розподілених мережах, то тут мережі IP уже стали практично безальтернативним варіантом, причому активно використовуються технології багатоадресної передачі (мультикаст) і VPN. Перша дозволяє передавати інформацію необмеженому числу учасників (все навантаження при цьому перекладається із сервера на мережу передачі даних, і тут уже неважливо, яке кількість плеєрів підключена до системи – 10 або 10 000), друга – поєднувати екрани, розташовані в різних географічно вилучених об'єктах, у різні групи по типі трансльованого контенту.

Інтерактивність

Динамічні цифрові вивіски повинні, по задуму, привертати більшу увагу користувачів у порівнянні зі своїми статичними попередниками. Останні роки багато говорять про інтерактивності, тобто про можливість користувача взаємодіяти із системою Digital Signage.

Одна справа – просто організувати віщання з екрана на стіні, а інша справа – спроектувати зручний і гарний інтерфейс для взаємодії з користувачами. Необхідні більше складна інтеграція, продумування автономної роботи (поки дуже актуально для України), забезпечення безпеки й конфіденційності й т.д.

Сучасна інтерактивна система – це не просто РК-панель із сенсорним екраном. Це може бути відеостіна з багатокористувальницьким інтерфейсом, засобу спостереження за положенням глядача (наприклад, частина контенту може

КБПЗ_2025

переміщатися по екранах, супроводжуючи глядача), система керування жестами, а також рішення, що забезпечують взаємодію із глядачами за допомогою їх персональних мобільних пристроїв (смартфонів).

Одним з найбільш перспективних способів взаємодії між системами Digital Signage і користувачами, як уважають багато експертів, повинне стати застосування маячков (Beacon), що використовують технологію Bluetooth з низьким енергоспоживанням (Low Energy, LE). Радіус дії таких маячков становить від декількох сантиметрів до декількох десятків метрів. Маячок фіксує поява мобільного пристрою у своїй «зоні обслуговування» і передає йому (у спеціальний додаток) свій ідентифікатор. Потім додаток звертається до сервера, що відправляє повідомлення із прив'язкою до конкретного маячку. Це можуть бути звичайна реклама або, наприклад, купон на знижку в магазині, до якого наблизився користувач. Якщо додаток на мобільному пристрої виключено, то при наближенні до маячку воно може бути автоматично відкрито.

Такий додаток також може відправляти дані про користувача на сервер. Але, з огляду на потенційні проблеми зі збором особистої інформації, далеко не всі компанії, що використовують технологію маячков, задіють цю можливість. Користувачі не палко бажають ділитися своїми даними, а саме їхнє одержання й аналіз істотно підвищують ефективність рішення.

Серйозна проблема використання технології маячков – висока вартість. Так, окремий маячок обійдеться недорого, однак побудова мережі таких маячков, інтеграція із серверами й додатками можуть виявитися дуже витратним проектом.

Однієї із цікавих тенденцій є вбудовування маячков у медіаплеєри системи Digital Signage. Використання функції BrightBeacon забезпечує двонаправлену передачу інформації між системами Digital Signage і мобільним пристроєм потенційного покупця, що перебуває в зоні дії конкретного плеєра. Наприклад, відвідувач магазина може на екрані вибрати товар, що сподобався йому «зберегти» його опис у себе на телефоні й придбати товар на касі.

КБПЗ_2025

Визначення характеристик користувача

У системах безпеки сьогодні усе активніше використовуються технології розпізнавання осіб і ідентифікації людей. Вони можуть застосовуватися й у торгових центрах. Розпізнавання клієнтів на вході в магазин дозволить індивідуально привітати постійних покупців, пропонуючи їм особливі умови, «обчислювати» небажаних персон із чорного списку, відслідковувати відповідність між переміщеннями користувачів і набором реально зроблених покупок і т.д. Зібрані дані дозволять виявляти зміни в трафіку продажів і споживчих переваг, а також створити більше повну базу даних по купівельних звичках кожного споживача або групи споживачів.

Але в системах Digital Signage, як правило, складні технології розпізнавання осіб і ідентифікації людей не застосовуються. Власникам таких систем часто буває досить визначити вік, підлогу, расову приналежність і інші загальні характеристики користувача, що перебуває в безпосередній близькості від екрана Digital Signage, оснащеного розумною камерою. (Щоб не порушувати недоторканність приватного життя, відомості про конкретних користувачів у таких системах звичайно не зберігаються.) Використання ціх даних дає можливість визначити, наскільки адекватно показуваний на екранах контент відповідає аудиторії, і підлаштовувати цей контент.

Сучасне ПЗ дозволяє досить точно розпізнати підлога, вік, емоції, але сьогодні воно мало затребуване, оскільки поки не знайдена ефективна концепція використання отриманої інформації. Відомості про те, що за місяць магазин відвідали, приміром, 10 тис. жінок і 9 тис. чоловіків, малоінформаційні. Додаткову інформацію (час знаходження відвідувача в залі, «глибину перегляду», виявлені переваги та ін.) поки складно інтерпретувати й використовувати для досягнення бізнес-результатів.

КБПЗ-2025

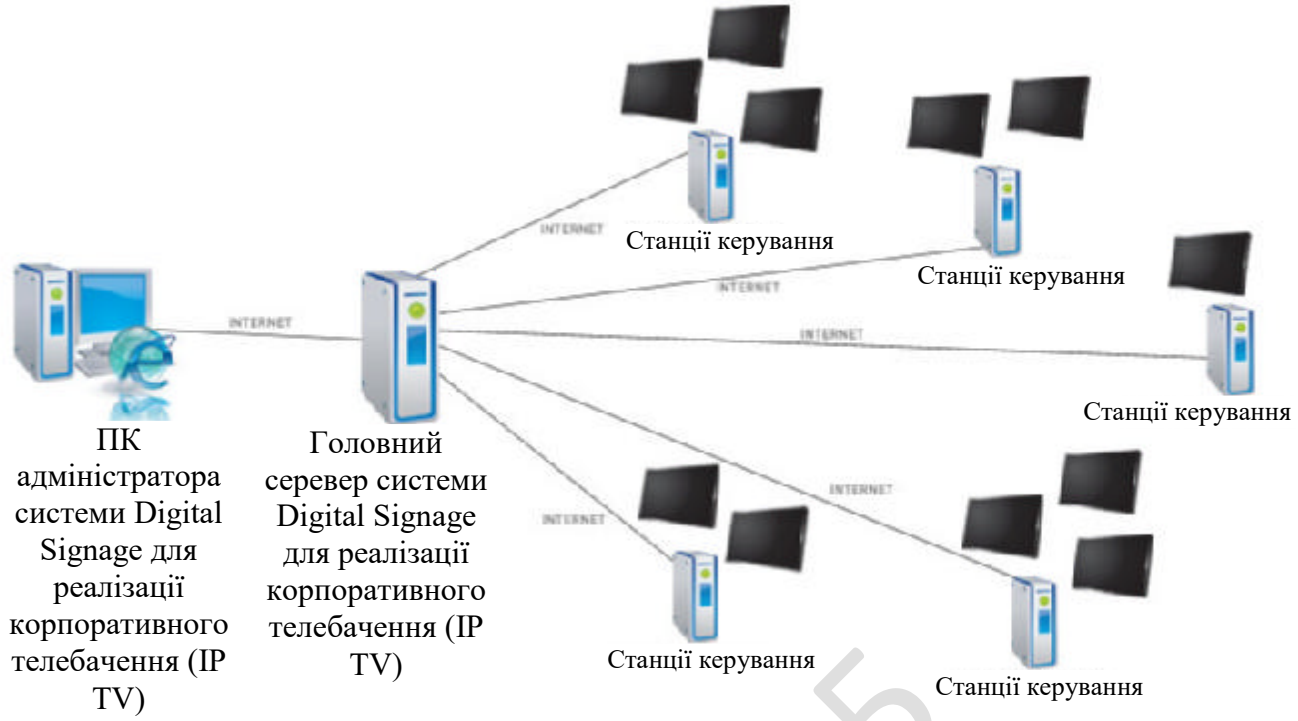


Рисунок 3.1 – Структурна схема системи

Системи відеоаналітики в Digital Signage поки що залишаються більше піаром. Успішних масових проектів з їхнім використанням для Digital Signage я не зустрічав. Зараз ряд компаній активно експериментують із автоматичною ідентифікацією клієнтів. Якщо цей напрямок виявиться вдалим, то наступним етапом буде інтеграція із системами Digital Signage, але не навпаки.

Технології визначення підлоги людини, що коштує перед екраном, уже застосована в деяких проектах – зокрема, для одного великого українського банку. Це дуже перспективний підхід, що уже активно використовується за рубежем, а в Україні тільки набирає оберти. Думаю, що в найближчі кілька років рішення, що дозволяють розпізнавати не тільки підлога, вік, але й настрій відвідувачів, зроблять справжню революцію на ринку.

У цілому системи Digital Signage, що ставилися ніколи винятково до AB-сегмента, стрімко збагачуються сучасними технологіями з «світу IT», такими як аналіз даних (у тому числі Великих Даних), високошвидкісні мережні протоколи, хмарні моделі й т.д. Все це в остаточному підсумку дає замовникам додаткові бізнес-переваги.

КОПЗ_2025

3.3 Розробка функціональної схеми

Digital Signage можливо використовувати також для організації відеоконференцій з використанням корпоративного телебачення (IP TV). Багатоточечний сервер відеоконференцій Digital Signage (ВК) разом із системою керування відеоконференціями є гнучким модульним багатофункціональним засобом для підтримки багатоточечних ВК Digital Signage. Функціональна схема сервера ВК Digital Signage наведена на рисунку 3.2.

Робочі місця учасників процесу відеоконференцій Digital Signage

Загальні вимоги

Під «робочим місцем» розуміється комплект основного, додаткового й супутнього технологічного устаткування, а також інших засобів, розміщених на спеціально виділених площах і необхідних учасникові відеоконференцій Digital Signage для виконання всіх передбачених технологією функцій і процесів. Робочі місця для різних учасників процесу відрізняються друг від друга по составі устаткування і їхньому оформленню залежно від характеру технологічних процесів, виконуваних кожним з учасників. Робоче місце оснащується технологічним устаткуванням і технологічними меблями, засобами висвітлення, електропостачання й життєзабезпечення (опалення, вентиляції й кондиціювання повітря).

Загальні вимоги містять у собі також вимоги до технологічності, ергономіці, електропостачанню, висвітленню, шуму, безпеці, гігієні праці й профілактиці профзахворювань.

Робоче місце оператора сервера відеоконференцій Digital Signage

В склад устаткування оператора сервера ВК Digital Signage входить система резервування й керування відеоконференціями, професійний персональний комп'ютер з модемом і точка підключення до Інтернету.

КБПЗ-2025

Робочі місця користувачів системи відеоконференцій Digital Signage

Види робочих місць

Залежно від видів конференції, учасників і категорії термінального устаткування можуть бути виділені чотири основних види робітників місць.

Індивідуальне робоче місце абонента (користувача) ВК Digital Signage збігається з його постійним робочим місцем в офісі. Групове робоче місце перебуває або в безпосередній близькості від постійного робочого місця одного з учасників ВК Digital Signage, або розміщається в окремо виділеному або пристосованому для цього приміщенні.

Робоче місце оператора ВК Digital Signage територіально розташовується або поблизу групового робочого місця абонента ВК Digital Signage, або на ділянці архіву ВК Digital Signage, залежно від виконуваної роботи. До складу устаткування робочого місця оператора ВК Digital Signage входить комплект устаткування монтажу, кодування відеофонограм і виготовлення архівних копій, а також термінал керування архівом ВК Digital Signage.

Робоче місце оператора сервера ВК Digital Signage розташовується на ділянці зв'язку поблизу сервера ВК Digital Signage і має термінал системи керування відеоконференціями.

Склад устаткування

Робоче місце користувача системи відеоконференцій Digital Signage включає абонентський термінал ВК Digital Signage індивідуального або групового застосування з відеокамерою, мікрофоном, відеотерміналом, гучномовцями й електронними блоками; додаткове устаткування (документальну відеокамеру, комп'ютер, додаткові відеокамеру, мікрофон і відеомагнітофон) і супутнє устаткування (відеопроектор, системи спецосвітлення й звукопідсилення). Залежно від призначення й цілей застосування робочого місця користувача додаткове й супутнє устаткування використовують у різній комплектації.

КБПЗ-2025

Вимоги до розміщення устаткування й людей

Розміщення устаткування й людей – користувачів системи ВК Digital Signage – повинне відповідати ряду суперечливих вимог, частина з яких нормована. По-перше, повинні бути виконані норми МСЕ-Р (сектора Р Міжнародного союзу електрозв'язку) і ЄСВ (Європейського союзу віщання), пропоновані до умов перегляду відеоматеріалу й прослуховуванню звукового матеріалу, по-друге, – технічні вимоги розміщення кожного з видів устаткування, по-третє, враховані норми СНіП у частині електро- і пожаропожежебезпеки. Параметри приміщення й умови розміщення устаткування й людей підлягають розрахунку з урахуванням наступних норм і документів:

– співвідношення сторін приміщення – відповідно до «благозвучного» співвідношеннями Болта [1, 2];

– обсяг приміщення й час реверберації – відповідно до норм МСЕ-Р (МККР) серії BS [3, 4] і вимогами ЄСВ [5];

– мінімальна відстань гучномовців термінала ВК Digital Signage від задньої стіни й розташування людей щодо гучномовців – відповідно до вимог МЕК [6] і МСЕ-Р [3, 7];

– шумові характеристики приміщення – відповідно до норм МСЕ-Р і ЄСВ [2, 3, 7];

– розташування людей щодо екрана відеомонітора – відповідно до вимог рекомендацій МСЕ-Р серії ВТ [8 -10];

– розміщення людей щодо стін приміщення – відповідно до вимог СНіП [11-13], а також з урахуванням вимог, пропонованих до висвітлення тла при відеозйомці.

Вимоги до розмірів приміщення визначаються з урахуванням акустичних вимог до часу реверберації, частотній характеристиці й раннім відбиттям [5, 7].

Оптимальне планування робочого місця досягається на підставі інженерного розрахунку.

КбПЗ_2025

В випадку розміщення учасників відеоконференції в залі використовують додаткове й супутнє устаткування. Основну відеокамеру, установлену звичайно на відеотерміналі, направляють убік доповідача на трибуні, а додаткову – у зал. Відеотерминал розташовують поблизу доповідача й оператора ВК Digital Signage, щоб вони могли стежити за зображенням. Зображення для учасників у залі проектується на екран. Для них у залі встановлюють додаткові мікрофони й передбачають систему звукопідсилення. Для забезпечення правильної передачі кольору в залах використовують систему спецосвітлення.

Особлива увага варто приділяти акустиці приміщень. Акустична обробка поверхонь стін, стелі й підлоги повинна бути такий, щоб запобігти раннім відбиттям звуку, видаваного учасниками й вихідного від гучномовців. Авторові доводилося зіштовхуватися з випадками, коли через зневагу акустичними вимогами помилка в наведенні відеокамери на робочому місці користувача ВК Digital Signage перевищувала 200. При цьому камера наводилася не на промовця, а на його сусіда.

Для проведення групових відеоконференцій Digital Signage важливо ретельно проробити світлотехнічні рішення. Так, колірна температура штучних джерел світла в залі повинна бути однакової щоб уникнути порушення передачі кольору.

Устаткування системи ВК Digital Signage

Можливі схеми з'єднання терміналів ВК Digital Signage

Термінали відеоконференцій Digital Signage з'єднуються один з одним за допомогою каналів зв'язку й коммунікаторів. Для багатоточечної системи відеоконференцій Digital Signage головним коммунікатором є багатоточечний сервер ВК Digital Signage. Залежно від взаємного розташування терміналів і сервера ВК Digital Signage, режимів роботи пристроїв, а також використовуваних мереж зв'язки можливі різноманітні схеми з'єднання терміналів і серверів ВК Digital Signage. При значній взаємній далекості терміналів ВК Digital Signage зв'язок між ними й сервером, а також між ними самими (в окремих

КБПЗ-2025

випадках) здійснюється з використанням каналів магістральних, міжрегіональних і регіональних мереж зв'язку.

У системі відеоконференцій Digital Signage багатоточечний відеоконференцзв’язок може вироблятися по протоколах H.320 і H.323 MCE-T. Зв'язок терміналів BK Digital Signage, що працюють по протоколі H.323, із сервером BK Digital Signage може здійснюватися через вузли (Gateway). З'єднання вузла Gateway із сервером BK Digital Signage виробляється по протоколах BRI і V.35/RS336.

Підключення сервера BK Digital Signage до магістральних, міжрегіональних і регіональних каналів зв'язку може вироблятися через маршрутизатор, магістральний мультиплексор і цифрову ATM. З'єднання сервера із цими пристроями може здійснюватися з використанням протоколів PRI/E1, G.703. Підключення терміналів BK Digital Signage до магістральних, міжрегіональних і регіональних каналів зв'язку може відбуватися через мультиплексори земних станцій (HUB) і абонентських станцій (VSAT) супутникового зв'язку, магістральні мультиплексори (MUX) і ATM. Залежно від відстані між терміналами й цим устаткуванням з'єднання можуть здійснюватися або безпосередньо з використанням протоколу V.35, або із застосуванням модемів або інверсних мультиплексорів.

Деякі особливості роботи підключення терміналів BK Digital Signage до каналоутворюючого устаткування

Безпосереднє підключення термінала BK Digital Signage до каналоутворюючому устаткування допускається в тих випадках, коли довжина сполучних кабелів невелика. У випадках, коли термінали й каналоутворюючое устаткування вилучені друг від друга, для їхнього з'єднання повинні використовуватися модеми або інверсні мультиплексори.

Для входу в супутникову мережу термінали BK Digital Signage повинні бути підключені до мультиплексору центральної, вузлової або абонентської земної станції (ЗС) з використанням інтерфейсної плати внутрішнього модуля ЗС.

КБПЗ_2025

Залежно від відстані між терміналом BK Digital Signage і внутрішнім модулем земної станції підключення виробляється прямо або із застосуванням мультиплексора (модему). З'єднання рекомендується здійснювати з використанням протоколу V.35. Подібні способи підключення мають на увазі застосування твердої структури з'єднань, слабко пристосованої до подальшого розвитку. Вони доречні для використання в кінцеві (не вузлових) пунктах мережі відеоконференцзв’язку.

У тих випадках, коли об'єкт відеоконференцзв’язку розглядається як вузловий об'єкт, підключення термінала BK Digital Signage до внутрішнього модуля ЗС супутникового зв'язку варто здійснювати через мультиплексор.

Термінал BK Digital Signage підключається до інтерфейсному модуля мультиплексора прямо або через модем або інверсний мультиплексор, залежно від відстані між терміналом і мультиплексором.

На вузлових об'єктах з розвитою телефонною мережею доцільно підключати термінали до магістральних мереж через цифрову АТМ. При використанні цифровий АТМ у якості коммунікатора потоків даних відеоконференцзв’язку варто мати на увазі, що ці дані не повинні піддаватися стиску в АТМ, передбаченому для звичайних телефонних сигналів. Інакше кажучи, на період сеансу відеоконференцзв'язку повинна бути як би виділена смуга на вимогу для передачі цифрового потоку з необхідною бітною швидкістю (у загальному випадку 256 кбіт/с) для кожного з напрямків відеоконференцзв’язку.

На центральній ЗС або в центральному вузлі інформатизації з розгалуженими комунікаціями підключення термінала BK Digital Signage може бути здійснене через магістральний мультиплексор, цифрову телефонну станцію або маршрутизатор.

Підключення терміналів BK Digital Signage до магістральних, міжрегіональних і регіональних каналів зв'язку через мультиплексори, цифрові АТМ або маршрутизатори забезпечує ряд переваг у порівнянні із прямим

КБПЗ-2025

підключенням до ЗС. По-перше, добре розвинені комунікаційні функції цих пристроїв дозволяють їх використовувати як вузли відеоконференцзв’язку (хоча з деякими обмеженнями). По-друге, завдяки широкому діапазону інтерфейсів, властивим цим пристроям, досягається можливість зв'язку терміналів ВК Digital Signage, що діють у різнорідних мережах (наприклад, супутникових і наземних, телефонних і мультимедійних і т.п.). При цьому забезпечується гнучкість структури системи відеоконференцзв’язку й подальше нарощування користувальницьких послуг, в основному програмними, а не апаратними засобами.

Крім модулів каналів зв'язку, сервер ВК Digital Signage містить модулі звукового процесора й відеопроцесора, синхронізатора зображення й звуку, транскодера H.261/H.263, а також набір інтерфейсних модулів, зв'язаних системною шиною. Керуючі модулі керують роботою всієї системи. На «транковом» рівні сервер ВК Digital Signage звичайно підтримує інтерфейси IDNX-PRI (Integrated Services Digital Network Primary Rate Interface) і T1 (E1) BBS (Robbed-Bit Signaling), на лінійному рівні – інтерфейс ISDN-BRI (Integrated Services Digital Network Basic Rate Interface), протокол цифрових з'єднань DCP (Digital Communications Protocol) і аналогові лінійні канали для з'єднання через модеми. Звичайно сервер ВК Digital Signage заснований на стандартах H.320 і T.120 MCE-T (Сектора Т Міжнародного союзу електрозв'язку) і має широкі можливості транскодування, що забезпечує сумісність його з найрізноманітнішими засобами відеоконференцій Digital Signage. В таблиці 3.1 для приклада наведені деякі характеристики одного із серверів ВК Digital Signage (MCU Lucent Technologies), що визначають його сумісність із іншим устаткуванням.

Сервер ВК Digital Signage підтримує відеостандарти, звукові стандарти й стандарти даних, перераховані в таблицях 3.2-3.3.

КБПЗ-2025

Термінали відеоконференцій Digital Signage

У системі відеоконференцій Digital Signage можуть бути використані моделі групових напольних, компактних і настільних відеотерміналів різних виробників. Властивості систем різних фірм у кожній із груп досить близькі.

Ці засоби дозволяють працювати по протоколах H.320 і H.323 і поряд із груповими й компактними терміналами ВК Digital Signage можуть бути включені в корпоративну систему ВК Digital Signage.

Таблиця 3.1 – Можливості транскодування

Категорія	Стандарти й параметри інтерфейсів
Транскодування сигналів зображення	G.711 і G.722
	G.711 і G.728
	G.711 і G.723 (через Gateway)
	Частота кадрів – від 7,5 до 30 кадр/с Розрішення – CIF/QCIF Стиск – H.261/H.263
Доступ до мережі	Цифрова швидкість – від 56 до 768 кбіт/с Режими – BONDing, багатошвидкісний, багатоканальний
Багатоточечні протоколи	H.320 і H.323
Конференція даних (T.120)	Допускається в комбінованих конференціях Змішання H.320/H.323 (через Gateway)

КБПЗ-2025

Комунікаційні властивості устаткування системи ВК Digital Signage

Основою технологічного устаткування відеоконференцзв'язку служать термінали ВК Digital Signage різних конфігурацій і сервер ВК Digital Signage із системою керування відеоконференціями. Сервер ВК Digital Signage є головним телекомунікаційним засобом багатоточечної системи ВК Digital Signage і має продуктивні цифрові інтерфейси для високошвидкісної передачі голосових повідомлень, зображень і даних. Він містить інтерфейси цифрових мереж

інтегрованих служб (ISDN): для базової цифрової швидкості (BRI) і первинної цифрової швидкості (PRI). Крім того, він підтримує протокол прямих цифрових з'єднань (DCP). При з'єднанні ISDN-PRI на рівні первинної групи групповторення E1 застосовується формат 30B + D, а при ISDN-BRI, – формат 2B + D, де B (64 кбіт/с) – потік для основної інформації служби, а D (16 кбіт/с) – потік керування й сигналізації для приєднаних каналів B. Протокол цифрових з'єднань DCP застосовується для зв'язку терміналів BK Digital Signage із системою керування за допомогою високошвидкісних і мультимедійних з'єднань (MML).

Таблиця 3.2 – Відеостандарти, підтримувані сервером BK Digital Signage

Відеостандарт MCE-T	Найменування, зміст
H.221	Стандарт структури кадрів для відеоконференцзв’язку
H.230	Стандарт кадрової синхронізації для відеоконференцзв’язку
H.231	Стандарт відеоконференцзв’язку, що визначає з'єднання між звуковізуальними терміналами
H.242	Стандарт, що визначає системи для подання з'єднань між звуковізуальними терміналами
H.243	Рекомендації ITU-T: процедура встановлення зв'язку між трьома або більше аудіовізуальними терміналами, що використовують канали зі швидкістю передачі цифрової інформації до 2 Мб/с.
H.261	Стандарт відеокодека для звуковізуальних служб зі швидкістю H.320
H.263	Кодування й декодування зображення для передачі з низькою швидкістю з поліпшеними характеристиками і якістю по каналах H.261

КБПЗ_2025

Таблиця 3.3 – Звукові стандарти, підтримувані сервером BK Digital Signage

Звуковий стандарт MCE-T	Параметри якості	Смуга, бітна швидкість
G.711	3,5 кГц	48/56/64 кбіт/с
G.722	7 кГц	48 кбіт/с на швидкості N*56 кбіт/с
		56 кбіт/с на швидкості N*64 кбіт/с
G.728	3.5 кГц	16 кбіт/с

Таблиця 3.4 – Стандарти даних, підтримувані сервером BK Digital Signage

Стандарти MCE-T	Найменування, зміст
T. 122	Багатоточечна служба зв'язку для звукографії й відеоконференцзв’язку
T.123	Стеки протоколів ISDNProfile-MLP для застосувань звукографії й відеоконференцзв’язку
T.124	Специфікація GCC (Genetic Conference Protocol)
T. 125	Протокол MC5
T.126	Протокол для нерухливих зображень і описів при багатоточечному зв'язку
T.127	Протокол перетворень багатоточечних бінарних файлів

КБПЗ_2025

Сервер BK Digital Signage може підключатися до кодеків H.320, H.CTX для передачі даних через інтерфейси EIA-530, RS449 або V.35 із застосуванням RS366 для сигналізації (дозвона). У випадку надання смуги на вимогу (BONDing) можна здійснювати проведення відеоконференцій Digital Signage без використання каналів ISDN-PRI і широкополосних каналів типу H0.

Функціональна схема сервера відеоконференцій Digital Signage зображена на рисунку 3.2.

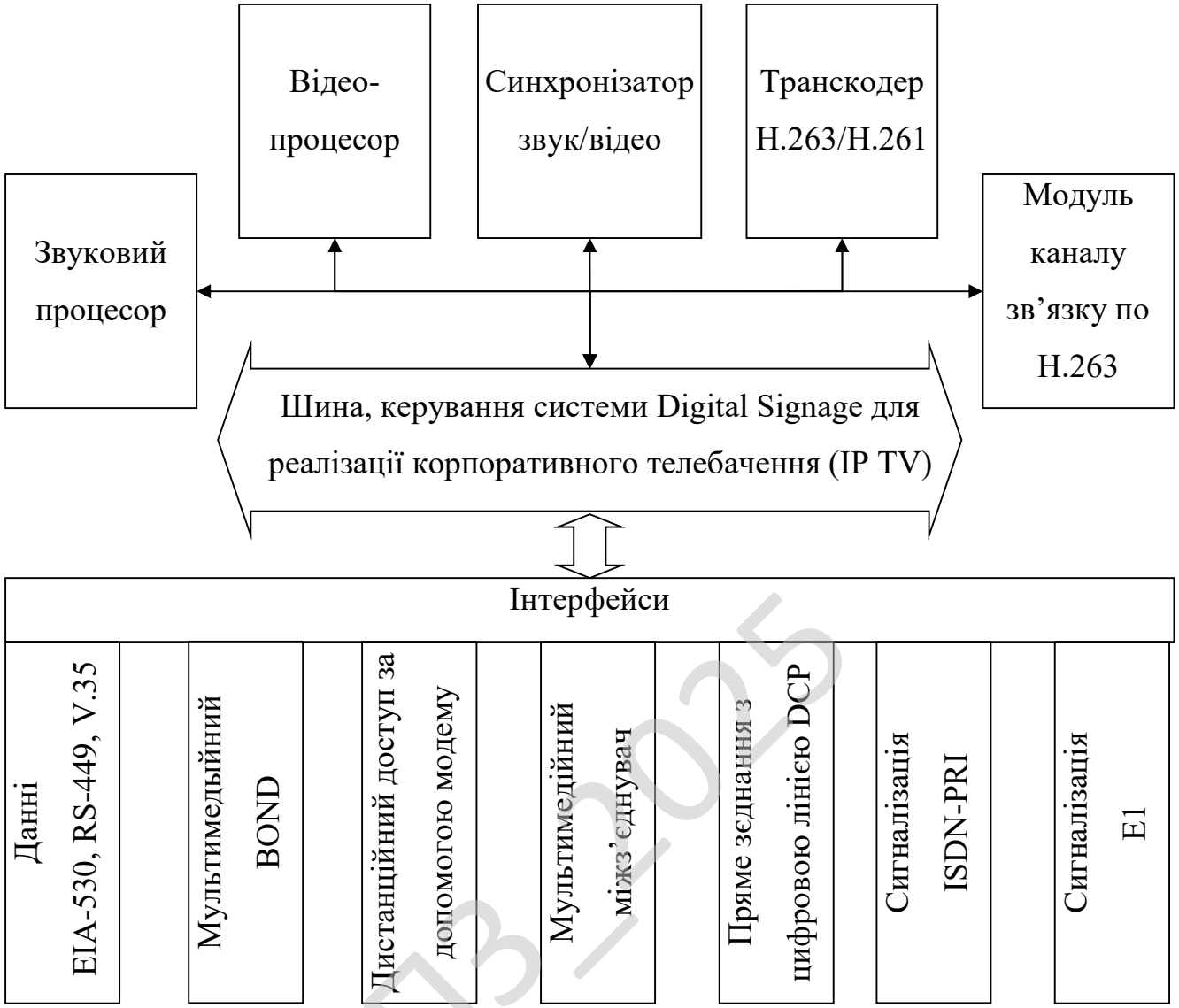


Рисунок 3.2 – Функціональна схема системи

КбПЗ_2025

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

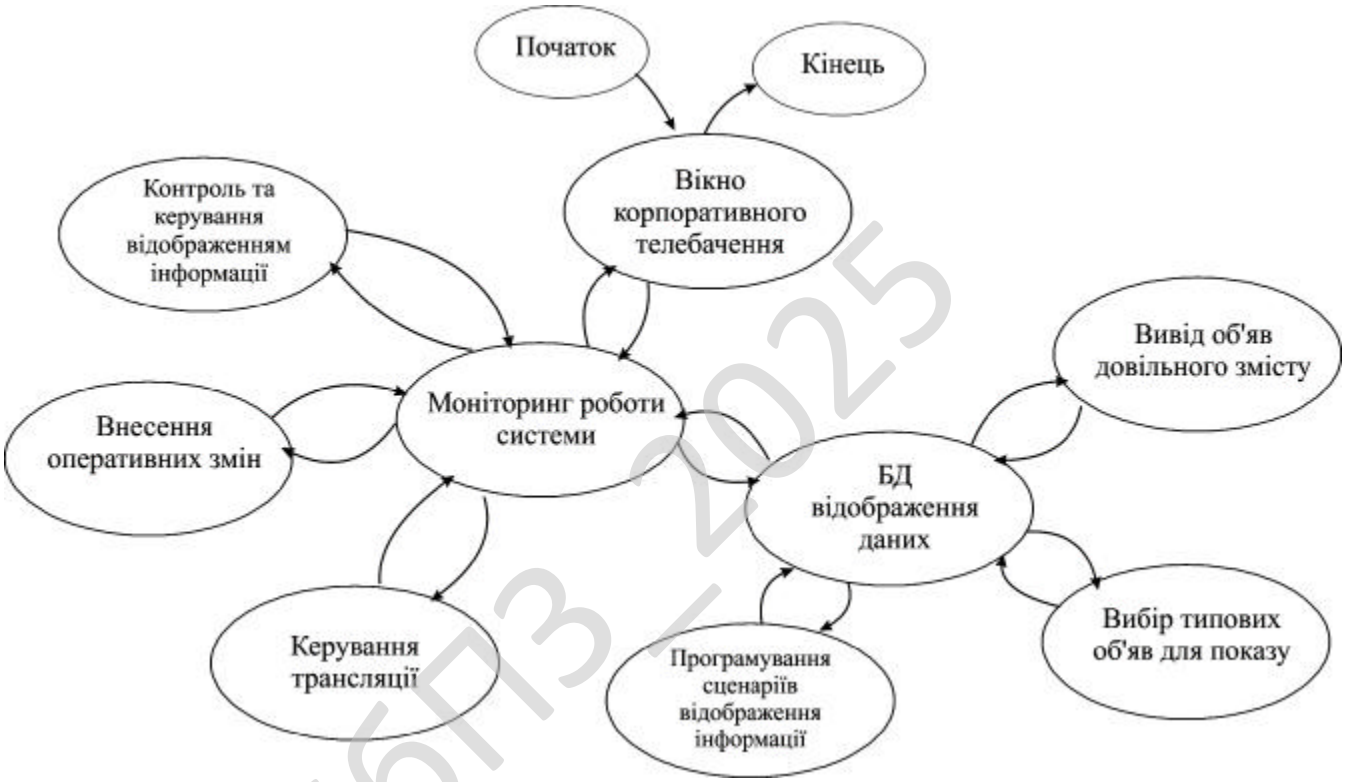


Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

КБПЗ – 2025

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

К6ПЗ_2025

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

КБПЗ-2025

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

КБПЗ_2025

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи Digital Signage для реалізації корпоративного телебачення (IP TV).

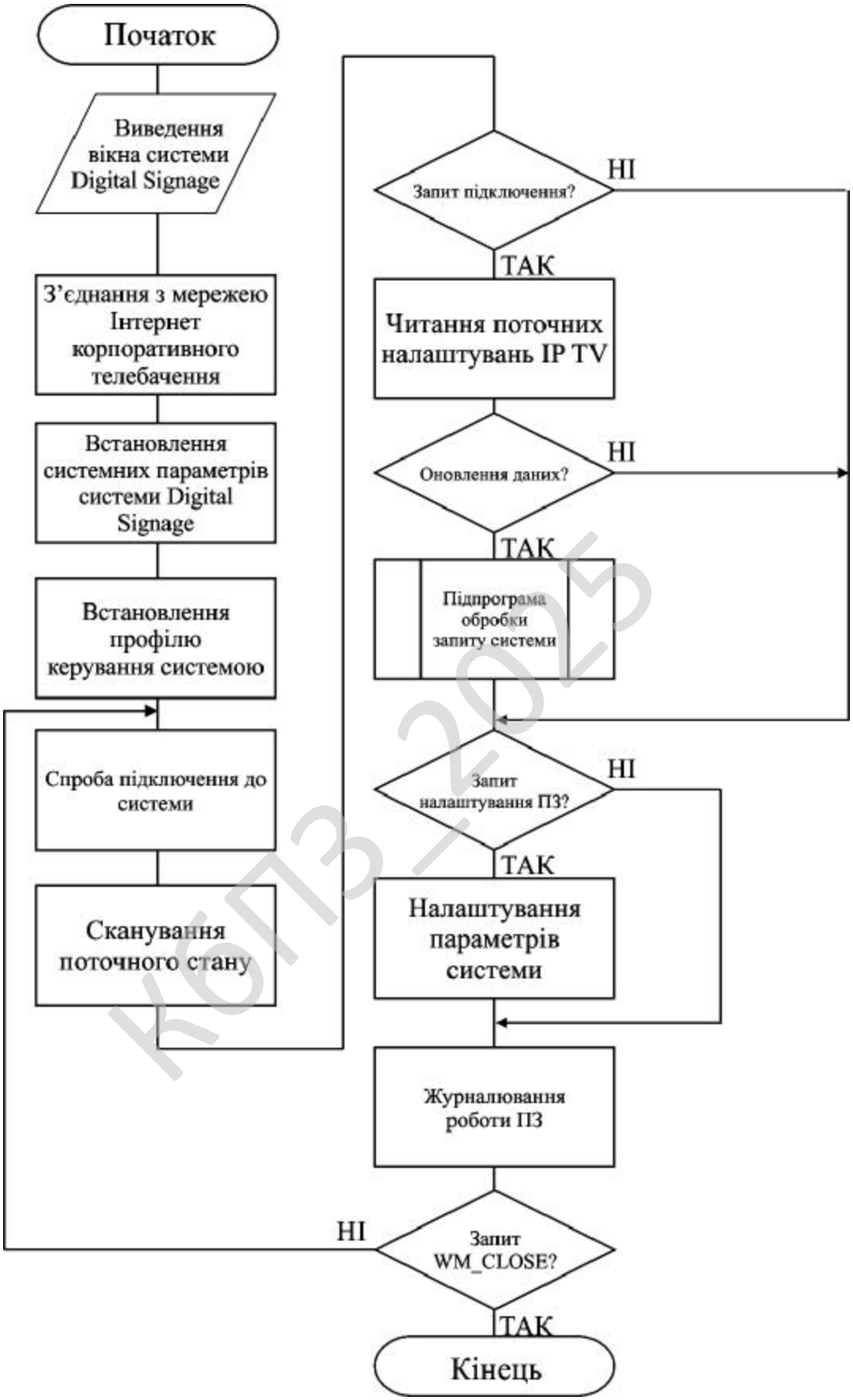
На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

КБПЗ-2025



КБПЗ – 2025

Рисунок 4.1 – Блок-схема основної програми

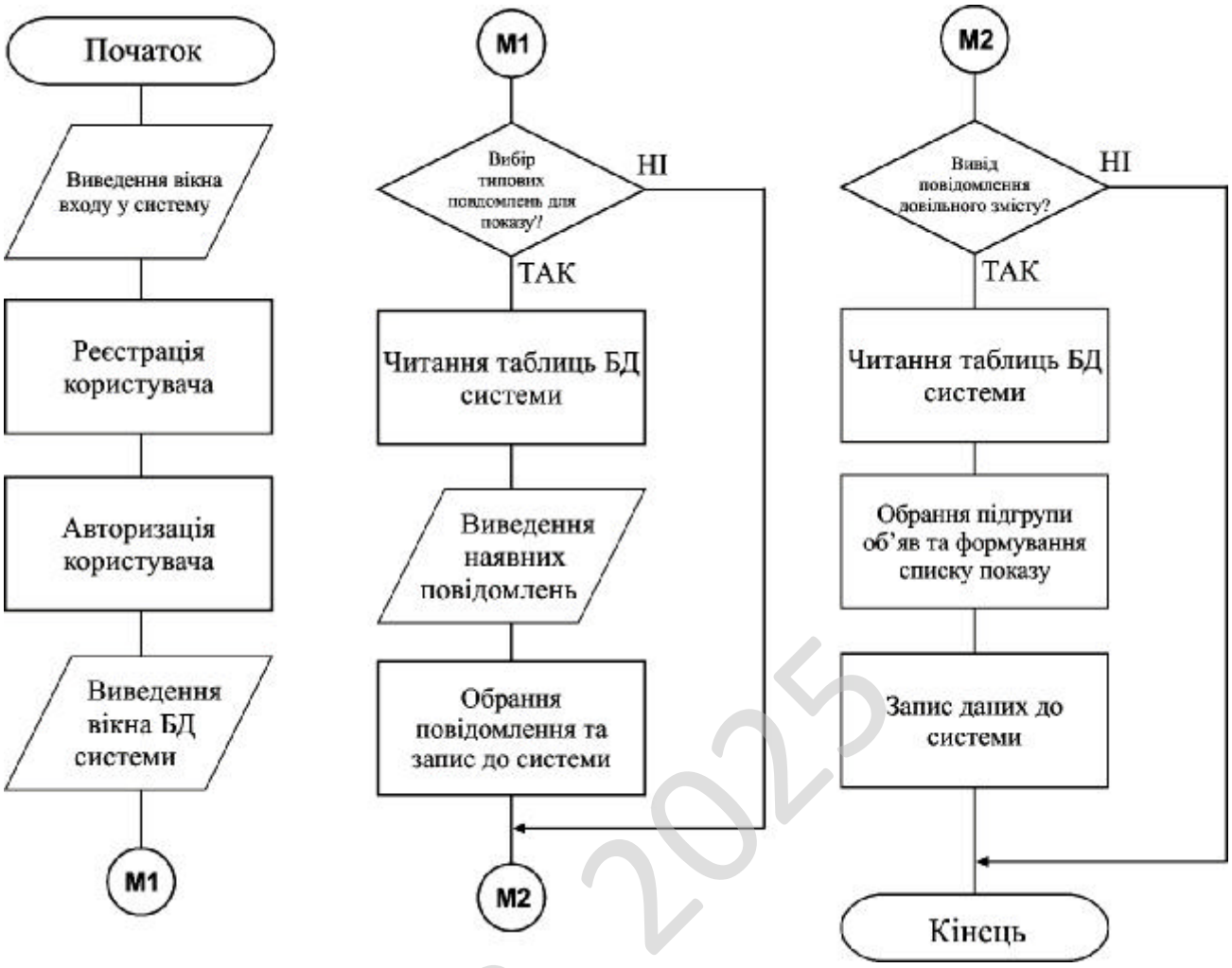


Рисунок 4.2 – Блок-схема роботи підпрограми

КБПЗ_2025

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об'єктно-орієнтованих мов програмування, що ще

більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

КБПЗ-2025

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню,

КБПЗ-2025

заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

КБПЗ_2025

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то);

– Дубль (повторює дефект, що вже знаходиться в роботі);

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо);

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

КБПЗ_2025

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

При розробці ПЗ було використано V-Model (або VEE модель) є моделлю розробки інформаційних систем (ІС), спрямованої на спрощення розуміння складнощів, пов'язаних з розробкою систем. Вона використовується для визначення єдиної процедури розробки програмного забезпечення, апаратного забезпечення та людино-машинного інтерфейсу.

Концепція V-подібної моделі була розроблена Німеччиною та США в кінці 1980-х років незалежно один від одного:

– Німецька V-модель була розроблена аерокосмічної компанією IABG в Оттобрунні поряд з Мюнхеном у сприянні з Федеральним департаментом з закупівлі озброєнь в Кобленці, для Міністерства оборони Німеччини. Модель була прийнята німецькою федеральною адміністрацією для цивільних потреб влітку 1992.

– Американська V-Model (VEE) була розроблена національною радою з системної інженерії (міжнародна – з 1995 року) для супутникових систем, включаючи обладнання, програмне забезпечення та взаємодію з користувачами.

КБПЗ-2025

Сучасною версією V-Model є V-Model XT, яка була затверджена в лютому 2005 року. V-модель використовується для управління процесом розробки програмного забезпечення для німецької федеральної адміністрації.

Зараз вона є стандартом для німецьких урядових і оборонних проектів, а також для виробників ПЗ в Німеччині. V-Model являє собою скоріше набір стандартів у галузі проектів, що стосуються розробки нових продуктів. Ця модель багато в чому схожа з Prince2 і описує методи як для проектного управління, так і для системного розвитку.

Основні принципи

Основний принцип V-подібної моделі полягає в тому, що деталізація проекту зростає при русі зліва направо, одночасно з плином часу, і ні те, ні інше не може повернути назад. Ітерації в проекті виробляються по горизонталі, між лівою і правою сторонами літери.

Стосовно до розробки інформаційних систем V-Model – варіація каскадної моделі, в якій завдання розробки йдуть зверху вниз по лівій стороні букви V, а завдання тестування – вгору по правій стороні букви V. Усередині V проводяться горизонтальні лінії, що показують, як результати кожної з фаз розробки впливають на розвиток системи тестування на кожній із фаз тестування.

Модель базується на тому, що приймально-здавальні випробування ґрунтуються, насамперед, на вимогах, системне тестування – на вимогах та архітектури, комплексне тестування – на вимогах, архітектурі та інтерфейсах, а компонентне тестування – на вимогах, архітектурі, інтерфейсах та алгоритмах

Цілі

V-модель забезпечує підтримку у плануванні та реалізації проекту. В ході проекту ставляться такі завдання:

1. Мінімізація ризиків: V-подібна модель робить проект більш прозорим і підвищує якість контролю проекту шляхом стандартизації проміжних цілей і опису відповідних їм результатів та відповідальних осіб. Це дозволяє виявляти

КБПЗ_2025

відхилення в проекті і ризики на ранніх стадіях і покращує якість управління проектом.

2. Підвищення та гарантії якості: V-Model – стандартизована модель розробки, що дозволяє домогтися від проекту результатів бажаної якості. Проміжні результати можуть бути перевірені на ранніх стадіях. Універсальне документування полегшує читаність, зрозумілість та контрольованість.

3. Зменшення загальної вартості проекту: Ресурси на розробку, виробництво, управління і підтримку можуть бути заздалегідь прораховані та проконтрольовані. Отримувані результати також універсальні і легко прогнозуються. Це зменшує витрати на подальші стадії та проекти.

4. Підвищення якості комунікації між учасниками проекту: Універсальний опис усіх елементів та умов полегшує взаєморозуміння всіх учасників проекту. Таким чином, зменшуються неточності у розумінні між користувачем, покупцем, постачальником і розробником.

Переваги:

– Користувачі V-Model беруть участь у розробці та підтримці V-моделі. Комітет з контролю за змінами підтримує проект і збирається раз на рік для обробки всіх отриманих запитів на внесення змін до V-Model.

– На старті будь-якого проекту V-подібна модель може бути адаптована під цей проект, так як ця модель не залежить від типів організацій та проектів.

– V-model дозволяє розбити діяльність на окремі кроки, кожен з яких буде включати в себе необхідні для нього дії, інструкції до них, рекомендації та докладне пояснення діяльності.

Обмеження. Наступні моменти не враховуються в V-моделі, але можуть бути розглянуті окремо, або можливо адаптувати модель під них:

– Не регулюється розміщення контрактів на обслуговування.

– Організація і виконання управління, обслуговування, ремонту та утилізації системи не враховуються в V-моделі. Однак, планування і підготовка до цих операцій моделлю розглядаються.

КБПЗ-2025

– V-подібна модель більше стосується розробки програмного забезпечення в проекті, ніж всієї організації процесу.

Переваги:

– У моделі особливе значення надається плануванню, спрямованому на верифікацію та атестацію розроблювального продукту на ранніх стадіях його розробки. Фаза модульного тестування підтверджує правильність деталізованого проектування. Фази інтеграції та тестування реалізують архітектурне проектування або проектування на вищому рівні. Фаза тестування системи підтверджує правильність виконання етапу вимог до продукту і його специфікації.

– У моделі передбачені атестація та верифікація всіх зовнішніх і внутрішніх отриманих даних, а не тільки самого програмного продукту.

– У V-подібної моделі визначення вимог виконується перед розробкою проекту системи, а проектування ПО – перед розробкою компонентів.

– Модель визначає продукти, які повинні бути отримані в результаті процесу розробки, причому кожні отримані дані повинні піддаватися тестуванню.

– Завдяки моделі менеджери проекту можуть відслідковувати хід процесу розробки, так як в даному випадку цілком можливо скористатися тимчасовою шкалою, а завершення кожної фази є контрольною точкою.

Недоліки:

– Модель не передбачає роботу з паралельними подіями.

– У моделі не передбачено внесення вимоги динамічних змін на різних етапах життєвого циклу.

– Тестування вимог в життєвому циклі відбувається занадто пізно, внаслідок чого неможливо внести змін, не вплинувши при цьому на графік виконання проекту.

– У модель не входять дії, спрямовані на аналіз ризиків.

– Деякий результат можна отримати тільки при досягненні низу букви V.

КБПЗ-2025

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних[2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього

КБПЗ_2025

переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;
– Лінійне перетворення π;
– Байтова перестановка τ;
– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

КБПЗ_2025

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ системи Digital Signage для реалізації корпоративного телебачення (IP TV)яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Файл; DS; Шаблони; Налаштування; Довідка.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок системи – основних функцій системи відповідно технічного завдання.

– Розділу обрання групи.

– Розділу виведення результату роботи системи.

КбПЗ_2025

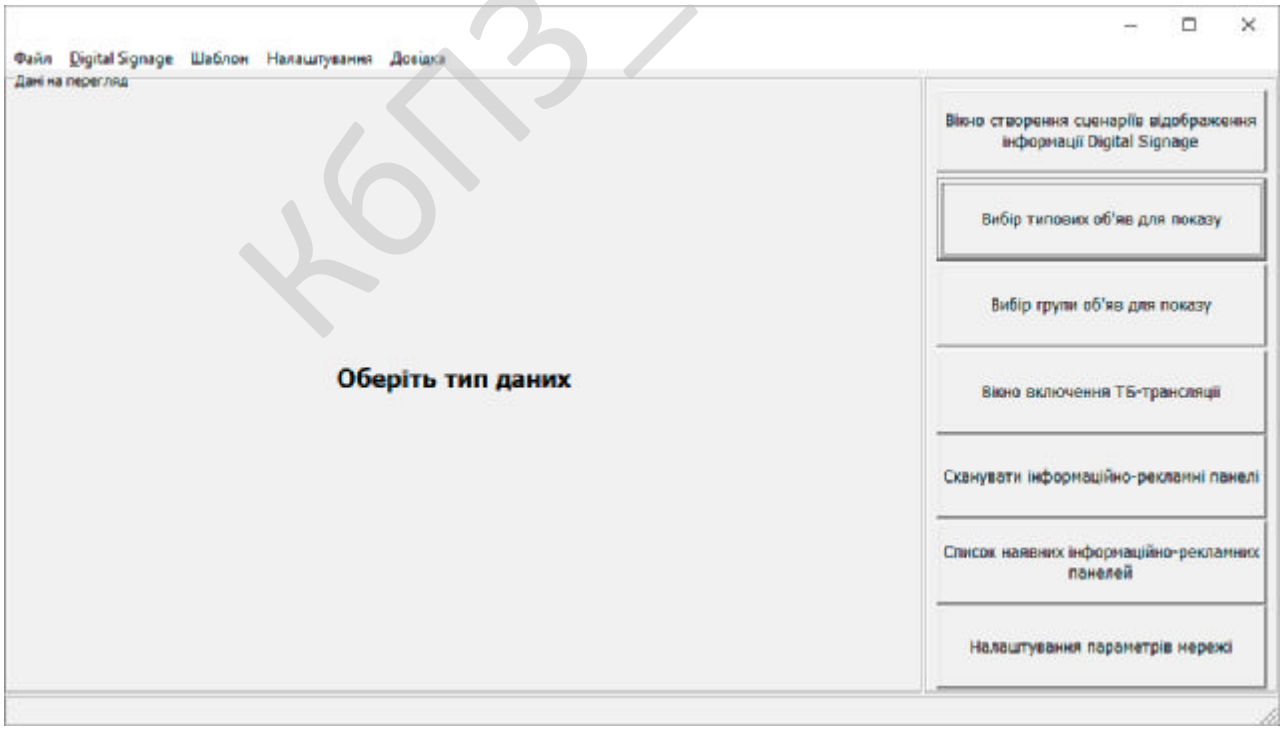


Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

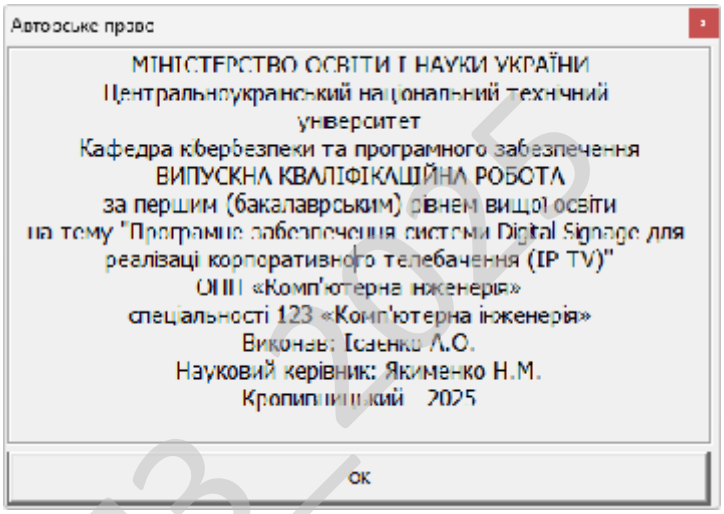


Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є

унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються

КБПЗ_2025

в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

КПЗ_2025

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Проводилось тестування чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

КБПЗ_2025

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

КБПЗ_2025

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

КБПЗ_2025

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи Digital Signage для реалізації корпоративного телебачення (IP TV).

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем Digital Signage для реалізації корпоративного телебачення (IP TV).

– Досліджена система Digital Signage для реалізації корпоративного телебачення (IP TV).

– На основі отриманих результатів досліджень створена програмна реалізація системи Digital Signage для реалізації корпоративного телебачення (IP TV).

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання Digital Signage для реалізації корпоративного телебачення (IP TV).

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи Digital Signage для реалізації корпоративного телебачення (IP TV). Це

КБПЗ_2025

дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CRYPTON.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

К6ПЗ_2025

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.

2. Peter Shirley, Steve Marschner. Fundamentals of Computer Graphics. 2009

3. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

4. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

5. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.

6. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. - Д.: НГУ, 2016. - 187 с.

7. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

8. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

9. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

10. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447

11. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in

КБПЗ-2025

software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

12. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

13. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

14. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

15. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

16. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w

17. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

18. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

КБПЗ-2025

19. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

20. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

21. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». Journal of theoretical and applied information technology Vol.98. No 21, 2020, P. 3334-3346.

22. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

23. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

24. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

25. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

КБПЗ_2025

26. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

27. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

28. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

29. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

30. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

31. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

32. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

33. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

КБПЗ – 2025

34. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

35. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

36. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

37. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

38. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

39. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

40. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

КБПЗ_2025

41. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

42. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

43. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у Кібербезпека та інформаційні технології: монографія. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

44. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

45. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

46. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

47. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

48. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan

КБПЗ-2025

D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

49. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

50. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

51. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

52. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

53. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп’ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

КбПЗ-2025

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування....2
2 Підстава для розробки....2
3 Мета та призначення розробки....2
4 Джерела розробки....2
5 Технічні вимоги....2
5.1 Вміст проекту....2
5.2 Показники призначення....3
5.3 Вимоги до функціональних характеристик....3
5.4 Вимоги до архітектури....3
5.5 Вимоги до надійності....3
5.6 Умови експлуатації....4
5.7 Вимоги до складу та параметрів технічних засобів....4
5.8 Вимоги до інформаційної і програмної сумісності....4
5.8.1 Обладнання....4
5.8.2 Мова програмування....4
5.8.3 Вхідні дані....5
5.8.4 Вихідні дані....5
6 Вимоги до програмної документації....5
7 Перелік документів, що розробляються....5
8 Етапи розробки....6
9 Порядок контролю та приймання....6

КБПЗ-2025

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи Digital Signage для реалізації корпоративного телебачення (IP TV).

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 48-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи Digital Signage для реалізації корпоративного телебачення (IP TV).

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

КБПЗ_2025

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи Digital Signage для реалізації корпоративного телебачення (IP TV);

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

КБПЗ_2025

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

КБПЗ_2025

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 93 аркуші.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

КБПЗ_2025

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 5.06.2025 р.

КбПЗ_2025

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Якименко Н.М.

Програмне забезпечення системи Digital Signage для реалізації корпоративного телебачення (IP TV)

КБПЗ_2025

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 21

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
#Імпорт стандартних бібліотек для роботи з системою
import sqlite3
#Імпорт бібліотеки для роботи з часом
import time
#Імпорт бібліотеки для роботи з потоками
import threading
#Імпорт бібліотеки для роботи з датою та часом
import datetime
#Імпорт бібліотеки для роботи з випадковими числами
import random
#Імпорт Flask для створення веб-сервера
from flask import Flask, request, jsonify, render_template_string

#Ініціалізація об'єкта Flask додатку
app = Flask(__name__)

#Створення класу DigitalSignageSystem для керування системою Digital Signage
class DigitalSignageSystem:
#Конструктор класу для ініціалізації змінних та підключення до бази даних
    def __init__(self, db_name="digital_signage.db"):
#Підключення до бази даних SQLite
        self.db_name = db_name
#Встановлення з'єднання з базою даних
        self.connection = sqlite3.connect(self.db_name, check_same_thread=False)
#Створення об'єкта курсора для виконання SQL запитів
        self.cursor = self.connection.cursor()
#Ініціалізація структури таблиць у базі даних
        self.setup_database()
#Створення списку потоків для роботи системи
        self.threads = []
#Ініціалізація змінних для зберігання статусу системи
        self.running = False

#Метод для налаштування бази даних та створення необхідних таблиць
    def setup_database(self):
#Створення таблиці для зберігання інформації про канали
        self.cursor.execute("CREATE TABLE IF NOT EXISTS channels (id INTEGER
PRIMARY KEY AUTOINCREMENT, name TEXT, url TEXT, description TEXT)")
#Виконання коміту для збереження змін у базі даних
        self.connection.commit()
#Створення таблиці для зберігання розкладу програм
        self.cursor.execute("CREATE TABLE IF NOT EXISTS schedule (id INTEGER
PRIMARY KEY AUTOINCREMENT, channel_id INTEGER, program_name TEXT, start_time
TEXT, end_time TEXT, FOREIGN KEY(channel_id) REFERENCES channels(id))")
#Виконання коміту для збереження змін у базі даних
        self.connection.commit()
#Створення таблиці для зберігання інформації про рекламні оголошення
        self.cursor.execute("CREATE TABLE IF NOT EXISTS ads (id INTEGER PRIMARY
KEY AUTOINCREMENT, ad_text TEXT, display_duration INTEGER)")
#Виконання коміту для збереження змін у базі даних
        self.connection.commit()
#Запис інформаційного повідомлення про успішне налаштування бази даних
        print("База даних налаштована та готова до використання")

#Метод для додавання нового каналу до системи
    def add_channel(self, name, url, description):
#SQL запит для додавання нового запису в таблицю channels
        self.cursor.execute("INSERT INTO channels (name, url, description)
VALUES (?, ?, ?)", (name, url, description))
#Виконання коміту для збереження нового каналу
        self.connection.commit()
```

КБПЗ – 2025

```
#Запис інформаційного повідомлення про додавання нового каналу
        print("Новий канал додано:", name)

#Метод для додавання нового рекламного оголошення
    def add_ad(self, ad_text, display_duration):
#SQL запит для додавання нового запису в таблицю ads
        self.cursor.execute("INSERT INTO ads (ad_text, display_duration) VALUES
(?, ?)", (ad_text, display_duration))
#Виконання коміту для збереження нового оголошення
        self.connection.commit()
#Запис інформаційного повідомлення про додавання рекламного оголошення
        print("Нове рекламне оголошення додано")

#Метод для планування програми для конкретного каналу
    def schedule_program(self, channel_id, program_name, start_time, end_time):
#SQL запит для додавання нового запису в таблицю schedule
        self.cursor.execute("INSERT INTO schedule (channel_id, program_name,
start_time, end_time) VALUES (?, ?, ?, ?)", (channel_id, program_name,
start_time, end_time))
#Виконання коміту для збереження розкладу
        self.connection.commit()
#Запис інформаційного повідомлення про успішне планування програми
        print("Програма", program_name, "запланована для каналу", channel_id)

#Метод для отримання списку всіх каналів з бази даних
    def get_channels(self):
#Виконання SQL запиту для отримання всіх каналів
        self.cursor.execute("SELECT * FROM channels")
#Отримання результатів запиту
        channels = self.cursor.fetchall()
#Повернення списку каналів
        return channels

#Метод для отримання розкладу програми для конкретного каналу
    def get_schedule_for_channel(self, channel_id):
#Виконання SQL запиту для отримання розкладу за каналом
        self.cursor.execute("SELECT * FROM schedule WHERE channel_id = ?",
(channel_id,))
#Отримання результатів запиту
        schedule = self.cursor.fetchall()
#Повернення розкладу програми
        return schedule

#Метод для отримання списку всіх рекламних оголошень
    def get_ads(self):
#Виконання SQL запиту для отримання всіх рекламних оголошень
        self.cursor.execute("SELECT * FROM ads")
#Отримання результатів запиту
        ads = self.cursor.fetchall()
#Повернення списку рекламних оголошень
        return ads

#Метод для запуску потоків для демонстрації потокового відтворення контенту
    def start_streaming(self):
#Встановлення статусу системи як активної
        self.running = True
#Створення нового потоку для відтворення каналів
        stream_thread = threading.Thread(target=self.stream_loop)
#Додавання потоку до списку активних потоків
        self.threads.append(stream_thread)
#Запуск потоку для відтворення каналів
        stream_thread.start()
#Запуск додаткового потоку для відтворення рекламних оголошень
```

КбПЗ_2025

```
        ads_thread = threading.Thread(target=self.ads_loop)
#Додавання потоку до списку активних потоків
        self.threads.append(ads_thread)
#Запуск потоку для відтворення рекламних оголошень
        ads_thread.start()
#Запис інформаційного повідомлення про старт системи потокового відтворення
        print("Система потокового відтворення запущена")

#Метод для зупинки потоків системи відтворення
    def stop_streaming(self):
#Встановлення статусу системи як неактивної
        self.running = False
#Очікування завершення всіх потоків
        for thread in self.threads:
            thread.join()
#Запис інформаційного повідомлення про зупинку системи потокового відтворення
        print("Система потокового відтворення зупинена")

#Основний цикл для відтворення контенту каналів
    def stream_loop(self):
#Поки система активна, продовжувати відтворення
        while self.running:
#Отримання списку каналів для відтворення
            channels = self.get_channels()
#Перевірка чи існують доступні канали
            if channels:
#Випадковий вибір каналу для відтворення
                channel = random.choice(channels)
#Отримання розкладу для вибраного каналу
                schedule = self.get_schedule_for_channel(channel[0])
#Запис інформаційного повідомлення про поточний канал та його розклад
                print("Відтворення каналу:", channel[1])
#Якщо для каналу існує розклад, вивести інформацію про наступну програму
                if schedule:
                    next_program = schedule[0]
                    print("Наступна програма:", next_program[2])
                else:
                    print("Для даного каналу немає запланованих програм")
#Симуляція відтворення шляхом очікування протягом кількох секунд
                time.sleep(5)
            else:
                print("Немає доступних каналів для відтворення")
                time.sleep(3)
#Запис інформаційного повідомлення про завершення циклу відтворення
        print("Вихід з циклу відтворення каналів")

#Основний цикл для відтворення рекламних оголошень
    def ads_loop(self):
#Поки система активна, продовжувати відтворення реклами
        while self.running:
#Отримання списку рекламних оголошень
            ads = self.get_ads()
#Перевірка чи існують доступні рекламні оголошення
            if ads:
#Випадковий вибір рекламного оголошення для відтворення
                ad = random.choice(ads)
#Запис інформаційного повідомлення про відтворення рекламного оголошення
                print("Відтворення реклами:", ad[1])
#Симуляція відтворення реклами протягом зазначеної тривалості
                time.sleep(ad[2])
            else:
                print("Немає рекламних оголошень для відтворення")
                time.sleep(3)
```

КОПЗ_2025

```
#Запис інформаційного повідомлення про завершення циклу відтворення реклами
        print("Вихід з циклу відтворення реклами")

#Метод для ініціалізації тестових даних у системі
    def initialize_test_data(self):
#Додавання тестового каналу 1
        self.add_channel("Корпоративний Новини", "http://example.com/stream1",
"Канал корпоративних новин")
#Додавання тестового каналу 2
        self.add_channel("Інформаційний Центр", "http://example.com/stream2",
"Канал інформаційного центру компанії")
#Додавання тестового рекламного оголошення 1
        self.add_ad("Реклама: Новий продукт від нашої компанії", 4)
#Додавання тестового рекламного оголошення 2
        self.add_ad("Реклама: Спеціальна пропозиція для клієнтів", 5)
#Отримання поточного часу
        current_time = datetime.datetime.now()
#Обчислення часу початку програми
        start_time = current_time.strftime("%Y-%m-%d %H:%M:%S")
#Обчислення часу завершення програми через 10 хвилин
        end_time = (current_time + datetime.timedelta(minutes=10)).strftime("%Y-
%m-%d %H:%M:%S")
#Додавання тестової програми для першого каналу
        self.schedule_program(1, "Ранкове Інформування", start_time, end_time)
#Запис інформаційного повідомлення про ініціалізацію тестових даних
        print("Тестові дані успішно ініціалізовані")

#Метод для оновлення розкладу програм (симуляція оновлення даних)
    def update_schedule(self):
#Бескінечний цикл для періодичного оновлення розкладу
        while self.running:
#Отримання списку каналів
            channels = self.get_channels()
#Перевірка чи існують канали для оновлення
            if channels:
#Ітерація по кожному каналу
                for channel in channels:
#Отримання поточного часу
                    current_time = datetime.datetime.now()
#Форматування часу для запису в базу даних
                    start_time = current_time.strftime("%Y-%m-%d %H:%M:%S")
#Обчислення часу завершення програми через 15 хвилин
                    end_time = (current_time +
datetime.timedelta(minutes=15)).strftime("%Y-%m-%d %H:%M:%S")
#Додавання нового запису в розклад для каналу
                    self.schedule_program(channel[0], "Автоматична Програма",
start_time, end_time)
#Запис інформаційного повідомлення про оновлення розкладу для даного каналу
                    print("Оновлено розклад для каналу", channel[1])
#Очікування перед наступною ітерацією оновлення розкладу
            time.sleep(60)
#Запис інформаційного повідомлення про завершення циклу оновлення розкладу
        print("Вихід з циклу оновлення розкладу")

#Метод для запуску додаткового потоку оновлення розкладу
    def start_schedule_update(self):
#Створення нового потоку для оновлення розкладу
        schedule_thread = threading.Thread(target=self.update_schedule)
#Додавання потоку до списку активних потоків
        self.threads.append(schedule_thread)
#Запуск потоку оновлення розкладу
        schedule_thread.start()
#Запис інформаційного повідомлення про старт оновлення розкладу
```

КОПЗ - 2025

```
        print("Старт оновлення розкладу програм")

#Функція для симуляції потокового відтворення контенту через HTTP запити
@app.route("/stream/<int:channel_id>")
def stream(channel_id):
#Отримання параметрів запиту від користувача
    mode = request.args.get("mode", "default")
#Симуляція обробки запиту для відтворення контенту
    content = f"Симульоване відео від каналу {channel_id} у режимі {mode}"
#Запис інформаційного повідомлення про обробку запиту
    print("Обробка запиту на відтворення каналу", channel_id)
#Повернення симульованого відеоконтенту у вигляді тексту
    return content

#Функція для отримання списку всіх каналів через HTTP запит
@app.route("/channels")
def list_channels():
#Створення екземпляру системи Digital Signage
    dss = DigitalSignageSystem()
#Отримання списку каналів з бази даних
    channels = dss.get_channels()
#Формування списку каналів у форматі словника для повернення у форматі JSON
    channels_list = [{"id": channel[0], "name": channel[1], "url": channel[2],
"description": channel[3]} for channel in channels]
#Запис інформаційного повідомлення про повернення списку каналів
    print("Повернення списку всіх каналів")
#Повернення списку каналів у форматі JSON
    return jsonify(channels_list)

#Функція для додавання нового каналу через HTTP POST запит
@app.route("/admin/add_channel", methods=["POST"])
def admin_add_channel():
#Отримання даних з тіла запиту
    data = request.get_json()
#Отримання імені каналу з даних запиту
    name = data.get("name", "")
#Отримання URL каналу з даних запиту
    url = data.get("url", "")
#Отримання опису каналу з даних запиту
    description = data.get("description", "")
#Перевірка наявності обов'язкових даних
    if not name or not url:
#Запис інформаційного повідомлення про помилку при додаванні каналу
        print("Помилка: відсутні необхідні дані для додавання каналу")
#Повернення повідомлення про помилку
        return jsonify({"error": "Необхідні дані відсутні"}), 400
#Створення екземпляру системи Digital Signage
    dss = DigitalSignageSystem()
#Додавання каналу до системи
    dss.add_channel(name, url, description)
#Запис інформаційного повідомлення про успішне додавання каналу
    print("Канал успішно додано через адміністративний інтерфейс")
#Повернення повідомлення про успішну операцію
    return jsonify({"message": "Канал додано успішно"}), 200

#Функція для додавання нового розкладу програми через HTTP POST запит
@app.route("/admin/schedule_program", methods=["POST"])
def admin_schedule_program():
#Отримання даних з тіла запиту
    data = request.get_json()
#Отримання ID каналу з даних запиту
    channel_id = data.get("channel_id", 0)
#Отримання імені програми з даних запиту
```

КБТЗ_2025

```
    program_name = data.get("program_name", "")
#Отримання часу початку програми з даних запиту
    start_time = data.get("start_time", "")
#Отримання часу завершення програми з даних запиту
    end_time = data.get("end_time", "")
#Перевірка наявності обов'язкових даних
    if not channel_id or not program_name or not start_time or not end_time:
#Запис інформаційного повідомлення про помилку при плануванні програми
        print("Помилка: відсутні необхідні дані для планування програми")
#Повернення повідомлення про помилку
        return jsonify({"error": "Необхідні дані відсутні"}), 400
#Створення екземпляру системи Digital Signage
    dss = DigitalSignageSystem()
#Додавання розкладу програми для каналу
    dss.schedule_program(channel_id, program_name, start_time, end_time)
#Запис інформаційного повідомлення про успішне планування програми
    print("Програма успішно запланована через адміністративний інтерфейс")
#Повернення повідомлення про успішну операцію
    return jsonify({"message": "Програма запланована успішно"}), 200

#Функція для перегляду розкладу програми для конкретного каналу через HTTP GET
запит
@app.route("/admin/get_schedule/<int:channel_id>")
def admin_get_schedule(channel_id):
#Створення екземпляру системи Digital Signage
    dss = DigitalSignageSystem()
#Отримання розкладу програми для заданого каналу
    schedule = dss.get_schedule_for_channel(channel_id)
#Формування списку розкладу у форматі словника для повернення у форматі JSON
    schedule_list = [{"id": item[0], "channel_id": item[1], "program_name":
item[2], "start_time": item[3], "end_time": item[4]} for item in schedule]
#Запис інформаційного повідомлення про повернення розкладу програми
    print("Повернення розкладу для каналу", channel_id)
#Повернення розкладу програми у форматі JSON
    return jsonify(schedule_list)

#Основна функція для запуску системи Digital Signage
def main():
#Створення екземпляру системи Digital Signage
    dss = DigitalSignageSystem()
#Ініціалізація тестових даних для демонстрації роботи системи
    dss.initialize_test_data()
#Запуск потоку для відтворення контенту каналів
    dss.start_streaming()
#Запуск потоку для оновлення розкладу програм
    dss.start_schedule_update()
#Запис інформаційного повідомлення про запуск основної функції
    print("Основна функція запущена. Сервер готовий до роботи.")
#Запуск Flask веб-сервера для обробки HTTP запитів
    app.run(host="0.0.0.0", port=8080)

#Перевірка, чи виконується даний модуль як основний
if __name__ == "__main__":
#Виклик основної функції для запуску системи
    main()

#Додатковий код для підтримки розширення функціональності системи
def auxiliary_function():
#Функція для демонстрації додаткових можливостей системи
    print("Виконується допоміжна функція")
#Виконання випадкової операції для демонстрації обробки даних
    result = sum([random.randint(1, 10) for _ in range(100)])
#Запис інформаційного повідомлення про результат допоміжної операції
```

КБПЗ 2025

```
    print("Результат допоміжної операції:", result)
#Повернення отриманого результату
    return result

#Виклик допоміжної функції для демонстрації роботи додаткового коду
auxiliary_function()

#Додатковий код для симуляції логування подій системи
def log_event(event_message):
#Отримання поточного часу для запису події
    current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
#Формування рядка логування з інформацією про час та повідомлення
    log_entry = f"{current_time} - {event_message}"
#Запис інформаційного повідомлення про логування події в консоль
    print("Лог події:", log_entry)
#Відкриття файлу для збереження логів
    with open("system.log", "a") as log_file:
#Запис логування події у файл
        log_file.write(log_entry + "\n")
#Повернення записаного логування
    return log_entry

#Виклик функції логування для демонстрації запису подій
log_event("Система Digital Signage успішно запущена")

#Додатковий код для перевірки цілісності бази даних
def check_database_integrity():
#Встановлення з'єднання з базою даних для перевірки
    connection = sqlite3.connect("digital_signage.db")
#Створення об'єкта курсора для виконання SQL запитів
    cursor = connection.cursor()
#Виконання SQL запиту для отримання інформації про таблиці
    cursor.execute("SELECT name FROM sqlite_master WHERE type='table'")
#Отримання списку таблиць у базі даних
    tables = cursor.fetchall()
#Запис інформаційного повідомлення про перевірку бази даних
    print("Перевірка таблиць у базі даних:", tables)
#Закриття з'єднання з базою даних
    connection.close()
#Повернення списку таблиць для подальшого аналізу
    return tables

#Виклик функції перевірки цілісності бази даних
check_database_integrity()

#Додатковий код для обробки користувацького вводу через командний рядок
def command_line_interface():
#Виведення привітального повідомлення для користувача
    print("Вітаємо у системі управління Digital Signage")
#Безкінечний цикл для обробки вводу користувача
    while True:
#Виведення інструкцій для користувача
        print("Введіть 'exit' для виходу або 'log' для перегляду останніх
подій")
#Отримання вводу користувача з командного рядка
        user_input = input("Введіть команду: ")
#Перевірка команди на вихід із системи
        if user_input.lower() == "exit":
#Запис інформаційного повідомлення про вихід з системи
            print("Вихід з командного інтерфейсу")
#Припинення циклу обробки вводу
            break
#Перевірка команди для перегляду логів
```

КОПЗ_2025

```
        elif user_input.lower() == "log":
#Відкриття файлу з логами для зчитування даних
            with open("system.log", "r") as log_file:
#Читання всіх рядків з файлу логів
                logs = log_file.read()
#Виведення вмісту логів у консоль
            print("Останні події системи:\n", logs)
#Обробка невідомих команд користувача
        else:
#Запис інформаційного повідомлення про невідому команду
            print("Невідома команда. Будь ласка, спробуйте ще раз.")
#Виклик функції логування для запису дії користувача
        log_event(f"Командний рядок отримав команду: {user_input}")

#Запуск командного інтерфейсу у окремому потоці
cli_thread = threading.Thread(target=command_line_interface)
cli_thread.start()
```

КБПЗ_2025

Файл ServerMonitor.py

```
import sqlite3
import time
import threading
import datetime
import random
from flask import Flask, request, jsonify
app = Flask(__name__)
class DigitalSignageSystem:
    def __init__(self, db_name="digital_signage.db"):
        self.db_name = db_name
        self.connection = sqlite3.connect(self.db_name, check_same_thread=False)
        self.cursor = self.connection.cursor()
        self.setup_database()
        self.running = False
        self.threads = []
    def setup_database(self):
        self.cursor.execute("CREATE TABLE IF NOT EXISTS channels (id INTEGER
PRIMARY KEY AUTOINCREMENT, name TEXT, url TEXT, description TEXT)")
        self.connection.commit()
        self.cursor.execute("CREATE TABLE IF NOT EXISTS schedule (id INTEGER
PRIMARY KEY AUTOINCREMENT, channel_id INTEGER, program_name TEXT, start_time
TEXT, end_time TEXT, FOREIGN KEY(channel_id) REFERENCES channels(id))")
        self.connection.commit()
        self.cursor.execute("CREATE TABLE IF NOT EXISTS ads (id INTEGER PRIMARY
KEY AUTOINCREMENT, ad_text TEXT, display_duration INTEGER)")
        self.connection.commit()
    def add_channel(self, name, url, description):
        self.cursor.execute("INSERT INTO channels (name, url, description)
VALUES (?, ?, ?)", (name, url, description))
        self.connection.commit()
    def add_ad(self, ad_text, display_duration):
        self.cursor.execute("INSERT INTO ads (ad_text, display_duration) VALUES
(?, ?)", (ad_text, display_duration))
        self.connection.commit()
    def schedule_program(self, channel_id, program_name, start_time, end_time):
        self.cursor.execute("INSERT INTO schedule (channel_id, program_name,
start_time, end_time) VALUES (?, ?, ?, ?)", (channel_id, program_name,
start_time, end_time))
        self.connection.commit()
    def get_channels(self):
        self.cursor.execute("SELECT * FROM channels")
        channels = self.cursor.fetchall()
        return channels
    def get_schedule_for_channel(self, channel_id):
        self.cursor.execute("SELECT * FROM schedule WHERE channel_id = ?",
(channel_id,))
        schedule = self.cursor.fetchall()
        return schedule
    def get_ads(self):
        self.cursor.execute("SELECT * FROM ads")
        ads = self.cursor.fetchall()
        return ads
    def stream_loop(self):
        while self.running:
            channels = self.get_channels()
            if channels:
                channel = random.choice(channels)
                schedule = self.get_schedule_for_channel(channel[0])
                if schedule:
                    next_program = schedule[0]
                time.sleep(5)
```

КОТЗ-2025

```
            else:
                time.sleep(3)
    def ads_loop(self):
        while self.running:
            ads = self.get_ads()
            if ads:
                ad = random.choice(ads)
                time.sleep(ad[2])
            else:
                time.sleep(3)
    def start_streaming(self):
        self.running = True
        stream_thread = threading.Thread(target=self.stream_loop)
        self.threads.append(stream_thread)
        stream_thread.start()
        ads_thread = threading.Thread(target=self.ads_loop)
        self.threads.append(ads_thread)
        ads_thread.start()
    def stop_streaming(self):
        self.running = False
        for thread in self.threads:
            thread.join()
    def initialize_test_data(self):
        self.add_channel("Корпоративний Новини", "http://example.com/stream1",
"Канал корпоративних новин")
        self.add_channel("Інформаційний Центр", "http://example.com/stream2",
"Канал інформаційного центру компанії")
        self.add_ad("Реклама: Новий продукт від нашої компанії", 4)
        self.add_ad("Реклама: Спеціальна пропозиція для клієнтів", 5)
        current_time = datetime.datetime.now()
        start_time = current_time.strftime("%Y-%m-%d %H:%M:%S")
        end_time = (current_time + datetime.timedelta(minutes=10)).strftime("%Y-
%m-%d %H:%M:%S")
        self.schedule_program(1, "Ранкове Інформування", start_time, end_time)
class AdminPanel:
    def __init__(self, dss):
        self.dss = dss
    def get_dashboard_data(self):
        channels = self.dss.get_channels()
        schedule_all = []
        for channel in channels:
            sch = self.dss.get_schedule_for_channel(channel[0])
            schedule_all.append({"channel": channel, "schedule": sch})
        ads = self.dss.get_ads()
        data = {"channels": channels, "schedule": schedule_all, "ads": ads}
        return data
    def update_channel_info(self, channel_id, name, url, description):
        self.dss.cursor.execute("UPDATE channels SET name = ?, url = ?,
description = ? WHERE id = ?", (name, url, description, channel_id))
        self.dss.connection.commit()
    def remove_channel(self, channel_id):
        self.dss.cursor.execute("DELETE FROM channels WHERE id = ?",
(channel_id,))
        self.dss.connection.commit()
    def add_new_channel(self, name, url, description):
        self.dss.add_channel(name, url, description)
    def get_all_users(self):
        return []
    def add_user(self, username, password):
        return True
    def update_user(self, user_id, username, password):
        return True
    def delete_user(self, user_id):
```

КоПЗ_2025

```
        return True
class ServerMonitor:
    def __init__(self):
        self.server_status = {}
        self.running = False
        self.thread = None
    def collect_metrics(self):
        cpu_usage = random.uniform(0, 100)
        memory_usage = random.uniform(0, 100)
        disk_usage = random.uniform(0, 100)
        return {"cpu": cpu_usage, "memory": memory_usage, "disk": disk_usage,
"timestamp": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")}
    def monitor_loop(self):
        while self.running:
            metrics = self.collect_metrics()
            self.server_status = metrics
            time.sleep(5)
    def start_monitoring(self):
        self.running = True
        self.thread = threading.Thread(target=self.monitor_loop)
        self.thread.start()
    def stop_monitoring(self):
        self.running = False
        if self.thread:
            self.thread.join()
    def get_status(self):
        return self.server_status
class SocialMediaIntegration:
    def __init__(self):
        self.feeds = []
    def fetch_facebook_feed(self):
        feed = [{"id": i, "content": "Facebook пост " + str(i), "timestamp":
datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")} for i in range(3)]
        return feed
    def fetch_twitter_feed(self):
        feed = [{"id": i, "content": "Twitter твітування " + str(i),
"timestamp": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")} for i in
range(3,6)]
        return feed
    def fetch_instagram_feed(self):
        feed = [{"id": i, "content": "Instagram фото " + str(i), "timestamp":
datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")} for i in range(6,9)]
        return feed
    def aggregate_feeds(self):
        fb = self.fetch_facebook_feed()
        tw = self.fetch_twitter_feed()
        ig = self.fetch_instagram_feed()
        self.feeds = fb + tw + ig
        self.feeds.sort(key=lambda x: x["timestamp"], reverse=True)
        return self.feeds
class NotificationSystem:
    def __init__(self):
        self.notifications = []
        self.lock = threading.Lock()
    def add_notification(self, message):
        notif = {"id": len(self.notifications) + 1, "message": message,
"timestamp": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")}
        with self.lock:
            self.notifications.append(notif)
        return notif
    def get_notifications(self):
        with self.lock:
            return list(self.notifications)
```

НБПЗ_2025

```
    def clear_notifications(self):
        with self.lock:
            self.notifications.clear()
class CRMIntegration:
    def __init__(self):
        self.crm_data = {}
    def fetch_crm_data(self):
        data = {"customers": []}
        for i in range(5):
            customer = {"id": i, "name": "Customer " + str(i), "email":
"customer" + str(i) + "@example.com", "status": "active" if i % 2 == 0 else
"inactive"}
            data["customers"].append(customer)
        self.crm_data = data
        return data
    def update_crm_customer(self, customer_id, name, email, status):
        for cust in self.crm_data.get("customers", []):
            if cust["id"] == customer_id:
                cust["name"] = name
                cust["email"] = email
                cust["status"] = status
                break
        return True
dss_instance = DigitalSignageSystem()
admin_panel = AdminPanel(dss_instance)
server_monitor = ServerMonitor()
social_media = SocialMediaIntegration()
notification_system = NotificationSystem()
crm_integration = CRMIntegration()
server_monitor.start_monitoring()
@app.route("/admin/dashboard", methods=["GET"])
def admin_dashboard():
    data = admin_panel.get_dashboard_data()
    return jsonify(data)
@app.route("/server/status", methods=["GET"])
def server_status():
    status = server_monitor.get_status()
    return jsonify(status)
@app.route("/social/feeds", methods=["GET"])
def social_feeds():
    feeds = social_media.aggregate_feeds()
    return jsonify(feeds)
@app.route("/notifications", methods=["GET", "POST", "DELETE"])
def notifications():
    if request.method == "GET":
        notifs = notification_system.get_notifications()
        return jsonify(notifs)
    elif request.method == "POST":
        data = request.get_json()
        message = data.get("message", "")
        notif = notification_system.add_notification(message)
        return jsonify(notif)
    elif request.method == "DELETE":
        notification_system.clear_notifications()
        return jsonify({"message": "Notifications cleared"})
    else:
        return jsonify({"error": "Method not allowed"}), 405
@app.route("/crm", methods=["GET", "PUT"])
def crm():
    if request.method == "GET":
        data = crm_integration.fetch_crm_data()
        return jsonify(data)
    elif request.method == "PUT":
```

КБПЗ_2025

```
        data = request.get_json()
        customer_id = data.get("id")
        name = data.get("name", "")
        email = data.get("email", "")
        status = data.get("status", "")
        crm_integration.update_crm_customer(customer_id, name, email, status)
        return jsonify({"message": "Customer updated"})
    else:
        return jsonify({"error": "Method not allowed"}), 405
def main():
    dss_instance.initialize_test_data()
    dss_instance.start_streaming()
    app.run(host="0.0.0.0", port=8080)
if __name__ == "__main__":
    main()
```

К6ПЗ_2025

Файл ViewAnalytics.py

```
import sqlite3
import time
import threading
import datetime
import random
from flask import Flask, request, jsonify
app = Flask(__name__)
class ViewAnalytics:
    def __init__(self, db_name="analytics.db"):
        self.db_name = db_name
        self.connection = sqlite3.connect(self.db_name, check_same_thread=False)
        self.cursor = self.connection.cursor()
        self.setup_database()
        self.lock = threading.Lock()
    def setup_database(self):
        self.cursor.execute("CREATE TABLE IF NOT EXISTS views (id INTEGER
PRIMARY KEY AUTOINCREMENT, channel_id INTEGER, view_time TEXT, user_ip TEXT)")
        self.connection.commit()
    def log_view(self, channel_id, user_ip):
        view_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        with self.lock:
            self.cursor.execute("INSERT INTO views (channel_id, view_time,
user_ip) VALUES (?, ?, ?)", (channel_id, view_time, user_ip))
            self.connection.commit()
    def get_view_count(self, channel_id):
        self.cursor.execute("SELECT COUNT(*) FROM views WHERE channel_id = ?",
(channel_id,))
        count = self.cursor.fetchone()[0]
        return count
    def get_views_by_date(self, date_str):
        self.cursor.execute("SELECT * FROM views WHERE DATE(view_time)=?",
(date_str,))
        return self.cursor.fetchall()
class MultimediaLibrary:
    def __init__(self, db_name="media_library.db"):
        self.db_name = db_name
        self.connection = sqlite3.connect(self.db_name, check_same_thread=False)
        self.cursor = self.connection.cursor()
        self.setup_database()
    def setup_database(self):
        self.cursor.execute("CREATE TABLE IF NOT EXISTS media (id INTEGER
PRIMARY KEY AUTOINCREMENT, title TEXT, file_path TEXT, media_type TEXT, duration
INTEGER)")
        self.connection.commit()
    def add_media(self, title, file_path, media_type, duration):
        self.cursor.execute("INSERT INTO media (title, file_path, media_type,
duration) VALUES (?, ?, ?, ?)", (title, file_path, media_type, duration))
        self.connection.commit()
    def get_media_list(self):
        self.cursor.execute("SELECT * FROM media")
        return self.cursor.fetchall()
    def search_media(self, keyword):
        self.cursor.execute("SELECT * FROM media WHERE title LIKE ?", ('%' +
keyword + '%',))
        return self.cursor.fetchall()
class GeoTargetedAds:
    def __init__(self, db_name="geo_ads.db"):
        self.db_name = db_name
        self.connection = sqlite3.connect(self.db_name, check_same_thread=False)
        self.cursor = self.connection.cursor()
```

КбПЗ-2025

```
        self.setup_database()
    def setup_database(self):
        self.cursor.execute("CREATE TABLE IF NOT EXISTS geo_ads (id INTEGER
PRIMARY KEY AUTOINCREMENT, ad_text TEXT, region TEXT, display_duration
INTEGER)")
        self.connection.commit()
    def add_geo_ad(self, ad_text, region, display_duration):
        self.cursor.execute("INSERT INTO geo_ads (ad_text, region,
display_duration) VALUES (?, ?, ?)", (ad_text, region, display_duration))
        self.connection.commit()
    def get_ads_for_region(self, region):
        self.cursor.execute("SELECT * FROM geo_ads WHERE region = ?", (region,))
        return self.cursor.fetchall()
    def get_all_ads(self):
        self.cursor.execute("SELECT * FROM geo_ads")
        return self.cursor.fetchall()
class InteractivePolls:
    def __init__(self, db_name="polls.db"):
        self.db_name = db_name
        self.connection = sqlite3.connect(self.db_name, check_same_thread=False)
        self.cursor = self.connection.cursor()
        self.setup_database()
        self.lock = threading.Lock()
    def setup_database(self):
        self.cursor.execute("CREATE TABLE IF NOT EXISTS polls (id INTEGER
PRIMARY KEY AUTOINCREMENT, question TEXT)")
        self.cursor.execute("CREATE TABLE IF NOT EXISTS poll_options (id INTEGER
PRIMARY KEY AUTOINCREMENT, poll_id INTEGER, option_text TEXT, votes INTEGER)")
        self.connection.commit()
    def create_poll(self, question, options):
        with self.lock:
            self.cursor.execute("INSERT INTO polls (question) VALUES (?)",
(question,))
            poll_id = self.cursor.lastrowid
            for option in options:
                self.cursor.execute("INSERT INTO poll_options (poll_id,
option_text, votes) VALUES (?, ?, ?)", (poll_id, option, 0))
            self.connection.commit()
            return poll_id
    def vote_poll(self, poll_id, option_id):
        with self.lock:
            self.cursor.execute("UPDATE poll_options SET votes = votes + 1 WHERE
id = ? AND poll_id = ?", (option_id, poll_id))
            self.connection.commit()
    def get_poll_results(self, poll_id):
        self.cursor.execute("SELECT option_text, votes FROM poll_options WHERE
poll_id = ?", (poll_id,))
        return self.cursor.fetchall()
    def get_all_polls(self):
        self.cursor.execute("SELECT * FROM polls")
        polls = self.cursor.fetchall()
        result = []
        for poll in polls:
            self.cursor.execute("SELECT * FROM poll_options WHERE poll_id = ?",
(poll[0],))
            options = self.cursor.fetchall()
            result.append({"poll": poll, "options": options})
        return result
class BackupSystem:
    def __init__(self, source_db, backup_db):
        self.source_db = source_db
        self.backup_db = backup_db
    def perform_backup(self):
```

КОД3_2025

```
        source_conn = sqlite3.connect(self.source_db)
        backup_conn = sqlite3.connect(self.backup_db)
        source_conn.backup(backup_conn)
        backup_conn.close()
        source_conn.close()
    def schedule_backup(self, interval_seconds):
        def backup_loop():
            while True:
                self.perform_backup()
                time.sleep(interval_seconds)
        backup_thread = threading.Thread(target=backup_loop)
        backup_thread.daemon = True
        backup_thread.start()
view_analytics = ViewAnalytics()
multimedia_library = MultimediaLibrary()
geo_ads = GeoTargetedAds()
interactive_polls = InteractivePolls()
backup_system = BackupSystem("digital_signage.db", "digital_signage_backup.db")
@app.route("/analytics/log_view/<int:channel_id>", methods=["POST"])
def log_view(channel_id):
    user_ip = request.remote_addr
    view_analytics.log_view(channel_id, user_ip)
    return jsonify({"message": "View logged"})
@app.route("/analytics/view_count/<int:channel_id>", methods=["GET"])
def view_count(channel_id):
    count = view_analytics.get_view_count(channel_id)
    return jsonify({"channel_id": channel_id, "view_count": count})
@app.route("/media/add", methods=["POST"])
def add_media():
    data = request.get_json()
    title = data.get("title", "")
    file_path = data.get("file_path", "")
    media_type = data.get("media_type", "")
    duration = data.get("duration", 0)
    multimedia_library.add_media(title, file_path, media_type, duration)
    return jsonify({"message": "Media added"})
@app.route("/media/list", methods=["GET"])
def list_media():
    media = multimedia_library.get_media_list()
    return jsonify(media)
@app.route("/media/search", methods=["GET"])
def search_media():
    keyword = request.args.get("keyword", "")
    results = multimedia_library.search_media(keyword)
    return jsonify(results)
@app.route("/geo_ads/add", methods=["POST"])
def add_geo_ad():
    data = request.get_json()
    ad_text = data.get("ad_text", "")
    region = data.get("region", "")
    display_duration = data.get("display_duration", 0)
    geo_ads.add_geo_ad(ad_text, region, display_duration)
    return jsonify({"message": "Geo ad added"})
@app.route("/geo_ads/region/<region>", methods=["GET"])
def ads_for_region(region):
    ads = geo_ads.get_ads_for_region(region)
    return jsonify(ads)
@app.route("/polls/create", methods=["POST"])
def create_poll():
    data = request.get_json()
    question = data.get("question", "")
    options = data.get("options", [])
    poll_id = interactive_polls.create_poll(question, options)
```

КБТЗ_2025

```
    return jsonify({"poll_id": poll_id})
@app.route("/polls/vote", methods=["POST"])
def vote_poll():
    data = request.get_json()
    poll_id = data.get("poll_id", 0)
    option_id = data.get("option_id", 0)
    interactive_polls.vote_poll(poll_id, option_id)
    return jsonify({"message": "Vote recorded"})
@app.route("/polls/results/<int:poll_id>", methods=["GET"])
def poll_results(poll_id):
    results = interactive_polls.get_poll_results(poll_id)
    return jsonify(results)
@app.route("/polls/all", methods=["GET"])
def all_polls():
    polls = interactive_polls.get_all_polls()
    return jsonify(polls)
@app.route("/backup/perform", methods=["POST"])
def perform_backup():
    backup_system.perform_backup()
    return jsonify({"message": "Backup performed"})
@app.route("/backup/schedule", methods=["POST"])
def schedule_backup():
    data = request.get_json()
    interval = data.get("interval", 3600)
    backup_system.schedule_backup(interval)
    return jsonify({"message": "Backup scheduled"})
def main():
    backup_system.schedule_backup(3600)
    app.run(host="0.0.0.0", port=8081)
if __name__ == "__main__":
    main()
```

К6ПЗ_2025

Файл FailureNotification.py

```
import sqlite3
import time
import threading
import datetime
import random
from flask import Flask, request, jsonify
app = Flask(__name__)
class FailureNotification:
    def __init__(self, db_name="failures.db"):
        self.db_name = db_name
        self.conn = sqlite3.connect(self.db_name, check_same_thread=False)
        self.cursor = self.conn.cursor()
        self.setup_db()
    def setup_db(self):
        self.cursor.execute("CREATE TABLE IF NOT EXISTS failures (id INTEGER
PRIMARY KEY AUTOINCREMENT, error_message TEXT, timestamp TEXT)")
        self.conn.commit()
    def log_failure(self, error_message):
        timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        self.cursor.execute("INSERT INTO failures (error_message, timestamp)
VALUES (?, ?)", (error_message, timestamp))
        self.conn.commit()
    def get_failures(self):
        self.cursor.execute("SELECT * FROM failures")
        return self.cursor.fetchall()
class MultilingualSupport:
    def __init__(self):
        self.languages = {"en":{}, "uk":{}, "hi":{}}
    def add_translation(self, lang, key, translation):
        if lang in self.languages:
            self.languages[lang][key] = translation
    def translate(self, lang, key):
        return self.languages.get(lang, {}).get(key, key)
    def get_all_translations(self, lang):
        return self.languages.get(lang, {})
class AntiFraudSystem:
    def __init__(self):
        self.suspicious_events = []
        self.lock = threading.Lock()
    def detect_anomaly(self, data):
        if random.random() < 0.1:
            event = {"event": "anomaly_detected", "data": data, "timestamp":
datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")}
            with self.lock:
                self.suspicious_events.append(event)
            return event
        return None
    def get_suspicious_events(self):
        with self.lock:
            return list(self.suspicious_events)
class CustomizationModule:
    def __init__(self, db_name="customization.db"):
        self.db_name = db_name
        self.conn = sqlite3.connect(self.db_name, check_same_thread=False)
        self.cursor = self.conn.cursor()
        self.setup_db()
    def setup_db(self):
        self.cursor.execute("CREATE TABLE IF NOT EXISTS themes (id INTEGER
PRIMARY KEY AUTOINCREMENT, user_id INTEGER, theme_name TEXT, settings TEXT)")
        self.conn.commit()
    def save_theme(self, user_id, theme_name, settings):
```

КБПЗ_2025

```
        self.cursor.execute("INSERT INTO themes (user_id, theme_name, settings)
VALUES (?, ?, ?)", (user_id, theme_name, settings))
        self.conn.commit()
    def get_user_themes(self, user_id):
        self.cursor.execute("SELECT * FROM themes WHERE user_id = ?",
(user_id,))
        return self.cursor.fetchall()
class UserAuthentication:
    def __init__(self, db_name="users.db"):
        self.db_name = db_name
        self.conn = sqlite3.connect(self.db_name, check_same_thread=False)
        self.cursor = self.conn.cursor()
        self.setup_db()
    def setup_db(self):
        self.cursor.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER
PRIMARY KEY AUTOINCREMENT, username TEXT UNIQUE, password TEXT)")
        self.conn.commit()
    def register_user(self, username, password):
        try:
            self.cursor.execute("INSERT INTO users (username, password) VALUES
(?, ?)", (username, password))
            self.conn.commit()
            return True
        except sqlite3.IntegrityError:
            return False
    def authenticate_user(self, username, password):
        self.cursor.execute("SELECT * FROM users WHERE username = ? AND password
= ?", (username, password))
        user = self.cursor.fetchone()
        return user is not None
    def get_all_users(self):
        self.cursor.execute("SELECT id, username FROM users")
        return self.cursor.fetchall()
failure_notification = FailureNotification()
multilingual_support = MultilingualSupport()
anti_fraud_system = AntiFraudSystem()
customization_module = CustomizationModule()
user_auth = UserAuthentication()
@app.route("/failures/log", methods=["POST"])
def log_failure():
    data = request.get_json()
    error_message = data.get("error_message", "")
    failure_notification.log_failure(error_message)
    return jsonify({"message": "Failure logged"})
@app.route("/failures", methods=["GET"])
def get_failures():
    failures = failure_notification.get_failures()
    return jsonify(failures)
@app.route("/translate", methods=["GET"])
def translate_text():
    lang = request.args.get("lang", "en")
    key = request.args.get("key", "")
    translation = multilingual_support.translate(lang, key)
    return jsonify({"translation": translation})
@app.route("/translate/add", methods=["POST"])
def add_translation():
    data = request.get_json()
    lang = data.get("lang", "en")
    key = data.get("key", "")
    translation = data.get("translation", "")
    multilingual_support.add_translation(lang, key, translation)
    return jsonify({"message": "Translation added"})
@app.route("/fraud/detect", methods=["POST"])
```

КБТВ_2025

```
def detect_fraud():
    data = request.get_json()
    event = anti_fraud_system.detect_anomaly(data)
    if event:
        return jsonify({"anomaly": event})
    return jsonify({"anomaly": "none"})
@app.route("/fraud/events", methods=["GET"])
def get_fraud_events():
    events = anti_fraud_system.get_suspicious_events()
    return jsonify(events)
@app.route("/customization/save", methods=["POST"])
def save_theme():
    data = request.get_json()
    user_id = data.get("user_id", 0)
    theme_name = data.get("theme_name", "")
    settings = data.get("settings", "")
    customization_module.save_theme(user_id, theme_name, settings)
    return jsonify({"message": "Theme saved"})
@app.route("/customization/themes/<int:user_id>", methods=["GET"])
def get_themes(user_id):
    themes = customization_module.get_user_themes(user_id)
    return jsonify(themes)
@app.route("/auth/register", methods=["POST"])
def register_user():
    data = request.get_json()
    username = data.get("username", "")
    password = data.get("password", "")
    success = user_auth.register_user(username, password)
    if success:
        return jsonify({"message": "User registered"})
    return jsonify({"message": "Registration failed"}), 400
@app.route("/auth/login", methods=["POST"])
def login_user():
    data = request.get_json()
    username = data.get("username", "")
    password = data.get("password", "")
    authenticated = user_auth.authenticate_user(username, password)
    if authenticated:
        return jsonify({"message": "Authenticated"})
    return jsonify({"message": "Authentication failed"}), 401
@app.route("/auth/users", methods=["GET"])
def list_users():
    users = user_auth.get_all_users()
    return jsonify(users)
def main():
    app.run(host="0.0.0.0", port=8082)
if __name__ == "__main__":
    main()
```

КБП3_2025