Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Воронюк Дмитро Сергійович

студент групи АС-151

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

CRM СИСТЕМА З ВИКОРИСТАННЯМ GOOGLE SHEETS

Спеціальність:

121 – Інженерія програмного забезпечення

Спеціалізація:

Інженерія програмного забезпечення

Керівник:

Комлева Наталія Олегівна,

кандидат технічних наук, доцент

Одеса – 2020

ЗМІСТ

[ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ 4](#_Toc58700579)

[ЗАВДАННЯ НА РОЗРОБКУ РОЗДІЛУ «ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» 6](#_Toc58700580)

[АНОТАЦІЯ 7](#_Toc58700581)

[ВСТУП 8](#_Toc58700582)

[1КРИТИЧНИЙ РОЗБІР ІСНУЮЧИХ РІШЕНЬ 10](#_Toc58700583)

[1.1 Аналіз проблем користувачів 10](#_Toc58700584)

[1.2 Аналіз аналогів 11](#_Toc58700585)

[2 ТЕОРЕТИЧНА ЧАСТИНА CRM СИСТЕМИ 14](#_Toc58700586)

[2.1 Аналіз даних 14](#_Toc58700587)

[2.1.1 Загальні відомості про структурукомпанії 14](#_Toc58700588)

[2.1.2 Детальний опис характеристиккомунальних послуг 15](#_Toc58700589)

[2.2 Аналіз функціоналу 20](#_Toc58700590)

[3 ФОРМУВАННЯ ВИМОГ ДО CRM СИСТЕМИ 22](#_Toc58700591)

[3.1 Формування функціональних вимог 22](#_Toc58700592)

[3.1.1 Функціональні вимогинастільного застосування 22](#_Toc58700593)

[3.1.2 Варіанти використання вебзастосування системи 26](#_Toc58700594)

[3.2 Формування нефункціональних вимог 29](#_Toc58700595)

[4 ПРОЕКТУВАННЯ CRM СИСТЕМИ 31](#_Toc58700596)

[4.1 Проектування архітектури системи 31](#_Toc58700597)

[4.1.1 Високорівнева архітектура системи 31](#_Toc58700598)

[4.1.2 Архітектура настільного застосування 32](#_Toc58700599)

[4.1.3 Архітектура вебзастосування 34](#_Toc58700600)

[4.2 Проектування структури і організації класів 34](#_Toc58700601)

[4.2.1 Організація класів настільного застосування 34](#_Toc58700602)

[4.2.2 Організація класів бібліотеки класів Models 37](#_Toc58700603)

[4.2.3 Організація класів бібліотеки класів GoogleLib 39](#_Toc58700604)

[4.2.4 Організація класів бібліотеки класів SheetsEF 40](#_Toc58700605)

[4.2.5 Організація класів вебзастосування 41](#_Toc58700606)

[4.3 Проектування інтерфейсу користувача CRM системи 44](#_Toc58700607)

[4.3.1 Проектування інтерфейсу настільного застосування 45](#_Toc58700608)

[4.3.2 Проектування інтерфейсу вебзастосування 46](#_Toc58700609)

[5 ПРОГРАМНА РЕАЛІЗАЦІЯ CRM СИСТЕМИ 49](#_Toc58700610)

[5.1 Створення проекту для сервісів Google 49](#_Toc58700611)

[5.2 Набір інструментальних засобів розробки 50](#_Toc58700612)

[5.3 Набір використаних бібліотек 52](#_Toc58700613)

[6 ТЕСТУВАННЯ ТА РОЗГОРТАННЯ CRM СИСТЕМИ 53](#_Toc58700614)

[6.1 Тестування настільного застосування 53](#_Toc58700615)

[6.2 Тестування вебзастосування 67](#_Toc58700616)

[6.3 Розгортання системи 76](#_Toc58700617)

[7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ 77](#_Toc58700618)

[ВИСНОВКИ 79](#_Toc58700619)

[СПИСОК ВИКОРИСТАННОЇ ЛІТЕРАТУРИ 81](#_Toc58700620)

[ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ 5](#_Toc58702558)

[ЗАВДАННЯ НА РОЗРОБКУ РОЗДІЛУ «ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» 7](#_Toc58702559)

[АНОТАЦІЯ 8](#_Toc58702560)

[ВСТУП 9](#_Toc58702561)

[1 КРИТИЧНИЙ РОЗБІР ІСНУЮЧИХ РІШЕНЬ 11](#_Toc58702562)

[1.1 Аналіз проблем користувачів 11](#_Toc58702563)

[1.2 Аналіз аналогів 12](#_Toc58702564)

[2 ТЕОРЕТИЧНА ЧАСТИНА CRM СИСТЕМИ 15](#_Toc58702565)

[2.1 Аналіз даних 15](#_Toc58702566)

[2.1.1 Загальні відомості про структурукомпанії 15](#_Toc58702567)

[2.1.2 Детальний опис характеристиккомунальних послуг 16](#_Toc58702568)

[2.2 Аналіз функціоналу 21](#_Toc58702569)

[3 ФОРМУВАННЯ ВИМОГ ДО CRM СИСТЕМИ 23](#_Toc58702570)

[3.1 Формування функціональних вимог 23](#_Toc58702571)

[3.1.1 Функціональні вимогинастільного застосування 23](#_Toc58702572)

[3.1.2 Варіанти використання вебзастосування системи 27](#_Toc58702573)

[3.2 Формування нефункціональних вимог 30](#_Toc58702574)

[4 ПРОЕКТУВАННЯ CRM СИСТЕМИ 32](#_Toc58702575)

[4.1 Проектування архітектури системи 32](#_Toc58702576)

[4.1.1 Високорівнева архітектура системи 32](#_Toc58702577)

[4.1.2 Архітектура настільного застосування 33](#_Toc58702578)

[4.1.3 Архітектура вебзастосування 35](#_Toc58702579)

[4.2 Проектування структури і організації класів 35](#_Toc58702580)

[4.2.1 Організація класів настільного застосування 35](#_Toc58702581)

[4.2.2 Організація класів бібліотеки класів Models 38](#_Toc58702582)

[4.2.3 Організація класів бібліотеки класів GoogleLib 40](#_Toc58702583)

[4.2.4 Організація класів бібліотеки класів SheetsEF 41](#_Toc58702584)

[4.2.5 Організація класів вебзастосування 42](#_Toc58702585)

[4.3 Проектування інтерфейсу користувача CRM системи 45](#_Toc58702586)

[4.3.1 Проектування інтерфейсу настільного застосування 46](#_Toc58702587)

[4.3.2 Проектування інтерфейсу вебзастосування 47](#_Toc58702588)

[5 ПРОГРАМНА РЕАЛІЗАЦІЯ CRM СИСТЕМИ 50](#_Toc58702589)

[5.1 Створення проекту для сервісів Google 50](#_Toc58702590)

[5.2 Набір інструментальних засобів розробки 51](#_Toc58702591)

[5.3 Набір використаних бібліотек 53](#_Toc58702592)

[6 ТЕСТУВАННЯ ТА РОЗГОРТАННЯ CRM СИСТЕМИ 54](#_Toc58702593)

[6.1 Тестування настільного застосування 54](#_Toc58702594)

[6.2 Тестування вебзастосування 68](#_Toc58702595)

[6.3 Розгортання системи 77](#_Toc58702596)

[7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ 78](#_Toc58702597)

[ВИСНОВКИ 80](#_Toc58702598)

[СПИСОК ВИКОРИСТАННОЇ ЛІТЕРАТУРИ 82](#_Toc58702599)

[ДОДАТОК А. ПУБЛІКАЦІЯ 84](#_Toc58702600)

[ДОДАТОК Б. ЛИСТІНГ ПРОГРАМНОЇ СИСТЕМИ 86](#_Toc58702601)

[ДОДАТОК В. ПИТАННЯ З ОХОРОНИ ПРАЦІ ТА БЕЗПЕКИ У НАДЗВИЧАЙНИХ СИТУАЦІЯХ 129](#_Toc58702602)

[В.1 Охорона праці. Організація та управління 129](#_Toc58702603)

[В.2 Охорона праці. Визначення основних параметрів умов праці згідно індивідуального завдання 131](#_Toc58702604)

[В.3 Охорона праці. Індивідуальний розрахунок 132](#_Toc58702605)

[В.4 Безпека у надзвичайних ситуаціях техногенного характеру. Індивідуальний розрахунок 134](#_Toc58702606)

[В.5 Заходи з охорони праці та безпеки у надзвичайних ситуаціях 137](#_Toc58702607)

[В.6 Безпека у суспільстві в умовах загрози COVID-19 138](#_Toc58702608)

Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_ Крісілов В. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2020 р.

# ЗАВДАННЯНА КВАЛІФІКАЦІЙНУ РОБОТУ

Воронюка Дмитра Сергійовича, група АС-151

1.Тема роботи: CRMсистема за використанням GoogleSheets

Керівник роботи: Комлева Наталія Олегівна,кандидат технічних наук, доцент

затверджені наказом ректора від 29 жовтня 2020 р. № 412-в.

2.Зміст роботи:

1. Критичний розбір існуючих рішень;
2. Теоретична частина CRM системи;
3. Формування вимог до CRMсистеми;
4. Проектування CRM системи;
5. Програмна реалізація CRM системи;
6. Тестування та розгортання CRM системи;
7. Охорона праці та безпека у надзвичайних ситуаціях.

3. Перелік ілюстративного матеріалу:

Згідно зі слайдами презентації.

4. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
| 7 | доц. Москалюк А. Ю. | 13.11 | 20.11 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

5. Дата видачі завдання: 2 вересня 2020 р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів кваліфікаційної роботи | Строк  виконання | Примітка |
| 1 | Критичний розбір існуючих рішень | 02.09 - 05.09 | Виконано |
| 2 | Теоретична частина CRM системи | 06.09 - 11.09 | Виконано |
| 3 | Формування вимог до CRM системи | 12.09 -18.09 | Виконано |
| 4 | Проектування CRM системи | 19.09-30.09 | Виконано |
| 5 | Програмна реалізація CRM системи | 01.10 - 03.11 | Виконано |
| 6 | Тестування та розгортання CRM системи | 04.11 - 12.11 | Виконано |
| 7 | Охорона праці та безпека у надзвичайних ситуаціях | 13.11 -20.11 | Виконано |
| 8 | Оформлення пояснювальної записки і графічного матеріалу | 21.11 - 10.12 | Виконано |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Здобувач вищої освіти** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. С. Воронюк

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. О. Комлева

# ЗАВДАННЯНА РОЗРОБКУ РОЗДІЛУ «ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ»

Воронюка Дмитра Сергійовича, група АС-151

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Тема роботи: CRM система з використаннямGoogleSheets.

Зміст розділу:

1. Організація та управління охороною праці на робочому місті програміста-розробника;
2. Визначення основних параметрів умов праці робочого місця;
3. Розрахунок заземлення;
4. Безпека під час надзвичайних ситуації техногенного характеру;
5. Заходи з охорони праці та безпеки у надзвичайних ситуаціях;
6. Безпека у суспільстві в умовах загрози COVID-19.

Керівник роботи Консультант з охорони праці та БНС

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. О. Комлева \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Ю. Москалюк

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 р. «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 р.

# АНОТАЦІЯ

Метою роботи є зменшення витрат на зберігання даних та скорочення часу на виконання бізнес процесів шляхом розробки CRM системи з безкоштовним варіантом зберігання даних за електронних таблиць компанії Google. Як результат виконана програмна реалізація CRMсистеми для сервісної компанії у сфері житлово-комунальних послуг (ЖКГ) зі зберіганням інформації за допомогою GoogleSheets.

Методи розробки базуються на технологіях .NETCore 3 (WPF, ASP.NETCore), GoogleAPI, Angular 10.

Ключові слова: CRMсистеми, житлово-комунальні господарство (ЖКГ), комунальні послуги, .NETCore, WPF, GoogleSheets, настільне застосування, вебзастосування.

**ABSTRACT**

The goal is to reduce storage costs and reduce business processes time by developing a CRM system with a free version of Google's spreadsheet storage. As a result, the software implementation of the CRM system for the service company in the field of housing and communal services with storage of information using Google Sheets.

Development methods are based on .NET Core 3 technologies (WPF, ASP.NET Core), Google API, Angular 10.

Keywords:CRM systems, housing and communal services, utilities, .NET Core, WPF, Google Sheets, desktop application, web application.

# ВСТУП

Ще 25-30 років тому компанії не здогадувалися, що в них може бути персоналізована система для виконання специфічних потреб компанії. Таке могли дозволити лише гіганти індустрії, але ситуація суттєво змінилася. Зараз навіть невеликі компанії можуть мати програмне забезпечення для автоматизації процесів та управління відносинами з клієнтами.

Управління відносинами з клієнтами (англ.Customerrelationshipmanagement або CRM) – поняття, що охоплює концепції, які використовуються компаніями для управління взаємовідносинами зі споживачами, включаючи збір, зберігання й аналіз інформації про споживачів, постачальників, партнерів та інформації про взаємовідносини з ними [1].

Управління відносинами з клієнтами потрібно для підприємств з різних сфер діяльності зі своїми унікальними властивостями, розрахунками і звітами. Тому універсальні системи часто не вирішують задачі так, як того потребує бізнес. Для вирішення конкретних задач створюються CRM системи для визначеної сфери діяльності або, навіть, для конкретної компанії.

У даній роботі представлено розробку CRM системи для конкретної компанії у сфері житлово-комунальних послуг (ЖКГ) зі зберіганням інформації за допомогою електронних таблиць компаніїGoogle (Google Sheets).

Актуальність. CRM системи зараз є дуже популярним напрямом розробки програмного забезпечення, адже така система може в декілька разів пришвидшити бізнес процеси компанії. Також вона дозволяє звільнити співробітників від монотонної роботи та зосередитися на інтелектуальній діяльності.

У даній роботі додатковий інтерес представляє спосіб зберігання інформації, з якою працює система. Зберігання інформації буде відбуватися не реляційній базі даних, а в експериментальному варіанті за допомогою електронних таблиць компанії Google.

Google таблиці (англ. Google Sheets) – програма електронних таблиць, що входить до складу безкоштовного веб-програмного пакету програмного забезпечення, пропонованого компанією Google у межах служби Google Drive.

Метою роботи є зменшення витрат на зберігання даних та скорочення часу на виконання бізнес процесів шляхом розробки системи з безкоштовним варіантом зберігання даних за допомогою таблицьGoogle.

Вирішувані задачі:

– обробка та трансформування даних компанії за останні 6 років (для формування архіву);

– розробка методу з безкоштовним варіантом зберігання даних;

– розробка програмної системи для реалізації запропонованого методу зберігання даних.

У роботі вперше розроблено метод та програмну систему для безкоштовного зберігання даних за допомогою Google Sheets. Система, розроблена за даним методом може зберігати 15 ГБ даних безкоштовно, що є великою перевагою у фінансовому плані, аніж використання відомих СУБД. Різні провайдери пропонують різні тарифи, але одне для них залишається не змінним – зберігання інформації потребує додаткових витрат.

Результати опубліковано у збірнику доповідей XIIІ міжнародної науково-практичної конференція «Інформаційні технології і автоматизація – 2020» [2].

# **1КРИТИЧНИЙ РОЗБІР ІСНУЮЧИХ РІШЕНЬ**

## 1.1 Аналіз проблем користувачів

Для аналізу проблем користувачів та варіантів засобів для вирішення цих проблем використаємо Valuepropositioncanvas.

«Шаблон ціннісної пропозиції (англ.Value proposition canvas) – єдиний цілісний документ, який є розширенням шаблону бізнес моделі, і дозволяє детально роздивитися процес створення цінності для користувача» [3].

Шаблон ціннісної пропозиції складається з 6 частин, причому 3 частини – блок опис профілю клієнта, а інші 3 частини – блок саме пропозиції продукту.

Перший блок складається з таких частин, як:

1. задачі покупця або customer jobs – задачі, які користувач має намір виконати або проблеми, які потребують вирішення;
2. біль покупця або pains – проблеми, з якими стикається покупець в процесі виконання своїх задач;
3. вигоди для покупця або gains – цінності, які покупець бажає отримати за допомогою послуг.

Другий блок включає в себе такі частини, як:

1. продукти чи послуги або products&services – це послуги та продукти, довкола яких будується ваша ціннісна пропозиція;
2. знеболюючі або painrelievers – опис того, як послуги або продукти зменшать проблеми в процесі виконання задач;
3. створення вигоди або gaincreators – опис того, як створюють цінності, які бажає отримати покупець, щоб бути задоволеним.

Створимо шаблон ціннісної пропозиції для нашої компанії у сфері житлово-комунальних послуг.

Customerjobs:

1. компанії має багато неавтоматизованих дрібних бізнес процесів;
2. компанії необхідно зберігати невеликий об’єм інформації.

Pains:

1. компанія виконує велику кількість робіт, які можна автоматизувати;
2. компанія витрачає багато грошей на сховище, яке вони повністю не використовують.

Gains:

1. компанії необхідно зменшити навантаження на співробітників;
2. компанії необхідно економити на зберіганні невеликий об’єму інформації;
3. компанії, яка працює з Microsoft Excel, потрібно швидко адаптуватися до системи.

Products&services:

1. програмна система для автоматизації процесів компанії з використанням бази даних.

Painrelievers:

1. програмна система дозволяє автоматизувати бізнес процеси компанії;
2. зберігання 15 ГБ інформації не потребує оплати.

Gain creators:

1. програмна система дозволяє виконати монотонні роботи для зменшення навантаження на робітників;
2. використання даної бази даних дає можливість економії на зберіганні інформації;
3. використання Google Sheets є дуже схожим з використанням Microsoft Excel.

## 1.2 Аналіз аналогів

У системи є декількааналогів у даній сфері. Розглянемо їх більш детально:

1. Розумний ЖКГ – цифрова платформа для забудовників, керівників підприємств, жителів і постачальників послуг. Технологічна основа для створення єдиного стандарту сервісу, зростання продажів і рентабельності управління житловим фондом;
2. Домовласник – програма для підприємств ЖКГ;
3. Контур.ЖКГ – програма для розрахунку квартплати громадян, ведення паспортного обліку, оформлення квитанційля власників квартир;
4. WS. Колекторське агентство (ЖКГ) – програма для обліку боржників житлово-комунальних послуг;
5. Дім.Контроль – сервіс для управління багатоквартирними будинками і котеджними селищами. Модульна система дозволяє автоматизувати процеси експлуатації та сервісу, а також підвищити фінансову дисципліну абонентів;
6. Розумний мешканець – система автоматизації управління і експлуатації нерухомістю.

Побудуємо таблицю аналізу закритеріями, які важливі для сервісної компанії замовника (табл. 1.1):

1. можливість генераціїквитанцій для боржників;
2. автоматичне щомісячне оновлення рахунків власників;
3. пошук в архіві платежів;
4. роздрукування історії платежів власника;
5. доступ мешканців до інформації про новини та стан оплати послуг;
6. вартість;
7. використання на різних операційних системах.

Таблиця 1.1 – Аналіз аналогів

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Генерація квитанцій | Пошук в історії платежів | Друк історії платежів | Оновлення рахунків | Доступ мешканців до інформації | Вартість | Платформи |
| Розумний ЖКГ | – | + | + | – | + | від 9000 руб./міс | Google Chrome, Android, IOS |
| Домовласник | – | + | – | + | + | від 4900 руб. | Google Chrome, Android, IOS |
| Контур. ЖКГ | – | + | + | – | + | від 6000 руб. | Android, IOS |
| WS. Колекторське агентство | + | + | + | + | – | від 63000 руб. | Windows |
| Дім. Контроль | – | + | – | + | + | від 5000 руб. | Android, IOS |
| Розумний мешканець | – | – | – | – | + | від 7500 руб. | Google Chrome, Android, IOS |
| Власна система  Комфорт24 | + | + | + | + | + | – | Google Chrome, Windows, Linux |

Як ми бачимо лише власна система відповідає вимогам на 100%, томуїї розробка є виправданою.

# 2 ТЕОРЕТИЧНА ЧАСТИНА CRM СИСТЕМИ

Розглянемо більш детально CRM систему у житлово-комунальній сфері з точки зору функціоналу та даних, які зберігаються у системі.

## 2.1 Аналіз даних

### 2.1.1 Загальні відомості про структуру компанії

Для початку розглянемо загальну структуру будинків сервісної компанії. На даний момент обслуговує 6 багатоповерхових будинків, і ще декілька у процесі будівництва. Кожен будинок містив по 96 квартир з початку функціонування сервісної компанії, але зараз є кілька квартир, які були розділені на 2.

Для зручного ведення бухгалтерського обліку раніше використовувалися 3Excelфайли.

Перший файл мав всю необхідну інформацію для ведення обліку стану платежів за утримання будинку (прибирання, ремонт ліфтів, вивіз мусора та ін.) та холодну воду. Введення обліку цих послуг залишилося таким, як і було спочатку функціонування сервісної компанії.

Другий файл у свою чергу містив облік стану платежів за опалення та гарячу воду. З часом облік гарячої води було знято у зв’язку з 2 причинами. Першою причиною було непопулярність у мешканців, так як майже всі установили індивідуальні установки для нагрівання води, таким чином платили за більш дешеву холодну воду і витрачали на нагрівання менше коштів, аніж оплата гарячої води. Другим фактором було дороге обслуговування і перевірки кожного року.

Третій файл містив інформацію для обліку платежів за електроенергію. З часом облік електроенергію було знято у зв’язку з юридичними питаннями.

Розглянемо загальну структуру даних Excel файлів. Кожен в файлів ділився на 7 сторінок, відповідно до 6 будинків та 1 сторінка для підсумкових даних усіх будинків. Кожен файл по строкам містив квартири, а по стовпцям характеристики щодо стану оплати з відповідної послуги (холодна вода та ін.).

Однією з вимог було мінімум змін в бухгалтерському обліку системи для швидкої адаптації працівників, так як ми будемо використовувати схожу на Microsoft Excel систему (Google Sheets), тому за основу можемо візьмемо стару структуру та видалити всі невикористовувані характеристики, залишивши лише актуальні дані. Так як файл з електроенергією більше не використовується то у нас залишається лише 2 перші файли.

Популярною практикою є використання стандартизованих методологічних прийомів бухгалтерського обліку – дебету та кредиту. В даних файлах дебет означає скільки мешканець повинен грошей компанії, а кредит – скільки компанія повинна мешканцю. Другий випадок можна побачити, коли мешканець переплачує, або платить наперед. Дебет та кредит є як на початку місяця, так і наприкінці, і називається відповідно сальдо на початок місяця та сальдо на кінець місяця.

### 2.1.2 Детальний опис характеристиккомунальних послуг

Розглянемо детальну структуру характеристик файлу обліку стану платежів за утримання будинку та холодну воду (табл. 2.1).

Таблиця 2.1 – Характеристики файлу обліку стану платежів за утримання будинку та холодну воду

|  |  |  |
| --- | --- | --- |
| Характеристика, міра | Опис характеристики | Група |
| *NF* | Номер квартири | Службова |
| *N* | Номер облікового запису |
| *Name* | ПІБ власника |
| *D1*, грн | Дебет на початок місяця | Сальдо на початок місяця |
| *C1*, грн | Кредит на початок місяця |
| Продовження таблиці 2.1 | | |
| Характеристика, міра | Опис характеристики | Група |
| *S*, м2 | Площа квартири | Утримання будинку |
| *Sum1*, грн | Нарахування за утримання будинку за місяць |
| *CW*, куб | Останні показники лічильника води | Холодна вода |
| *PW*, куб | Попередні показники лічильника води |
| *WV*, куб | Об’єм використаної води |
| *Sum2*, грн | Нарахування за холодну воду за місяць |
| *Sum3*, грн | За послуги (нарахування за місяць) | Разом за послуги |
| *P*, грн | Пільги |
| *Sum4*, грн | Разом до оплати |
| *K*, грн | Каса | Оплата |
| *B*, грн | Банк |
| *D2*, грн | Дебет на кінець місяця | Сальдо на кінець місяця |
| *C2*, грн | Кредит на кінець місяця |

Для розрахунків нам необхідно знати тарифи для утримання будинку – загальний (*RGW*) та спеціальний (*RSW*), а також тариф для води(*RW*). За спеціальним тарифом за утримання будинку платять мешканці перших поверхів, тобто перші 6 квартир (для них не враховуються оплата для за обслуговування ліфтів). Тариф за холодну воду не регулюється сервісною компанією, тому він є загальним для всіх будинків. У свою чергу тарифи для утримання будинку, як загальний, так і спеціальний для кожного будинку різний і це зв’язано з тим,що будинки мають різні прибудинкові території для прибирання, а також різне обладнання (різне споживання електроенергії).

Для розрахунку нарахування за утримання будинку за місяць в залежності від номеру квартири використовуємо наступні формули:

*Sum1= S × RGW*, якщо *NF*> 6 (3.1)

*Sum1= S× RSW*, якщо*NF*<= 6 (3.2)

Об’єм використаної води розраховується на наступною формулою:

*WV = CW – PW* (3.3)

Для розрахунку нарахування за холодну воду за місяць використаємо формулу:

*Sum2 = WV× RW* (3.4)

Загалом послуги за місяць будуть розраховуватися як сума нарахування за холодну воду та утримання будинку:

*Sum3 = Sum1 + Sum2* (3.5)

Сума до сплати за місяць розраховується наступним чином (з урахуванням боргу на початок місяця):

*Sum4 = Sum3 – Р + D1 – С1* (3.6)

Останні значення для розрахунку це дебет та кредит на кінець місяця, які враховують оплату протягом місяця:

*Т = Sum4 – К – В,* (3.7)

якщо*Т > 0,* то *D2 = Т, С2 = 0*

якщо*Т < 0,* то *D2 = 0, С2 = Т × (–1)*

якщо*Т = 0,* то *D2 = 0, С2 = 0*

Розглянемо детальну структуру характеристик файлу обліку стану платежів за опалення (табл. 2.2).

Таблиця 2.2 – Характеристики файлу обліку стану платежів за опалення

|  |  |  |
| --- | --- | --- |
| Характеристика, міра | Опис характеристики | Група |
| *NF* | Номер квартири | Службова |
| *N* | Номер облікового запису |
| *Name* | ПІБ власника |
| *D1*, грн | Дебет на початок місяця | Сальдо на початок місяця |
| *C1*, грн | Кредит на початок місяця |
| *SL*, м2 | Житлова площа квартири | Опалення |
| *Туре* | Тип теплового лічильника |
| *CH* | Останні показники теплового лічильника |
| *PH* | Попередні показники теплового лічильника |
| *HV* | Об’єм |
| *Sum1*, грн | За послуги (нарахування за місяць) | Разом за послуги |
| *P*, грн | Пільги |
| *Sum2*, грн | Разом до оплати |
| *K*, грн | Каса | Оплата |
| *B*, грн | Банк |
| *D2*, грн | Дебет на кінець місяця | Сальдо на кінець місяця |
| *C2*, грн | Кредит на кінець місяця |

Для розрахунків нам необхідно знати також тарифи для опалення – загальний (*RGН*) та спеціальний (*RSН*). За спеціальним тарифом за опалення платять мешканці,які мають теплові лічильники, а інші, які платять по квадратним метрам використовують загальний тариф.

Зараз у різних мешканців установлені 4 типи теплолічильників, які ведуть показники у 4 величинах, а саме у гігакалоріях (Гкал), гігаджоулях (ГДж), кіловатах (кВт) та мегаватах (МВт). Таким чином маємо:

*Туре = { Гкал, ГДж, кВт, МВт}* (3.8)

Для кожного типу теплолічильника є своя формула розрахунку нарахування за опалення. Також своя формула розрахунку є і для тих, хто платить за опалення за площею (відсутній тепловий лічильник).

Для нарахування за опалення необхідно знайти використаний об’єм, в залежності від типу теплолічильника:

*НV = CН – PН* (3.9)

Для розрахунку нарахування за опалення за місяць в залежності від типу теплового лічильника використовуємо наступні формули:

*Sum1= HW× RSH× 1.1*, якщо*Type = Гкал* (3.10)

*Sum1= HW× RSH× 1.1/ 4.187*, якщо*Type = ГДж* (3.11)

*Sum1= HW× RSH× 1.1/ 1162.2*, якщо*Type = кВт* (3.12)

*Sum1= HW× RSH× 1.1× 0.86*, якщо*Type = МВт* (3.13)

*Sum1= SL× RGH*, якщо значення *Type* відсутнє (3.14)

Сума до сплати за місяць розраховується наступним чином (з урахуванням боргу на початок місяця):

*Sum2 = Sum1 – Р + D1 – С1* (3.15)

Останні значення для розрахунку це дебет та кредит на кінець місяця, які враховують оплату протягом місяця:

*Т = Sum2 – К – В,* (3.16)

якщо *Т > 0,* то*D2 = Т, С2 = 0*

якщо *Т < 0,* то*D2 = 0, С2 = Т × (–1)*

якщо *Т = 0,* то*D2 = 0, С2 = 0*

## 2.2 Аналіз функціоналу

У даному підрозділі детально розглядається процес генерації квитанцій та оновлення рахунків.

Спочатку розглянемо генерацію квитанцій. Для початку виконання необхідно отримати інформацію з Google Sheets. Результатом генерації буде файл MicrosoftWordіз заповненими квитанціями для мешканців будинків (один файл – один будинок) за шаблоном представленим нижче (табл. 2.3).

Таблиця 2.3 – Шаблон квитанції

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **КВИТАНЦІЯ** | | **Інформація щодо компанії, телефон** | | | | | | | | |
|  | | | | | | | | | | |
| **{NM} ( обліковий запис : {AD} )** | | | | | | | | | | |
| **Адреса** | | | **{FA} за {MT} {YR}рік** | | | | | | | |
|  | **Борг**  **(на 01.{MS}.{YS})** | | | **Поточні показники** | **Попередні показники** | **Об’єм** | **Тариф**  **(за 1 Гкал)** | **Сума** | **Оплата** | **Борг**  **(на 01.{ME}.{YE})** |
| **Опалення** | **{HSS}** | | | **{CHV}** | **{PHV}** | **{HV}** | **{HR}** | **{FH}** | **{HP}** | **{HSE}** |
| **Утримання будинку** | **{WRSS}** | | | **-** | **-** | **-** | **{WRR}** | **{FWR}** | **{WRP}** | **{WRSE}** |
| **Холодна вода** | **-** | | | **{CWV}** | **{PWV}** | **{WV}** | **{WTR}** | **{FWT}** | **{WTP}** | **-** |
| **При оплаті комунальних послуг в обов’язково зазначайте адресу!** | | | | | | | | | | |

У даному шаблоні замість значень у фігурних скобках буде підставлятися відповідне значення для облікового запису.

Для пришвидшення процесу генерації можна буде виконувати заповнення даних в кількох потоках, так як вони є незалежними процесами.

Тепер розглянемо алгоритм дій для оновлення рахунків. Дану функцію необхідно буде виконувати на початку кожного місяця і лише один раз, тому першою дією є виконання перевірки наявності переходу у поточному місяці. Далі необхідно зберегти копії поточних файлів обліку платежів (утримання будинку (разом з холодною водою) та опалення), ці файли є звітом за місяць. Наступним кроком є додавання платежів до архіву, у якому буде вестися пошук платежів. Останнім етапом є редагування поточних файлів обліку.

Під редагуванням поточних файлів розуміється виконання наступних дій над даними:

* 1. у файлі з інформацією про утримання будинку та холодну воду:
* *D1 = D2*(для зберігання боргу станом на кінець місяця);
* *C1 = C2*(для зберігання боргу станом на кінець місяця);
* *PW = СW*(для зберігання останніх показників води);
* *P = 0*(для анулювання місячних пільг);
* *K = 0*(для анулювання оплати (у касі) у новому місяці);
* *B = 0*(для анулювання оплати (через банк) у новому місяці).
  1. у файлі з інформацією про опалення:
* *D1 = D2*(для зберігання боргу станом на кінець місяця);
* *C1 = C2*(для зберігання боргу станом на кінець місяця);
* *PН = СН* (для зберігання останніх показників теплолічильників);
* *P = 0* (для анулювання місячних пільг);
* *K = 0* (для анулювання оплати (у касі) у новому місяці);
* *B = 0* (для анулювання оплати (через банк) у новому місяці).

При виконанні зазначених дій інші показники файлів автоматично оновляться за формулами представленими у попередньому підрозділі.

# 3 ФОРМУВАННЯ ВИМОГ ДО CRM СИСТЕМИ

## 3.1 Формування функціональних вимог

Функціональні вимоги регламентують функціонування або поведінку системи. Функціональні вимоги відповідають на запитання «що має робити система» (але не «як має працювати система») в тих або інших ситуаціях. Функціональні вимоги встановлюють цілі, задачі та сервіси, які надаються системою замовнику, і визначають роботу розробника [4].

Функціональні вимоги записуються у вигляді варіантів використання – популярний і досить продуктивний спосіб надання вимог.

За основу для діаграми варіантів використання візьмемо критеріями, які важливі для сервісної компанії замовника.

Система поділена на 2 основні проекти. Першим проектом є настільне застосування, яке має можливості адміністрування системою. Другим проектом є вебзастосування для доступу мешканців до інформації.

### 3.1.1 Функціональні вимоги настільного застосування

Діаграма варіантів використання настільного застосування системи представлена на риc. 3.1.

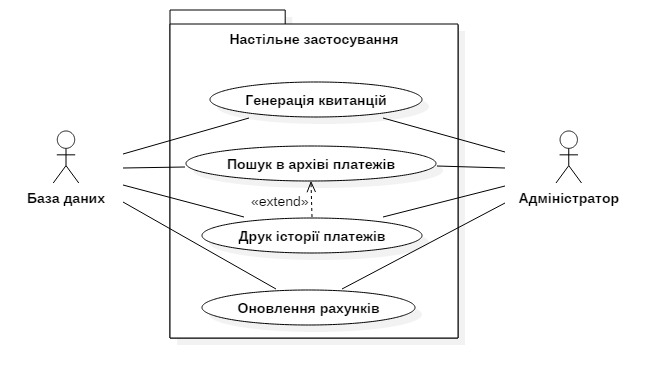


Рисунок 3.1 – Діаграма варіантів використання настільного застосування

Сценарії варіантів використання.

1. Назва прецеденту: генерація квитанцій.

Актори: адміністратор, Google Sheets, настільне застосування.

Ініціатор: адміністратор.

Мета: згенерувати квитанції для всіх власників.

Передумова: наявність підключення до мережі Інтернет.

Післяумова: адміністратор успішно згенерував квитанції.

Основний успішний сценарій:

1. адміністратор переходить до вкладки з генерацією;
2. адміністратор натискає кнопку згенерувати квитанції;
3. настільне застосування звертається до сервісу Google Sheets для отримання інформації щодо стану оплати;
4. настільне застосування починає формувати квитанції, заповнюючи отриманою інформацією. Настільне застосування виводить «Створення квитанцій …%»;
5. квитанції успішно згенеровано. Настільне застосування виводить відповідне повідомлення на екран.

Альтернативний сценарій:

3.1) відсутній Інтернет. Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2;

3.2) перевищено кількість спроб для доступу до сервісу Google Sheets. Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2;

4.1) відсутній шаблон квитанції. Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2.

2. Назва прецеденту: пошук в архіві платежів.

Актори: адміністратор, Google Sheets, настільне застосування.

Ініціатор: адміністратор.

Мета: знайти інформацію за останні роки про обліковий запис за ідентифікатором.

Передумова: наявність підключення до мережі Інтернет при першому пошуку.

Післяумова: адміністратор успішно знайшов інформацію в архіві.

Основний успішний сценарій:

1. адміністратор переходить до вкладки з пошуком інформації;
2. адміністратор вводить ідентифікатор облікового запису та натискає на кнопку пошук інформації;
3. настільне застосування звертається до кешу для отримання інформації щодо історії платежів облікового запису;
4. інформацію успішно знайдено. Настільне застосування виводить знайдену інформацію на екран.

Альтернативний сценарій:

2.1) адміністратор ввів некоректний ідентифікатор (під «некоректний» розуміється той, який не складається лише з 4 цифр). Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2;

3.1) інформацію не знайдено у кеші (перший пошук), то звертається до сервісу GoogleSheetsдля отримання інформації та зберігає її в кеші. Настільне застосування переходить до пункту 3;

3.2) відсутній Інтернет та інформацію не знайдено у кеші. Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2;

3.3) адміністратор ввів недійсний ідентифікатор(під «недійсний» розуміється той, якого не існує у системі). Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2.

3. Назва прецеденту: друк історії платежів.

Актори: адміністратор, настільне застосування.

Ініціатор: адміністратор.

Мета: роздрукувати історію платежіввласника обліковий запису.

Післяумова: адміністратор успішно роздрукував інформацію.

Основний успішний сценарій:

1. настільне застосування починає виконання варіанта використання «Пошук в історії платежів»;
2. адміністратор натискає на кнопку роздрукувати інформацію;
3. настільне застосування починає друк;
4. інформація успішно роздрукована. Настільне застосування виводить відповідне повідомлення на екран.

Альтернативний сценарій:

3.1) настільне застосування не знайшло доступних принтерів для друку. Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2.

4. Назва прецеденту: оновлення рахунків.

Актори: адміністратор, GoogleSheets, настільне застосування.

Ініціатор: адміністратор.

Мета: оновити рахунки всіх облікових записів на початку місяця.

Передумова: наявність підключення до мережі Інтернет.

Післяумова: перехід на новий місяць успішно виконано.

Основний успішний сценарій:

1. адміністратор переходить до вкладки оновлення рахунків;
2. адміністратор натискає на кнопку оновити рахунки;
3. настільне застосування звертається до сервісуGoogleSheets для отримання інформації;
4. настільне застосування перевіряє наявність оновлення у поточному місяці;
5. настільне застосування перевіряє коректністьтипів теплолічильників (важливе для відновлення формул у таблиці);
6. настільне застосування копіює звіти за місяць і зберігає їх у відповідній папці на Google Drive;
7. настільне застосування додає інформацію про поточний місяць в архів сервісу Google Sheets. Настільне застосування виводить «Додавання інформації в архів»;
8. настільне застосування виконує необхідні розрахунки та оновлює дані у сервісі Google Sheets. Настільне застосування виводить «Оновлення рахунків х будинку»;
9. оновлення рахунків успішно виконано. Настільне застосування виводить відповідне повідомлення на екран.

Альтернативний сценарій:

3.1) відсутній Інтернет. Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2;

3.2) перевищено кількість спроб для доступу до сервісу Google Sheets. Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2;

4.1) оновлення рахунків у поточному місяці вже було виконано. Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2;

5.1) інформація некоректна щодо теплолічильників. Настільне застосування виводить відповідне повідомлення на екран. Настільне застосування переходить до пункту 2.

### 3.1.2 Варіанти використання вебзастосування системи

Діаграма варіантів використання вебзастосування системи представлена на риc. 3.2.

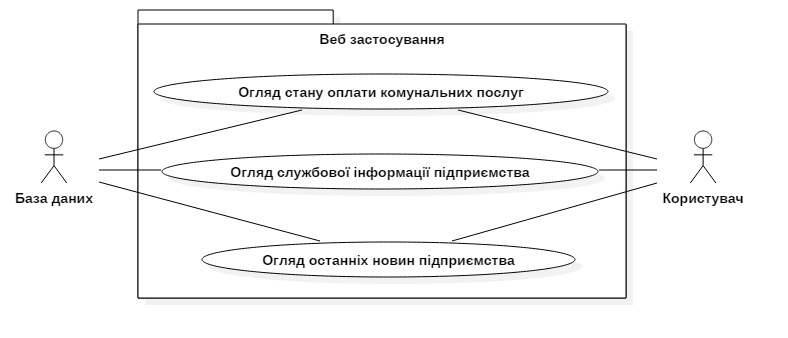


Рисунок 3.2 – Діаграма варіантів використання вебзастосування

Сценарії варіантів використання.

1. Назва прецеденту: огляд стану оплати комунальних послуг.

Актори: користувач, Google Sheets, вебзастосування.

Ініціатор: користувач.

Мета: отримати інформацію щодо стану оплати комунальних послуг за ідентифікатором.

Передумова: наявність підключення до мережі Інтернет.

Післяумова: користувач успішно отримав інформацію.

Основний успішний сценарій:

1. користувач переходить блоку пошуку облікового запису за ідентифікатором;
2. користувач вводить ідентифікатор облікового запису;
3. вебзастосування звертається до сервісу Google Sheets для отримання інформації щодо стану оплати облікового запису;
4. інформацію успішно знайдено. Вебзастосування виводить знайдену інформацію на екран.

Альтернативний сценарій:

2.1) користувач ввів некоректний ідентифікатор. вебзастосування виводить відповідне повідомлення на екран. Вебзастосування переходить до пункту 2;

3.1) втрачено зв'язок з Інтернетом. Вебзастосування виводить відповідне повідомлення на екран. Вебзастосування переходить до пункту 2.

2. Назва прецеденту: огляд службової інформації підприємства.

Актори: користувач, Google Sheets, вебзастосування.

Ініціатор: користувач.

Мета: отримати службову інформацію щодо підприємства (години роботи, реквізити для оплати комунальних послуг).

Передумова: наявність підключення до мережі Інтернет.

Післяумова: користувач успішно отримав інформацію.

Основний успішний сценарій:

1. користувач переходить блоку службової інформації;
2. вебзастосування звертається до сервісу Google Sheets для отримання службової інформації;
3. інформацію успішно знайдено. Вебзастосування виводить оновлену інформацію на екран.

Альтернативний сценарій:

2.1) втрачено зв'язок з Інтернетом. Вебзастосування виводить стандартну службову інформацію.

3.1) інформація відсутня. Вебзастосування виводить стандартну службову інформацію.

3. Назва прецеденту: огляд останніх новин підприємства.

Актори: користувач, Google Sheets, вебзастосування.

Ініціатор: користувач.

Мета: отримати останні новини щодо підприємства.

Передумова: наявність підключення до мережі Інтернет.

Післяумова: користувач успішно отримав новини.

Основний успішний сценарій:

1. користувач переходить блоку з новинами;
2. вебзастосування звертається до сервісу Google Sheets для отримання останніх новин;
3. інформацію успішно знайдено. Вебзастосування виводить оновлену інформацію на екран.

Альтернативний сценарій:

2.1) втрачено зв'язок з Інтернетом. Вебзастосування виводить «На сайті буде публікуватися актуальна інформація».

3.1) інформація відсутня. Вебзастосування виводить «На сайті буде публікуватися актуальна інформація».

## 3.2 Формування нефункціональних вимог

Нефункціональні вимоги регламентують внутрішні та зовнішні умови або атрибути функціонування системи. Виділяють такі основні групи нефункціональних вимог:

– зовнішні інтерфейси (найбільш важливим є інтерфейс користувача);

– атрибути якості (застосовність, надійність, продуктивність, експлуатаційна придатність);

– обмеження.

**Зовнішні інтерфейси.**

Програмна система має мати стриманий і простий інтерфейс користувача для як настільного застосування, так і для вебзастосування. Система для сервісної компанії не має містити розважальний контент. Для вебзастосування можливе використання картинок та іконок для придання застосуванню пристойного вигляду, як на екранах з великим розширенням, так і на екрані мобільного пристрою.

**Атрибути якості.**

Застосовність. Вебзастосування зможе допомагати жителю житлового комплексу, який матиме змогу дізнатися інформацію щодо оплати не виходячи з дому. Настільне застосування дозволить автоматизувати монотонну роботу колективу в офісі.

Надійність. Система оперує відкритої інформацією, тому створення авторизації не потрібно. Але для роботи системи має використовуватися протокол з шифруванням задля підвищення довіри користувачів в браузері.

Продуктивність. Максимальне навантаження на систему прийдеться на кінець місяця, при переході на новий місяць. Але навіть на максимальному навантаженні не потребується великої кількості ресурсів, тому до продуктивності системи не висувається спеціальних вимог.

Експлуатаційна придатність. Дана система може оновлюватися за необхідності для додавання нових функцій, при цьому зберігаючи старий функціонал.

**Обмеження.**

В даний період не розробляється мобільних застосувань, але для мобільних пристроїв планується розробити особливий дизайн сайту для перегляду інформації у мобільному браузері.

# 4 ПРОЕКТУВАННЯ CRM СИСТЕМИ

## 4.1 Проектування архітектури системи

У CRM системі сервісної компанії присутні кілька архітектур на різних рівнях, розглянемо їх детальніше.

### 

### 4.1.1 Високорівнева архітектура системи

На високому рівні архітектура системи є клієнт-серверною, де клієнтами виступають настільне застосування та вебзастосування, а сервером – сервіс Google Sheets.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти набір серверів, набір клієнтів, мережу (забезпечує взаємодію між клієнтами та серверами).

Модель функціонування системи полягає в наступному: клієнт робить запит серверу, сервер отримує запит, виконує його і відсилає результат клієнту.

В залежності від того, як між серверною і клієнтською частиною розподілені функції розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Система, що розробляється в даній роботі зосереджує більшість бізнес – логіки на клієнті.

Отже, клієнти даної системи товстими, так як велика кількість функцій зосереджена саме на клієнтській стороні, а відповідно сервер – тонкий, тому

що він потрібен для отримання інформації.

Система складається з 5 проектів, а саме:

1. Desktop– це проект настільного застосування. Проект необхідний для адміністрування системою, у ньому знаходиться весь функціонал автоматизації, а також роботи з архівними даними;
2. GoogleLib – це проект бібліотеки класів, в якому реалізовано ідентифікацію і зв’язування з сервісами Google;
3. Models–це проект бібліотеки класів, в якому містяться загальні для всієї системи класи моделі предметної області;
4. SheetsEF – це проект бібліотеки класів, який містить у собі функціонал для автоматичного оновлення інформації з Google Sheets кожні 5 хвилин і додавання її до кешу для максимальної швидкості виконання запиту;
5. Web – це проект вебзастосування. Він містить в собі як Frontend, так і Backend частину. Проект необхідний для доступу мешканців до інформації щодо стану оплати комунальних послуг.

На рис. 4.1 наочно показано, як зв’язані проекти системи.

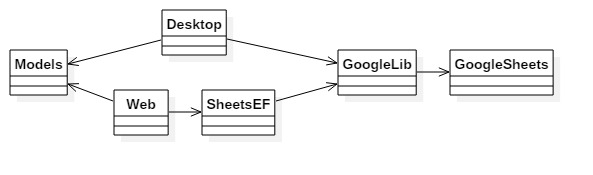


Рисунок 4.1 – Діаграма високорівневої архітектури системи

### 4.1.2 Архітектура настільного застосування

Архітектура настільного застосування будується на основі архітектурного шаблону MVVM.

Шаблону MVVM (Model-View-ViewModel) дозволяє відокремити логіку застосування від візуальної частини.

Даний шаблон був представлений Джоном Госсманом в 2005 році як модифікація шаблону PresentationModel і був спочатку націлений на розробку застосувань в WPF. І хоча зараз цей шаблон вийшов за межі WPF і застосовується в самих різних технологіях, в тому числі при розробці під Android, iOS.

MVVM складається з трьох компонентів: моделі (Model), моделі подання (ViewModel) і уявлення (View).

**Model.**Модель описує використовувані в застосуванні дані. Моделі можуть містити логіку, безпосередньо пов'язану цими даними, наприклад, логіку валідації властивостей моделі. У той же час модель не повинна містити ніякої логіки, пов'язаної з відображенням даних і взаємодією з візуальними елементами управління.

Нерідко модель реалізує спеціальні інтерфейси, які дозволяють повідомляти систему про зміни властивостей моделі. Завдяки цьому полегшується прив'язка до візуальної частини, хоча знову ж пряму взаємодію між моделлю і представленням відсутня.

**View.**View або представлення визначає візуальний інтерфейс, через який користувач взаємодіє із застосуванням. Представлення не виконує жодних події за рідкісним винятком, а виконує дії в основному за допомогою команд.

**ViewModel.** ViewModel або модель уявлення пов'язує модель і уявлення через механізм прив'язки даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю спеціальних інтерфейсів автоматично відбувається зміна відображуваних даних в представленні, хоча безпосередньо модель і представлення не пов'язані.

ViewModel також містить логіку по отриманню даних з моделі, які потім передаються в представлення. І також ViewModel визначає логіку по оновленню даних в моделі.

Оскільки елементи представлення, тобто візуальні компоненти типу кнопок, не використовують події, то уявлення взаємодіє з ViewModel за допомогою команд.

Отже, причинами вибору шаблону MVVM є функціональний розподіл програми на три компонента, які простіше розробляти і тестувати, а також в подальшому модифікувати і підтримувати.

### 4.1.3 Архітектура вебзастосування

Вебзастосування будується на основі клієнт-серверної архітектури, де клієнтом виступаєFrontendчастина, а сервером – Backend частина вебзастосування.

У даному випадку клієнт вебзастосування є товстим, так як функціонал з обробки даних знаходиться саме на клієнтській стороні, а відповідно сервер – тонкий, тому що він потрібен для передачі інформації з сервісу Google Sheets до клієнта вебзастосування.

## 4.2 Проектування структури і організації класів

Для візуалізації організації класів будемо використовувати діаграми класів.

«Діаграма класів – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення» [5].

### 4.2.1 Організація класів настільного застосування

Організація класів настільного застосування повинна відповідати обраній архітектурі MVVM. Візуалізацію відношень між класами настільного застосування можна побачити на діаграмі класів на рис. 4.2.

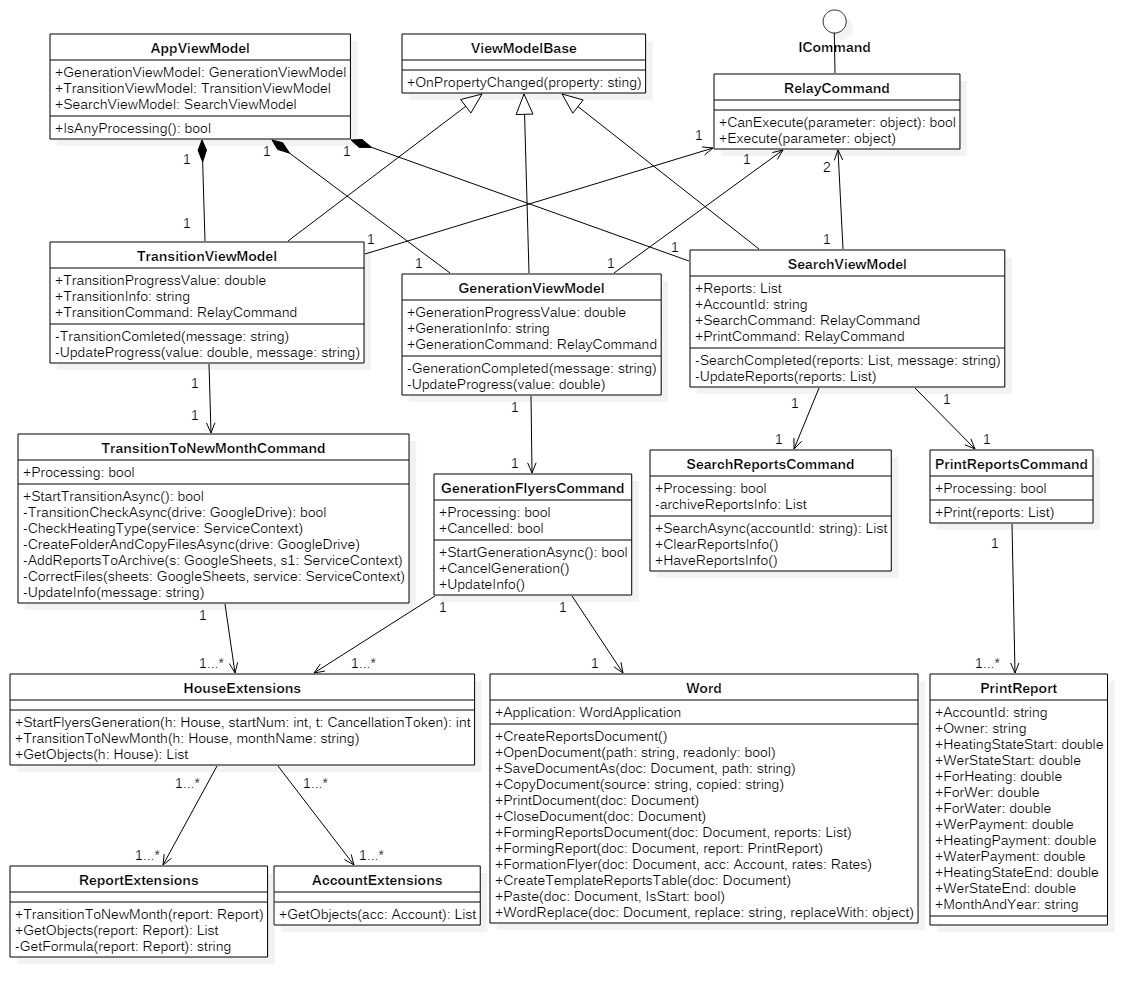


Рисунок 4.2 – Діаграма класів настільного застосування

З назв класів на діаграми видно, що проект ділиться на декілька частин.

**ViewModel**частина представлена наступними класами:

* 1. ViewModelBase – базовий клас для ViewModel і містить загальну структуру для ViewModel класів;
     + OnPropertyChanged(string) – метод для оновлення екрану відповідно до значень змінних.
  2. AppViewModel – клас, який містить посилання на всі ViewModel у проекті;
     + IsAnyProcessing() – метод для перевірки, чи відбувається якась з операцій.
  3. GenerationViewModel – клас, який відповідає за взаємодію представлення з командою генерації квитанцій;
     + GenerationCommand – запускає команду генерації квитанцій;
     + GenerationCompleted(string) – метод, який виконує необхідні дії після завершення генерації квитанцій;
     + UpdateProgress(double) – метод для оновлення полоси завершення генерації.
  4. TransitionToNewMonthViewModel – клас, який відповідає за взаємодію представлення з командою оновлення даних;
     + TransitionCommand – запускає команду оновлення рахунків.
  5. SearchViewModel – клас, який відповідає за взаємодію представлення з командами пошуку та друку інформації.
     + SearchCommand – запускає команду пошуку платежів;
     + PrintCommand – запускає команду друку історії платежів.

**Command** частина представлена наступними класами:

* 1. RelayCommand – основний клас для команд, який реалізує методи з ІCommand;
  2. GenerationCommand – клас відповідальний за генерацію квитанцій;
     + GenerationStartAsync() – метод для запуску процесів генерації квитанцій для всіх будинків, кожен в окремому потоці.
  3. TransitionToNewMonthCommand – клас відповідальний за оновлення даних;
     + StartTransitionAsync() – метод для запуску процесу оновлення рахунків для всіх будинків.
  4. SearchCommand – клас відповідальний за пошук платежів в архіві;
     + SearchAsync() – метод для запуску процесупошуку платежів.
  5. PrintCommand – клас відповідальний за друк платежів.
     + Print() – метод для запуску процесу друкування платежів.

**Model** частина представлена бібліотекою класів Models та наступним класом:

1. PrintReport– клас для формування інформації для друкування.

**Extensions**частина представлена для розширенняфункціональності відповідних класів, представлена наступними класами:

1. HouseExtensions – клас, який додає необхідну функціональність у клас моделі House;
   * + StartFlyersGeneration() – метод для запуску процесу генерації квитанцій для конкретного будинку;
     + TransitionToNewMonth() - метод для запуску процесу оновлення рахунків для конкретного будинку.
2. AccountExtensions – клас, який додає необхідну функціональність у клас моделіAccount;
3. ReportExtensions – клас, який додає необхідну функціональність у клас моделіReport;

**Utilities**частина представлена наступним класом:

1. Word – клас, який містить в собі весь необхідний функціонал для роботи з MicrosoftWord(для квитанцій або файлу для друку).
   * + FormingFlyer(Document, Account, Rates) - метод для формування однієї квитанціїдля конкретного облікового запису;
     + FormingReportsDocument(Document, List) – метод для формування документа з історією платежів для конкретного облікового запису.

**View**частина представлена розміткою на XAML (розширення для XML).

### 4.2.2 Організація класів бібліотеки класів Models

Візуалізацію відношень між класами бібліотеки класівModels можна побачити на діаграмі класів на рис. 4.3.

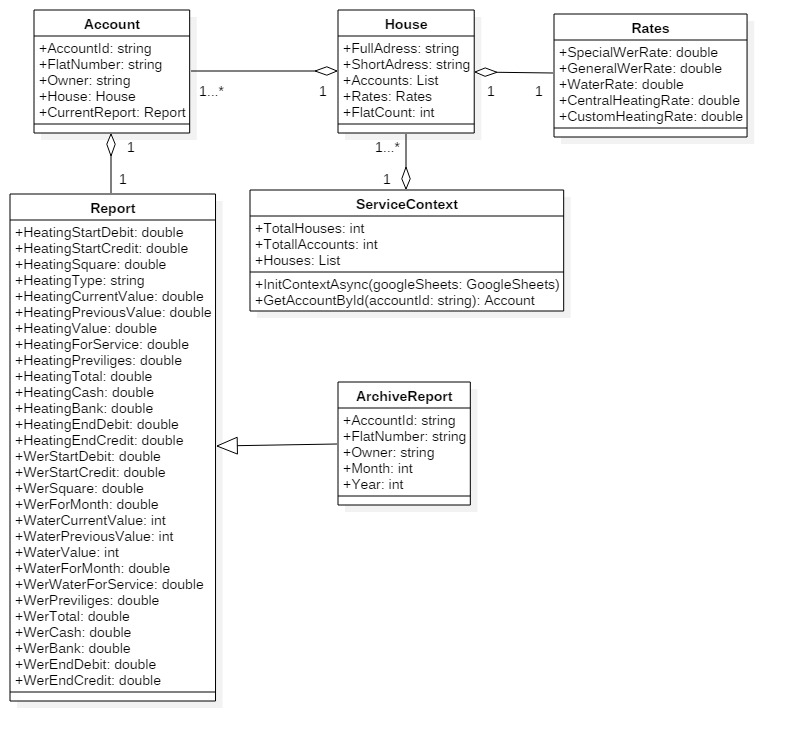


Рисунок 4.3 – Діаграма класів бібліотеки класів Models

Розглянемо класи моделей предметної області загальні для системи:

1. Report – клас для зберігання поточного стану характеристик комунальних послуг;
2. ArchiveReport – клас для зберігання стану характеристик комунальних послуг у відповідний період. Є нащадком класу Report. Даний клас є моделлю архіву у системі;
3. Account – клас для зберігання інформації щодо облікового запису;
   * + AccountId – унікальний номер облікового запису;
     + Owner – ПІБ власника квартири;
     + Number – номер квартири.
4. House – клас для зберігання інформації щодо будинку;
   * + FullAdress –повна адреса будинку;
     + Accounts – облікові записи у даному будинку;
     + Rates – значення тарифів для конкретного будинку.
5. Rates – клас для зберігання інформації щодо тарифів для будинку;
6. ServiceContext – клас для ініціалізації та зручного доступу до класів моделей предметної області.
   * + Houses – список будинків у системі;
     + GetAccountById(string) – метод, який виконує пошук облікового запису у всій будинках.

### 4.2.3 Організація класів бібліотеки класів GoogleLib

Візуалізацію відношень між класами бібліотеки класів GoogleLib можна побачити на діаграмі класів на рис. 4.4.

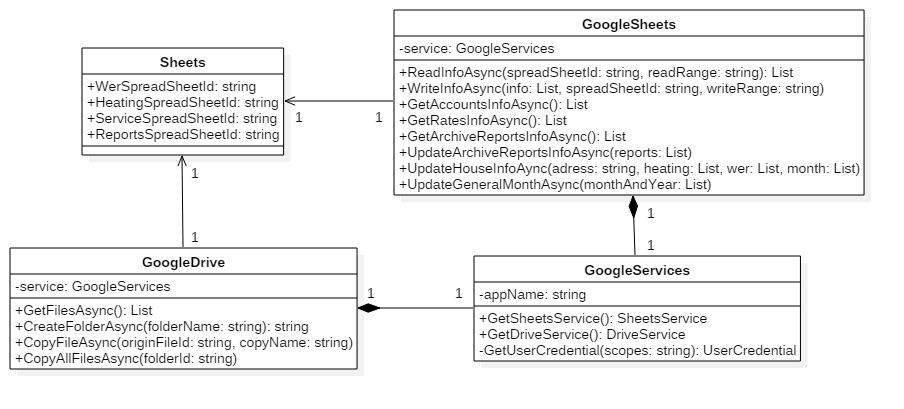


Рисунок 4.4 – Діаграма класів бібліотеки класів GoogleLib

Розглянемо класи бібліотеки для підключення до сервісів Google:

1. Sheets – клас для зберігання ідентифікаторів таблиць з даними для системи;
2. GoogleDrive – клас, який містить в собі весь функціонал для роботи з сервісом GoogleDrive;
   * + CreateFolderAsync(string) – метод, який створює папку з відповідною назвою на платформі GoogleDrive;
     + CopyAllFilesAsync(string) – метод, який виконує копіювання поточних файлів, таким чином, створюючи звіт.
3. GoogleSheets – клас, який містить в собі весь функціонал для роботи з сервісом Google Sheets;
   * + ReadInfoAsync(string, string) – метод, який виконує отримання інформації з конкретного місця електронної таблиці;
     + WriteInfoAsync(string,string, List) – метод, який записує інформацію в конкретне місце електронної таблиці.
4. GoogleServices – клас для ідентифікації користувача у сервісах Google.
   * + GetUserCredential(string) – метод, який виконує ідентифікацію користувача, згідно з правами доступу.

### 4.2.4 Організація класів бібліотеки класів SheetsEF

Візуалізацію відношень між класами бібліотеки класів SheetsEF можна побачити на діаграмі класів на рис. 4.5.

Розглянемо класи бібліотеки автоматичногооновлення даних у певний період:

1. ServiceCollectionExtensions – клас для зручного доступу до функціональності автоматичного оновлення даних у іншому проекті;
2. ApplicationContextBase –базовий клас для контексту Google Sheets, тобто це клас через який буде відбуватися взаємодія з отриманими даними. Містить узагальнені реалізації, тому потребує спадкування;
   * + GetDataFromSheetsAsync(string) – метод, отримує інформацію з електронних таблиць для вебзастосування;
     + UpdateSheetsAsync() – метод, який виконує оновлення даних для вебзастосування.

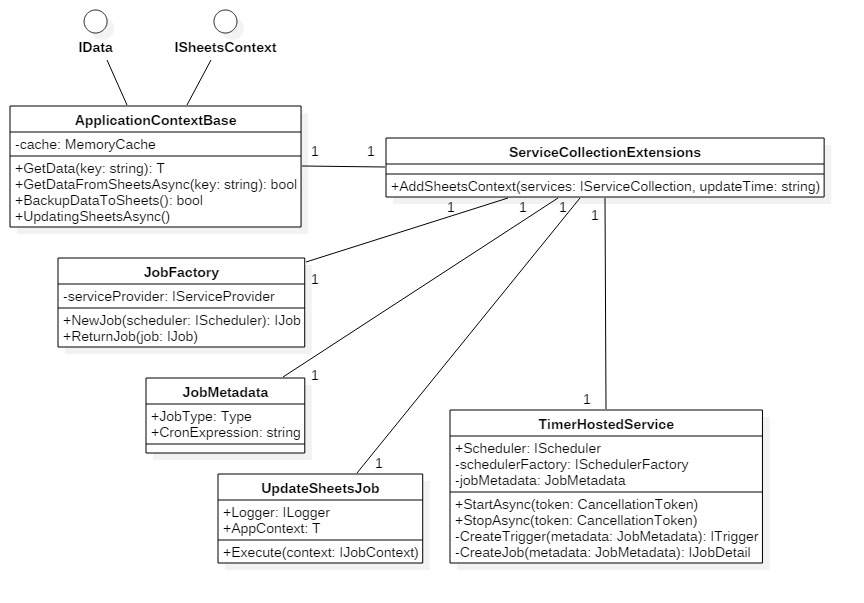


Рисунок 4.5 – Діаграма класів бібліотеки класів SheetsEF

1. JobFactory – клас для створення об’єктів класу UpdateSheetsJob та JobMetadata. Створення об’єктів буде виконуватися згідно з розкладом;
2. JobMetadata – клас для службової інформації щодо об’єкту виконання;
3. UpdateSheetsJob – клас, що містить функціонал для оновлення даних з таблиць;
4. TimerHostedService – клас, який кожні 5 хвилин буде запускати оновлення даних з таблиць.
   * + StartAsync() – метод, який запускає оновлення даних для вебзастосування за розкладом.

### 4.2.5 Організація класів вебзастосування

Вебзастосування ділиться на 2 частини – Frontend і Backend. Візуалізацію відношень Backend частини можна побачити на діаграмі класів на рис. 4.6.

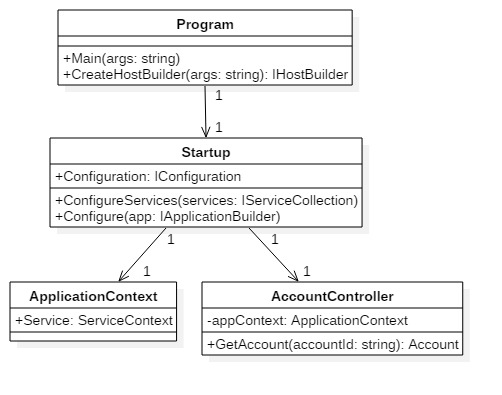


Рисунок4.6 – Діаграма класів Backend частини вебзастосування

Розглянемо класи бібліотеки Backend частини:

1. Program – головний клас, саме з нього починається виконання вебзастосування. На основі класу Startup створює об’єкт, який буде ідентифікувати проект як вебзастосування;
   * + Main(string) – метод, який стартує виконання вебзастосування.
2. Startup – клас, який визначає яким чином буде оброблюватися запит до веб серверу;
   * + Configure(IApplicationBuilder) – метод, який відповідає за функціонування обробки запитів;
     + ConfigureServices(IServiceProviderCollection) – метод, який відповідає за зручний доступ до змінних.
3. ApplicationContext – клас, який є нащадком ApplicationContextBase, і містить реалізацію з конкретними типами даних, які необхідно отримати з GoogleSheets;
4. AccountController – клас, який відповідає за обробку вхідних запитів до серверу.
   * + appContext – змінна, яка відповідає за доступ до функціоналу роботи з даними;
     + GetAccount(string) – метод для пошуку облікових записів у всіх будинках.

Візуалізацію відношень Frontend частини представлено діаграмою класів на рис. 4.7.

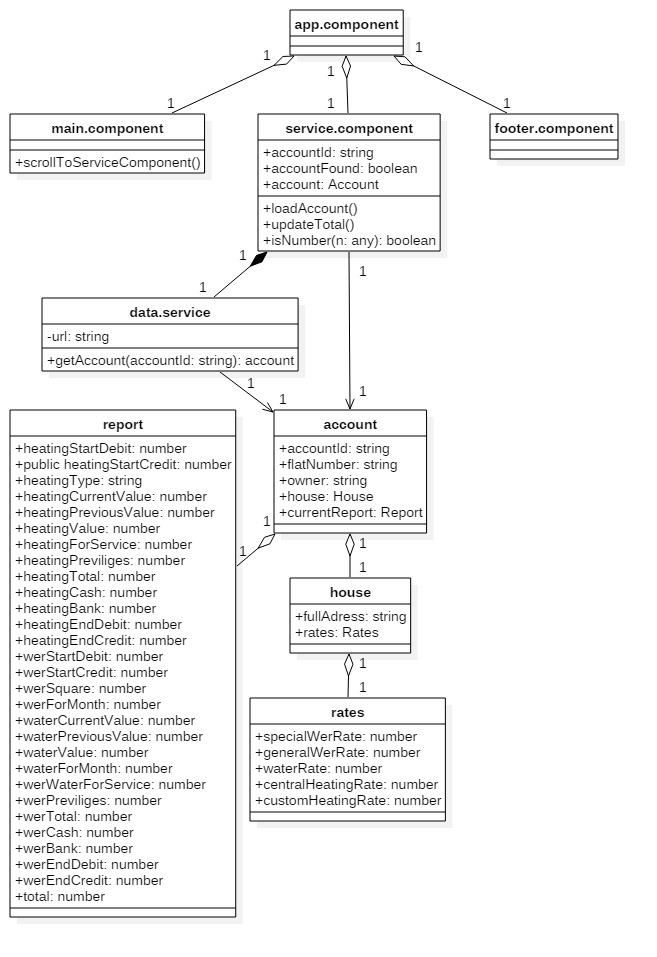


Рисунок 4.7 – Діаграма класів Frontend частини вебзастосування

Розглянемо компоненти Frontend частини:

1. App.component – головний компонент, містить у собі всі інші компоненти;
2. Main.component – компонент головного екрану, містить блок з останніми новинами;
3. Service.component – компонент для запиту і відображення інформації щодо стану оплати комунальних платежів з сервера;
   * + loadAccount() – метод, для завантаження і виведення інформації на екран;
     + updateTotal() – метод, який оновлює дані на сторінці, коли користувач вводить нові дані (наприклад, поточний стан показників лічильника води).
4. Footer.component – компонент для відображення службової інформації щодо роботи сервісної компанії;
5. Data.service – сервіс, який використовується для запиту інформації з сервера;
   * + getAccount() – метод, який відправляє запит на сервер для отримання облікового запису.
6. Account – клас для зберігання інформації щодо облікового запису;
7. House – клас для зберігання інформації щодо будинку;
8. Report – клас для зберігання поточного стану характеристик комунальних послуг;
9. Rates – клас для зберігання інформації щодо тарифів для будинку.

## 4.3 Проектування інтерфейсу користувача CRM системи

«Проектування інтерфейсу користувача – створення загальної моделі інтерфейсу з описом всіх значущих блоків системи і переходами між ними» [6]. Він повинен відповідати трьом найважливішим критеріям:

1. ефективність – наскільки успішно продукту вдається сприяти користувачеві у виконанні свого завдання;
2. швидкість – скільки дій знадобиться користувачу для пошуку і виконання свого завдання;
3. простота вивчення – наскільки швидко користувач зможе почати використовувати продукт.

У системі є 2 інтерфейси. Перший для настільного застосування і цей інтерфейс буде бачити лише адміністратор системи. Другий для вебзастосування буде загальнодоступним для всіх мешканців будинків.

### 4.3.1 Проектування інтерфейсу настільного застосування

Проектування інтерфейсунастільного застосування почнемо навігаційної панелі. У застосуванні ми маємо 3 функціональні блоки:

1. генерація квитанцій;
2. оновлення рахунків;
3. пошук та друк платежів (друк розширює функціональність пошуку, тому їх було вирішено об’єднати в одинблок).

Кожен з них є відповідає за свою функціональність тому розташування їх на одному екрані можуть викликати перевантаження інформацією. В даному випадку можна зробити кожен функціональний блок окремою вкладкою на панелі навігації. Так як вкладок небагато, то зручно використовувати звичне для користувачів розташування вкладок зверху. Такий підхід узгоджується з правилом 7 ± 2 (водночас на інтерфейсі не повинно знаходитися більш, ніж 9 елементів), але у той самий час, коли потрібно перейти на іншийблок застосування це можна зробити швидко та за допомогою одного кліку.

Інтерфейс програми потрібно візуально узгоджувати з інтерфейсом операційної системи, тому що це забезпечує користувачу передбачуване робоче середовище. Саме тому будемо використовувати стандартні, звичні для користувача, шрифти.

Тепер треба обрати кольорову гаму настільного застосування. Так як застосуваннямбудуть користуватися кожен день, то потрібно використовувати ненасічені світлі кольори, які не будуть набридати адміністратору. Таким кольором є блакитний та його відтінки. Для гарного контрасту на блакитномуфоні для написів підійде чорний колір, але не чисто чорний (#000000), краще візьмемо темний відтінок сірого кольору, наприклад (#121212).

### 4.3.2 Проектування інтерфейсу вебзастосування

Проектування інтерфейсу вебзастосування почнемо з розташовування блоків елементів. В вебзастосуванні є декілька блоків, а саме:

1. головний блок – блок, який першим з’являється на екрані і повинен містити логотип компанії та актуальні новини щодо комунальних послуг;
2. блок послуг – блок, який повинен містити строку пошуку облікового запису та після вводу коректного ідентифікатора відображати у таблиці поточний стан його оплати;
3. блок службової інформації – блок, який містить інформацію щодо графіку роботи сервісної компанії, телефону, а також розрахункових рахунків.

Інформації для розміщення у головному блоці інтерфейсу вебзастосування не дуже багато, тому було вирішено розділити екран навпіл для розміщеннянеобхідної інформації у лівій частині екрану, а фото одного з будинків у правій (рис. 4.8).

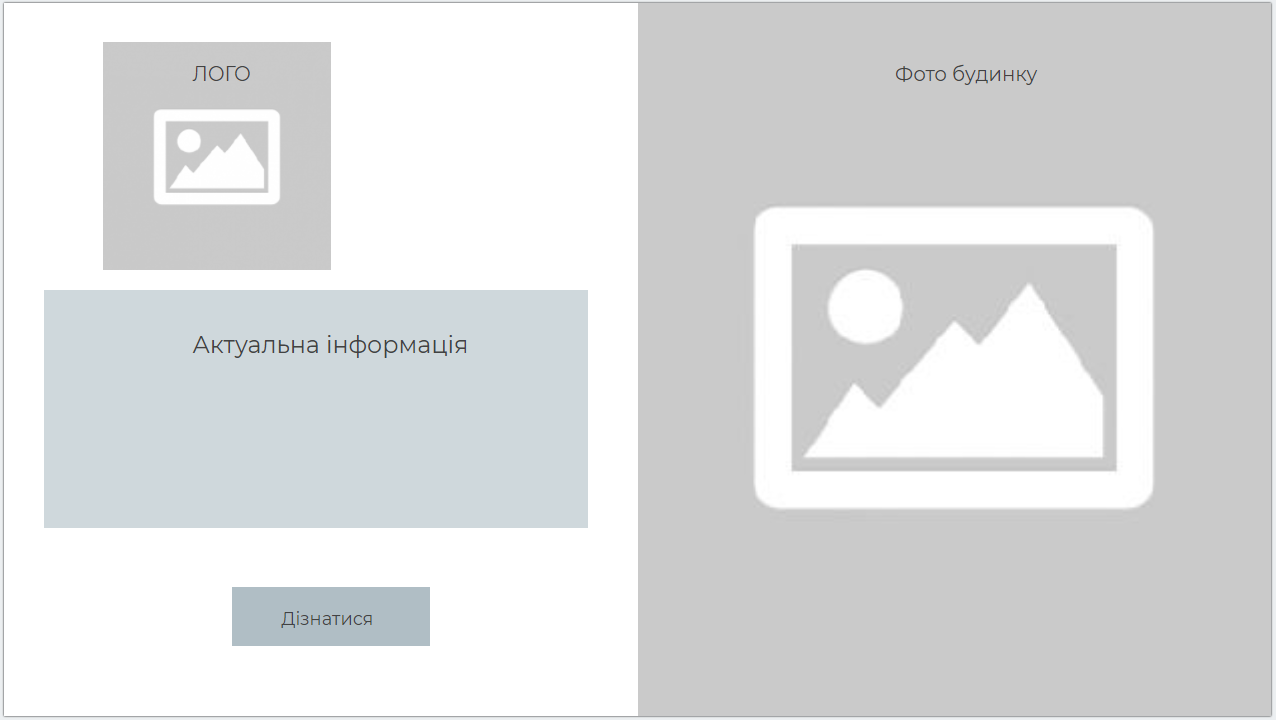


Рисунок 4.8 – Макет головного блоку вебзастосування

Дизайн інтерфейсу блоку послуг виконано у мінімалістичному стилі, лише строку пошуку, після вводу коректного ідентифікатора з’явиться таблиця з усіма параметрами оплати комунальних платежів (рис. 4.9).

Блок службової інформації (футеру) буде розміщуватися знизу сторінки та мати кілька стовпців (рис. 4.9).

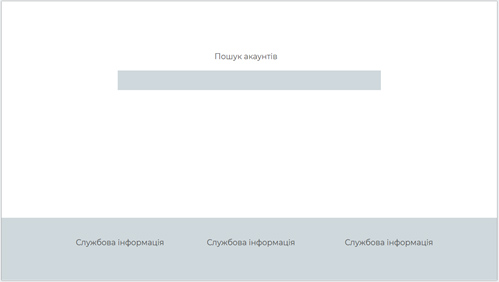


Рисунок4.9 – Макет блокупослуг та службової інформації вебзастосування

Вже зараз більшість запитів в Інтернеті здійснюються зісмартфонів та планшетів (56%) [7], тому дуже важливо зробити дизайн адаптивним, щоб навіть з мобільних пристроїв було зручно взаємодіяти з вебзастосуванням.

У дизайні відбудуться наступні зміни (рис. 4.10):

1. права частина головного блоку (фотографія будинку) перейде нагору, а ліва частина відповідно перейде донизу;
2. у футері інформація розташується в одну колонку;
3. пропорційно зменшуються шрифти та картинки.

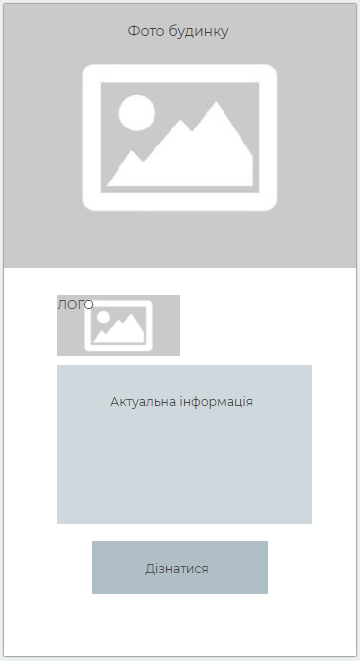


Рисунок 4.10 – Макет блоків вебзастосування на мобільному пристрої

Такий дизайн також узгоджується з правилом 7 ± 2 (водночас на інтерфейсі не повинно знаходитися більш, ніж 9 елементів).

Не менш важливим є підбір іконок вебзастосування. Вони мають бути виконані в єдиному стилі та ретельно підібрані відповідно до тексту, до якого вони відносяться.

Останньою важливою дією є вибір кольорів. Так як настільне і вебзастосування є єдиною системою, то ж доречно їх виконати в одній кольоровій гамі, використовуючи ненасічені світлі кольори блакитного відтінку та текст темного кольору.

# 5 ПРОГРАМНА РЕАЛІЗАЦІЯ CRM СИСТЕМИ

## 5.1Створення проекту для сервісів Google

Створення програмної реалізації потрібно розпочати з реєстрації нового облікового запису у Google, якщо його не існує. Після реєстрації необхідно перейти на платформу GoogleConsoleDeveloper[8]. На платформі потрібно створити проект і ввести назву програмної системи. Потім потрібно перейти до бібліотеки API у проекті і ввімкнути підтримку необхідних API. Для даної CRMсистеми необхідними є GoogleSheetsAPI(для взаємодії з даними)та GoogleDrive API (для створення та видалення таблиць з диска). Останнім кроком на платформі є створення облікових даних (client\_secret.json) для доступу доствореного проекту в програмному коді. Для цього на сторінці однієї з доданих бібліотек треба перейти до налаштування, а потім створити облікові дані. При створенні потрібно вказати наступні характеристики:API – GoogleSheets, тип програми – веб-сервер, використання даних програми. В наступному вікні ввести ім’я облікового запису служби та вибрати розширення .json. Автоматично розпочнеться завантаження файлу, який потрібно буде додати у папкуз проектом програмної системи для взаємодії з доданими Google API.

Далі допроекту програмної системитакож потрібно додати бібліотеки для використання API, а самеGoogleSheetsAPI та GoogleDriveAPI (використовуючи, наприклад, Nuget). На платформі GoogleDevelopers[9] можна знайти код для використання цих API (є версії для всіх популярних мов програмування).

Є деякі особливості використанняелектронних таблицьGoogleSheets замість бази даних. У цьому випадку файл таблиці розглядається як схема, оскільки він зберігає інформацію в аркушах, де у свою чергурозташовується необхідна інформація. Тому один файл електронної таблиці може містити кілька таблиць на різних аркушах.

Також існує обмеження (квота) кількості запитів за певний час від одного користувача – 100 запитів / 100 секунд. Але дане обмеження ми обійшли на етапі проектування, створивши окрему бібліотеку класів SheetsEF, яка додає інформацію в кеш та оновлює інформацію за розкладом (раз в 5 хвилин).

## 5.2 Набір інструментальних засобів розробки

Для реалізації всіх проектів системи (окрім Frontendчастини) було обрано мову програмування C# (об’єктно-орієнтована мова програмування компанії Microsoft).Дана мова була обрана тому щоразом з технологіями платформи .NET вона дозволяє розробляти як настільні, так і вебзастосування.

«Платформа ASP.NET Core представляє технологію від компанії Microsoft, призначену для створення різного роду вебзастосувань: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів. ASP.NET Core є платформою з відкритим вихідним кодом. Платформа може працює за допомогою кроccплатформного середовища .NET Core, яке може бути розгорнуте на основних популярних операційних системах: Windows, Mac OS X, Linux» [10].

Платформа WPF (WindowsPresentationFoundation) також є частиною екосистеми .NET та представляє собою технологію для створеннякроссплатформенних (починаючи з .NETCore 3.0)настільних застосувань з багатим графічним інтерфейсом. Однією з важливих особливостей є використання мови декларативною розмітки інтерфейсу XAML, заснованого на XML, він дозволяє створювати насичений графічний інтерфейс, використовуючи або декларативне оголошення інтерфейсу, або код на керованих мовах C # і VB.NET, або поєднувати і те, і інше.

Для Frontend частини системиобрано фреймворк Angular. Angular представляє фреймворк від компанії Google для створення клієнтських застосувань. Перш за все він націлений на розробку SPA-рішень (англ. SinglePageApplication), тобто одно сторінкових застосувань. Angular надає таку функціональність, як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому, шаблони, маршрутизація і так далі.

Однією з ключових особливостей Angular (і вибором для Frontend частини системи) є те, що він використовує в якості мови програмування TypeScript.TypeScript представляє мову програмування на основі JavaScript, але є строго типізованою і компільованою мовою. Хоча на виході компілятор створює все той же JavaScript, який потім виконується браузером, однак строга типізація зменшує кількість потенційних помилок, які могли б виникнути при розробці на JavaScript. Також TypeScript реалізує багато концепцій, які властиві об'єктно-орієнтованим мовам, як, наприклад, спадкування, поліморфізм, інкапсуляція, модифікатори доступу і так далі.

Для пришвидшення процесу розробки було використано редактор коду Microsoft VisualStudio Express 2019.Microsoft VisualStudio– лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Дані продукти дозволяють розробляти консольні застосування, застосування з графічним інтерфейсом, а також веб-сайти, веб-застосування, веб-служби, які можуть бути розгорнуті на всіх популярних операційних системах [11].

Для розгортання вебзастосування було обрано Docker. «Docker – система пакетування програмного забезпечення в стандартні блоки для розробки, доставки і розгортання» [12].

Контейнер – це стандартна одиниця програмного забезпечення, яка упаковує код і всі його залежності, тому застосування швидко і надійно переміщається з одного обчислювального середовища в інше.

Образ контейнера Docker – це легкий автономний виконуваний пакет програмного забезпечення, який включає в себе все необхідне для запуску програми: код, середу виконання, системні інструменти, системні бібліотеки та налаштування. Образи контейнерів стають контейнерами під час виконання, а в разі контейнерів Docker - образи стають контейнерами, коли вони працюють в DockerEngine. Контейнерне програмне забезпечення, доступне як для застосувань Linux, так і для Windows, завжди буде працювати однаково, незалежно від інфраструктури. Контейнери ізолюють програмне забезпечення від його середовища і гарантують, що воно працює однаково, незважаючи на відмінності, наприклад, між розробкою і стадією.

Переваги використання Docker:

1. Стандарт. Docker створив галузевий стандарт для контейнерів, тому їх можна переносити куди завгодно;
2. Легковажність. Контейнери спільно використовують системне ядро ОС машини і, отже, не вимагають ОС для кожної програми, що забезпечує більш високу ефективність сервера і знижує витрати на сервер і ліцензування;
3. Безпека. Застосування більш безпечні в контейнерах, а Docker надає найнадійніші можливості ізоляції за замовчуванням в галузі.

## 5.3Набір використаних бібліотек

Для створенняпрограмної системи було використано наступні зовнішні бібліотеки:

1. Microsoft.AspNetCore– набір бібліотек для створення вебзастосувань на мові програмування C#;
2. Microsoft.AspNetCore.SpaServices.Extensions – бібліотека для додавання функціональності створення Frontend іBackendчастини в одному проекті вебзастосування;
3. Microsoft.TypeScript.MSBuild – бібліотека для компіляції Frontend частини;
4. Google.Apis.Sheets.v4 – бібліотека для взаємодії з GoogleSheets;
5. Google.Apis.Drive.v3 – бібліотека для взаємодії з GoogleDrive.

# 6 ТЕСТУВАННЯ ТА РОЗГОРТАННЯ CRM СИСТЕМИ

Для тестування системи будемо використовуватиодночасно функціональне тестування та ручне тестування.

Функціональне тестування (англ. Functionaltesting) – це тестування, метою якого є виявлення невідповідностей між реальною поведінкою реалізованих функцій та очікуваною поведінкою відповідно до специфікацій, складених до програми.

Функціональні тести повинні охоплювати всі реалізовані функції з урахуванням найбільш ймовірних типів помилок. Тестові сценарії, що поєднують окремі тести, орієнтовані на перевірку якості розв’язку функціональних задач.

«Ручне тестування (англ. Manualtesting) – це процес ручної перевірки програмного забезпечення на помилки. Тестувальник має відігравати роль користувача програмної системи й використовувати властивості програми для знаходження помилок у роботі»[13].

## 6.1 Тестування настільного застосування

У таблицях 6.1 – 6.7 представлено функціональні тести для варіантів використання настільного застосування.

У табл. 6.1 показано функціональний тест варіанта використання «Генерація квитанцій».

Таблиця 6.1 – Функціональний тест варіанта використання «Генерація квитанцій»

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| У  м  о  в  а | Відсутнє підключення до мережі Інтернет | T | T | T | T | F | F | F | F |
| Перевищено кількість спроб для доступу до сервісу GoogleSheets | T | T | F | F | T | T | F | F |
| Відсутній шаблон для генерації | T | F | T | F | T | F | T | F |
| Д  і  я | Вивести повідомлення «Відсутнє підключення до мережі Інтернет» | + | + | + | + |  |  |  |  |
| Вивести повідомлення «Перевищено кількість спроб для доступу до сервісу GoogleSheets» |  |  |  |  | + | + |  |  |
| Вивести повідомлення «Відсутній шаблон для генерації» |  |  |  |  |  |  | + |  |
| Отримання інформації з GoogleSheets |  |  |  |  |  |  |  | + |
| Заповнення шаблонів  Вивести повідомлення «Створення квитанцій …%» |  |  |  |  |  |  |  | + |
| Вивести повідомлення «Генерацію успішно виконано» |  |  |  |  |  |  |  | + |

Виконаємо склейку функціонального тесту варіанта використання «Генерація квитанцій» (табл. 6.2).

Таблиця 6.2 – Скорочений функціональний тест варіанта використання «Генерація квитанцій»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| У  м  о  в  а | Відсутнє підключення до мережі Інтернет | T | F | F | F |
| Перевищено кількість спроб для доступу до сервісу GoogleSheets | х | T | F | F |
| Відсутній шаблон для генерації | х | х | T | F |
| Д  і  я | Вивести повідомлення «Відсутнє підключення до мережі Інтернет» | + |  |  |  |
| Вивести повідомлення «Перевищено кількість спроб для доступу до сервісу GoogleSheets» |  | + |  |  |
| Вивести повідомлення «Відсутній шаблон для генерації» |  |  | + |  |
| Отримання інформації з GoogleSheets |  |  |  | + |
| Заповнення шаблонів  Вивести повідомлення «Створення квитанцій …%» |  |  |  | + |
| Вивести повідомлення «Генерацію успішно виконано» |  |  |  | + |

Перевіримо виконання скороченогофункціонального тесту у програмній системі (рис. 6.1 – 6.5). Виконання генерації квитанцій зайняло 5 хвилин 30 секунд, замість приблизно 1 робочого дня (8 годин) без CRMсистеми.

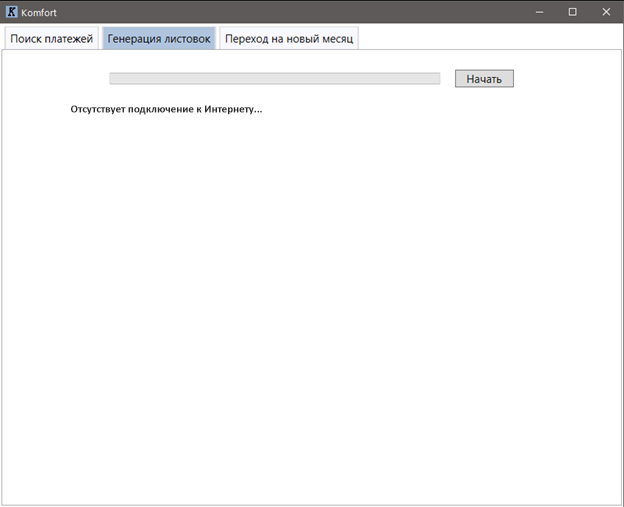


Рисунок6.1 – Відсутнє підключення до мережі Інтернет

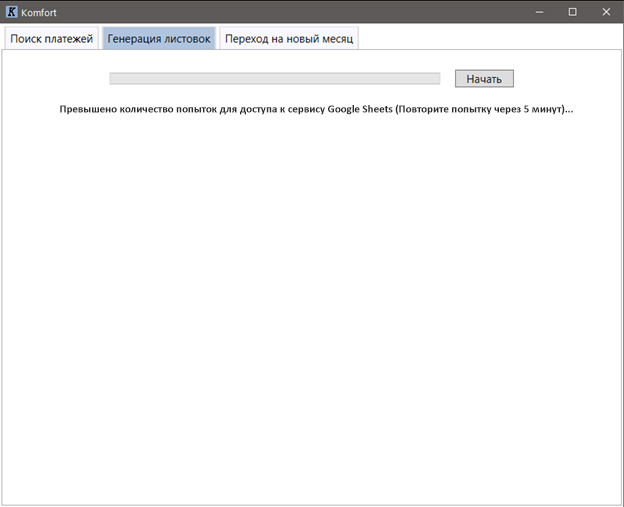


Рисунок 6.2 – Перевищено кількість спроб для доступу до сервісу Google Sheets

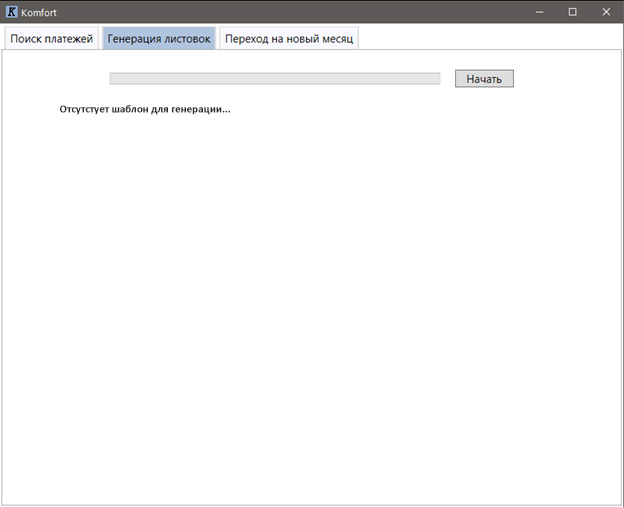


Рисунок 6.3 – Відсутній шаблон для генерації

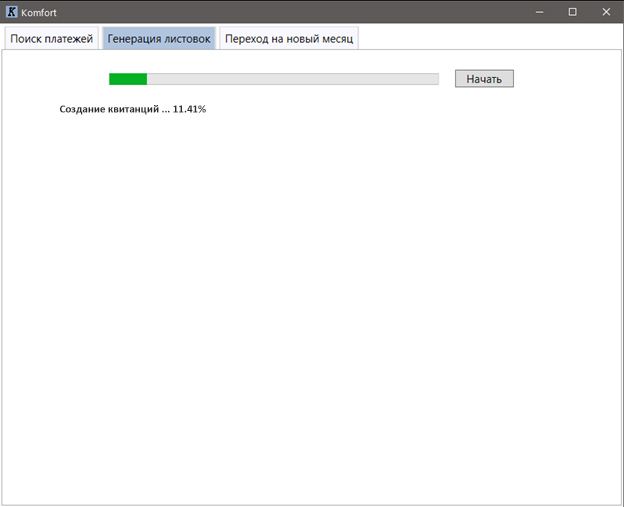


Рисунок 6.4 – Створення квитанцій

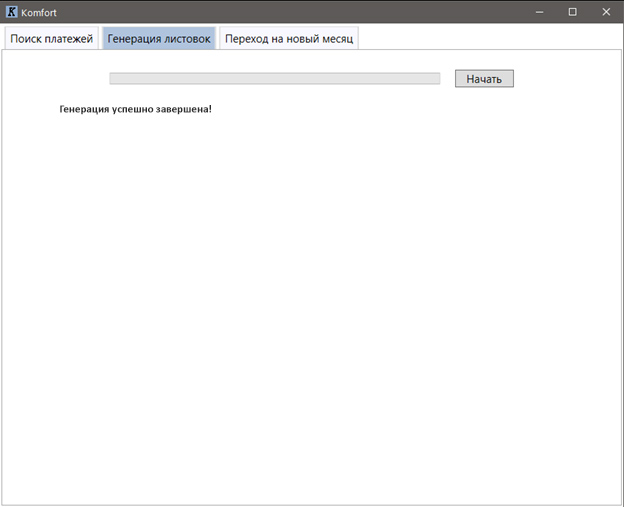


Рисунок 6.5 – Генерацію успішно виконано

У табл. 6.3 показано функціональний тест варіанта використання «Пошук в архіві платежів».

Таблиця 6.3 – Функціональний тест варіанта використання «Пошук в архіві платежів»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| У  м  о  в  а | Адміністратор ввів некоректний ідентифікатор | T | T | T | T | T | T | T | T | F | F | F | F | F | F | F | F |
| Відсутня інформація у кеші (перший пошук) | T | T | T | T | F | F | F | F | T | T | T | T | F | F | F | F |
| Відсутнє підключення до мережі Інтернет | T | T | F | F | T | T | F | F | T | T | F | F | T | T | F | F |
| Адміністратор ввів недійсний ідентифікатор | T | F | T | F | T | F | T | F | T | F | T | F | T | F | T | F |
| Д  і  я | Вивести повідомлення «Некоректний ідентифікатор» | + | + | + | + | + | + | + | + |  |  |  |  |  |  |  |  |
| Отримання інформації з Google Sheets та зберегти у кеші |  |  |  |  |  |  |  |  | + | + | + | + |  |  |  |  |
| Вивести повідомлення «Відсутнє підключення до мережі Інтернет» |  |  |  |  |  |  |  |  | + | + |  |  |  |  |  |  |
| Виконання пошуку інформації в кеші |  |  |  |  |  |  |  |  |  |  |  |  | + | + | + | + |
| Вивести повідомлення «Недійсний ідентифікатор» |  |  |  |  |  |  |  |  |  |  |  |  | + |  | + |  |

Продовження таблиці 6.3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Вивести знайдену інформацію на екран |  |  |  |  |  |  |  |  |  |  |  |  |  | + |  | + |

Виконаємо склейку функціонального тесту варіанта використання «Пошук в архіві платежів» (табл. 6.4).

Таблиця 6.4 – Скорочений функціональний тест варіанта використання «Пошук в архіві платежів»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| У  м  о  в  а | Адміністратор ввів некоректний ідентифікатор | T | F | F | F | F |
| Відсутня інформація у кеші (перший пошук) | х | T | T | F | F |
| Відсутнє підключення до мережі Інтернет | х | T | F | х | х |
| Адміністратор ввів недійсний ідентифікатор | х | х | х | T | F |
| Д  і  я | Вивести повідомлення «Некоректний ідентифікатор» | + |  |  |  |  |
| Отримання інформації з GoogleSheets та зберегти у кеші |  | + | + |  |  |
| Вивести повідомлення «Відсутнє підключення до мережі Інтернет» |  | + |  |  |  |
| Виконання пошуку інформації в кеші |  |  |  | + | + |
| Вивести повідомлення «Недійсний ідентифікатор» |  |  |  | + |  |
| Вивести знайдену інформацію на екран |  |  |  |  | + |

Перевіримо виконання скороченого функціонального тесту у програмній системі (рис. 6.6 – 6.9).Виконання пошуку платежів в архіві займає 10 секунд (при першому пошуку), а при наступних спробах пошук виконується за 0.1 секунди.

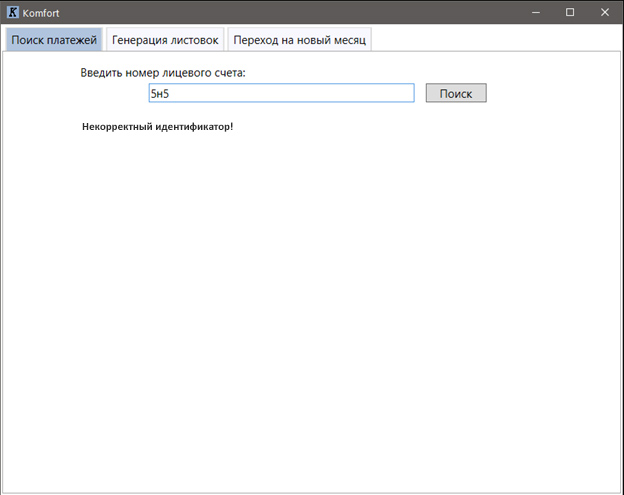


Рисунок6.6 – Некоректний ідентифікатор

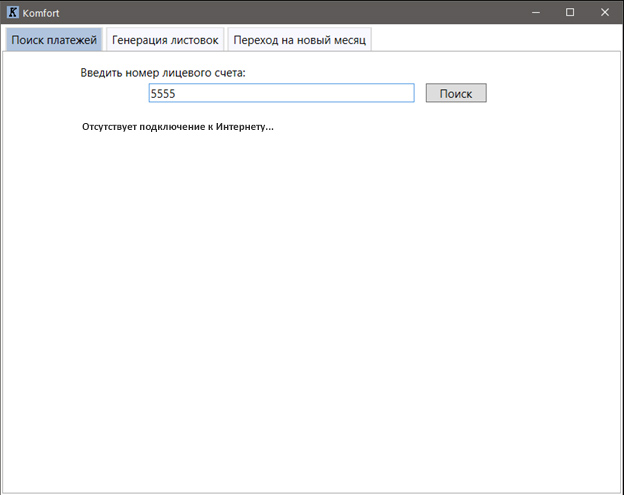


Рисунок6.7 – Відсутнє підключення до мережі Інтернет

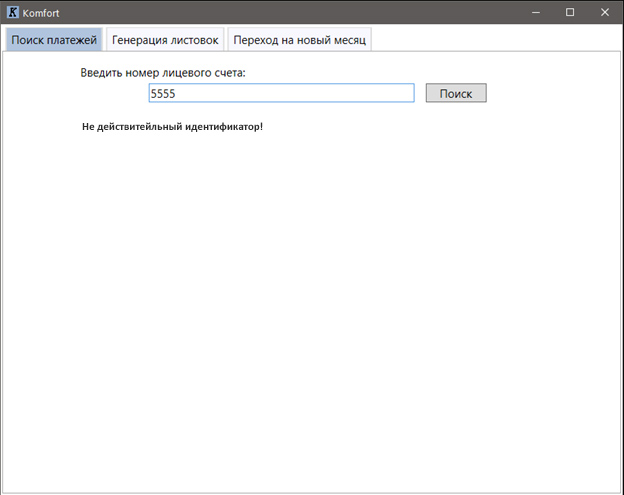


Рисунок 6.8 – Недійсний ідентифікатор

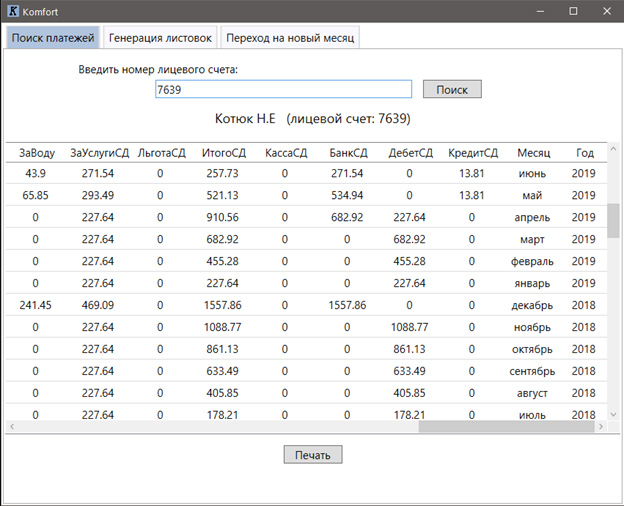


Рисунок 6.9 – Пошук успішно виконано

У табл. 6.5 показано функціональний тест варіанта використання «Друк історії платежів».

Таблиця 6.5 – Функціональний тест варіанта використання «Друк історії платежів»

|  |  |  |  |
| --- | --- | --- | --- |
| У  м  о  в  а | Не знайдено доступних принтерів | T | F |
| Д  і  я | Виконання варіанта використання «Пошук в історії платежів» | + | + |
| Вивести повідомлення «Немає доступних принтерів» | + |  |
| Вивести повідомлення «Виконується друкування документа» |  | + |

У даному функціональному тесті (табл. 6.5) склейку виконувати не потрібно.

Перевіримо виконанняфункціонального тесту у програмній системі(рис. 6.10 – 6.11).

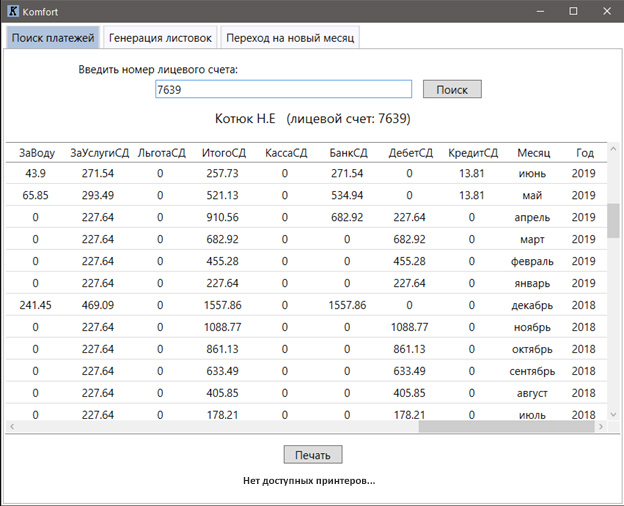


Рисунок 6.10 – Немає доступних принтерів

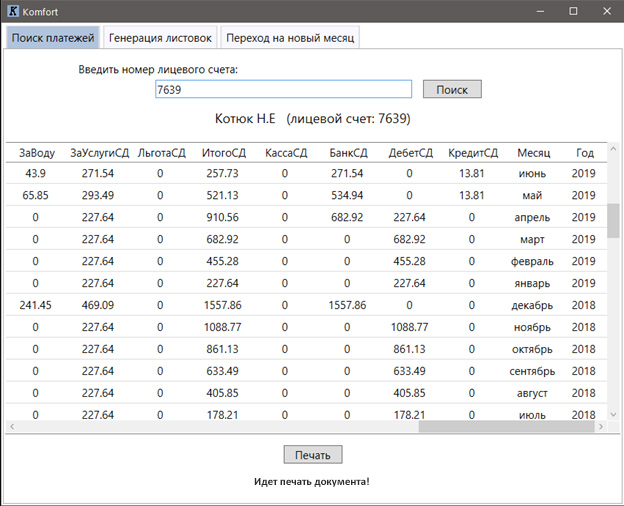


Рисунок 6.11 – Виконується друкування документа

У табл. 6.6 показано функціональний тест варіанта використання «Оновлення рахунків».

Таблиця 6.6 – Функціональний тест варіанта використання «Оновлення рахунків»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| У  м  о  в  а | Оновлення рахунків у поточному місяці вже було виконано | T | T | T | T | T | T | T | T | F | F | F | F | F | F | F | F |
| Відсутнє підключення до мережі Інтернет | T | T | T | T | F | F | F | F | T | T | T | T | F | F | F | F |
| Перевищено кількість спроб для доступу до сервісу Google Sheets | T | T | F | F | T | T | F | F | T | T | F | F | T | T | F | F |
| Некоректна інформація щодо теплолічильників | T | F | T | F | T | F | T | F | T | F | T | F | T | F | T | F |
| Д  і  я | Вивести повідомлення «Відсутнє підключення до мережі Інтернет» | + | + | + | + | + | + | + | + |  |  |  |  |  |  |  |  |
| Вивести повідомлення «Перевищено кількість спроб для доступу до сервісу Google Sheets» |  |  |  |  |  |  |  |  | + | + | + | + |  |  |  |  |
| Вивести повідомлення «Оновлення рахунків у поточному місяці вже було виконано» |  |  |  |  |  |  |  |  |  |  |  |  | + | + |  |  |
| Вивести повідомлення «Виправите тип теплолічильника у обліковому записі х» |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |  |

Продовження таблиці 6.6

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Зберігання звітів у відповідній папці |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |
| Додавання інформації про поточний місяць до архіву  Вивести повідомлення «Додавання інформації в архів» |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |
| Виконання розрахунків щодо оновлення Вивести повідомлення «Оновлення рахунків х будинку» |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |
| Вивести повідомлення «Оновлення рахунків пройшло успішно» |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |

Виконаємо склейку функціонального тесту варіанта використання «Оновлення рахунків» (табл. 6.7).

Таблиця 6.7 – Скорочений функціональний тест варіанта використання «Оновлення рахунків»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| У  м  о  в  а | Оновлення рахунків у поточному місяці вже було виконано | T | F | F | F | F |
| Відсутнє підключення до мережі Інтернет | х | T | F | F | F |
| Перевищено кількість спроб для доступу до сервісу Google Sheets | х | х | T | F | F |
| Некоректна інформація щодо теплолічильників | х | х | х | T | F |
| Д  і  я | Вивести повідомлення «Відсутнє підключення до мережі Інтернет» | + |  |  |  |  |
| Вивести повідомлення «Перевищено кількість спроб для доступу до сервісу Google Sheets» |  | + |  |  |  |
| Вивести повідомлення «Оновлення рахунків у поточному місяці вже було виконано» |  |  | + |  |  |
| Вивести повідомлення «Виправите тип теплолічильника у обліковому записі х» |  |  |  | + |  |
| Зберігання звітів у відповідній папці |  |  |  |  | + |
| Додавання інформації про поточний місяць до архіву  Вивести повідомлення «Додавання інформації в архів» |  |  |  |  | + |
| Виконання розрахунків щодо оновлення  Вивести повідомлення «Оновлення рахунків х будинку» |  |  |  |  | + |
| Вивести повідомлення «Оновлення рахунків пройшло успішно» |  |  |  |  | + |

Перевіримо виконання скороченого функціонального тесту у програмній системі (рис. 6.12 – 6.18). Виконання оновлення рахунків зайняло 2 хвилин 15 секунд, замість приблизно 2 годин роботи співробітника без CRMсистеми.

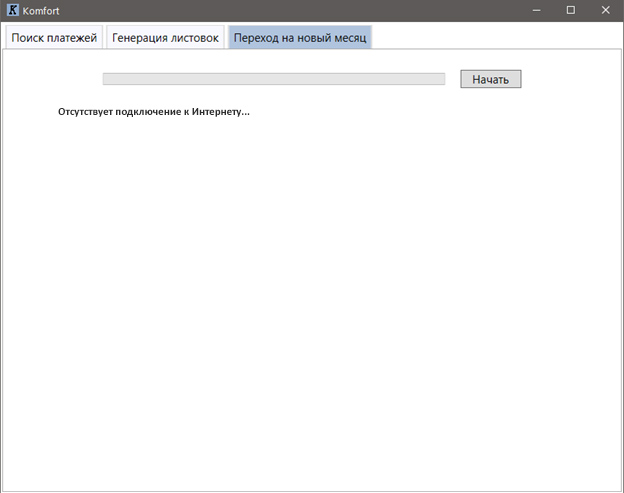


Рисунок 6.12 – Відсутнє підключення до мережі Інтернет

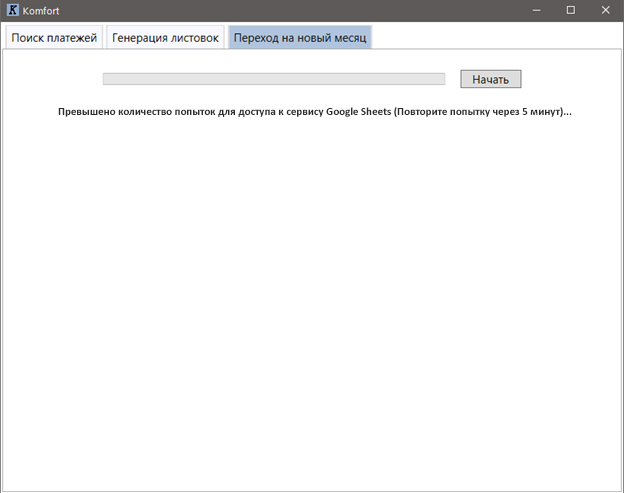


Рисунок 6.13 – Перевищено кількість спроб для доступу до сервісу GoogleSheets

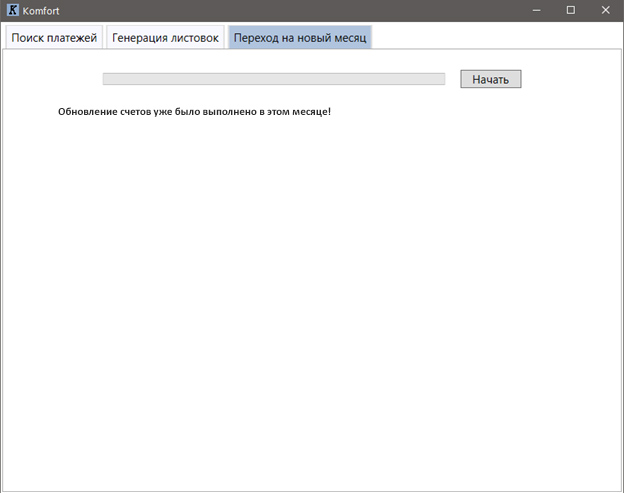


Рисунок 6.14 – Оновлення рахунків у поточному місяці вже було виконано

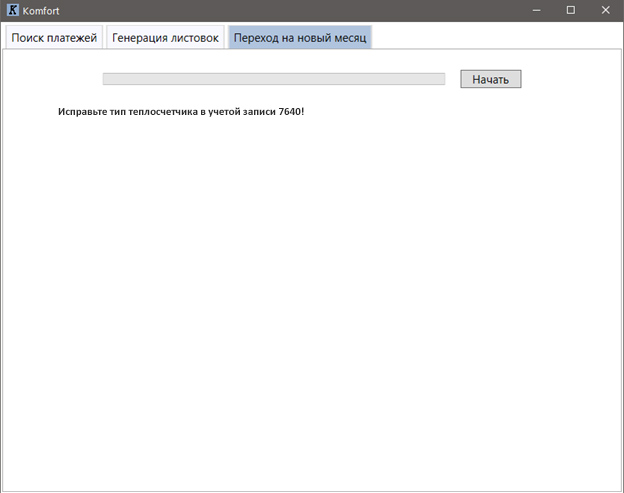


Рисунок 6.15 – Виправите тип теплолічильника у обліковому записі

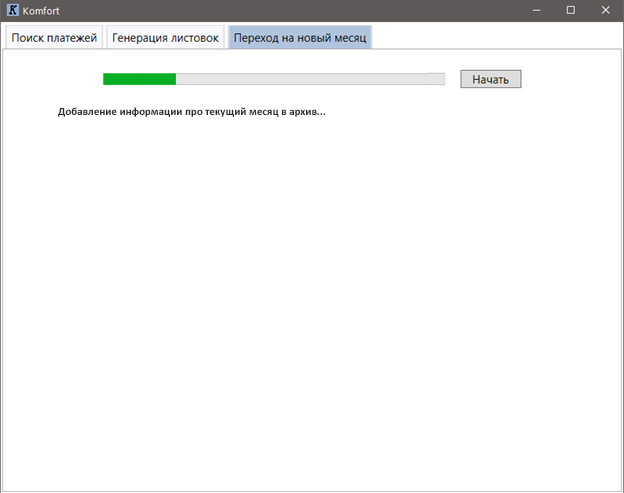


Рисунок 6.16 – Додавання інформації в архів

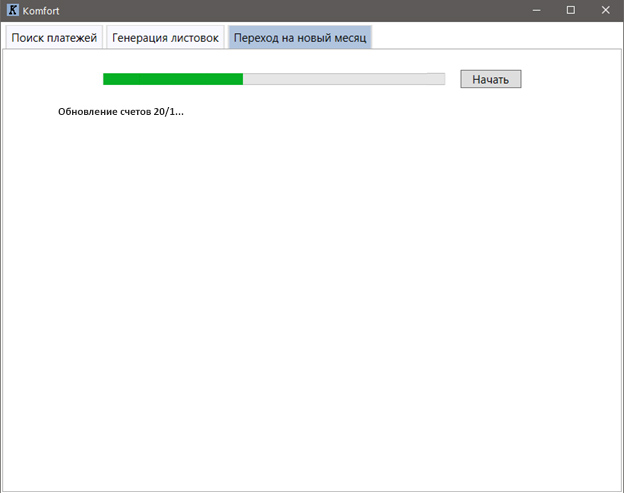


Рисунок 6.17 – Оновлення рахунків будинку

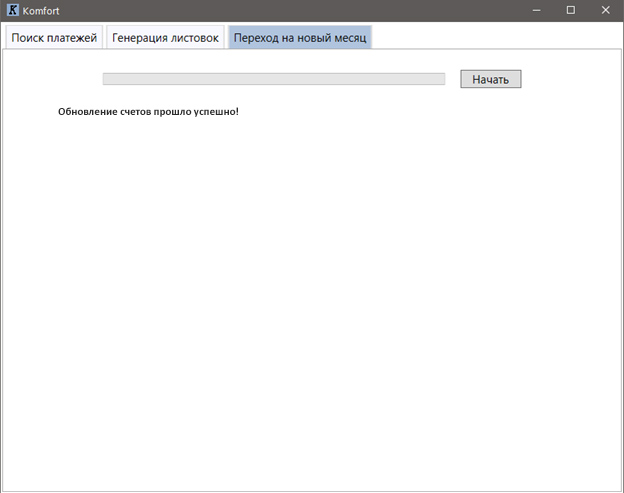


Рисунок 6.18 – Оновлення рахунків пройшло успішно

## 6.2 Тестування вебзастосування

У таблицях 6.8 – 6.13 представлено функціональні тести для варіантів використання вебзастосування.

У табл. 6.8 показано функціональний тест варіанта використання «Огляд стану оплати комунальних послуг».

Таблиця 6.8 – Функціональний тест варіанта використання «Огляд стану оплати комунальних послуг»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| У  м  о  в  а | Користувач ввів некоректний ідентифікатор | T | T | F | F |
| Втрачено зв’язок з Інтернетом | T | F | T | F |
| Д  і  я | Вивести повідомлення «Введено некоректний ідентифікатор» | + | + |  |  |
| Вивести повідомлення «Втрачено зв’язок з Інтернетом» |  |  | + |  |
| Отримання інформації щодо облікового запису |  |  |  | + |
| Вивести отриману інформацію |  |  |  | + |

Виконаємо склейку функціонального тесту варіанта використання «Огляд стану оплати комунальних послуг» (табл. 6.9).

Таблиця 6.9 – Скорочений функціональний тест варіанта використання «Огляд стану оплати комунальних послуг»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| У  м  о  в  а | Користувач ввів некоректний ідентифікатор | T | F | F |
| Втрачено зв’язок з Інтернетом | х | T | F |
| Д  і  я | Вивести повідомлення «Введено некоректний ідентифікатор» | + |  |  |
| Вивести повідомлення «Втрачено зв’язок з Інтернетом» |  | + |  |
| Отримання інформації щодо облікового запису |  |  | + |
| Вивести отримати інформацію |  |  | + |

Перевіримо виконання скороченого функціонального тесту у програмній системі (рис. 6.19 – 6.24). Виконання пошуку облікового запису зайняло 0.4 секунди, тобто майже миттєво.

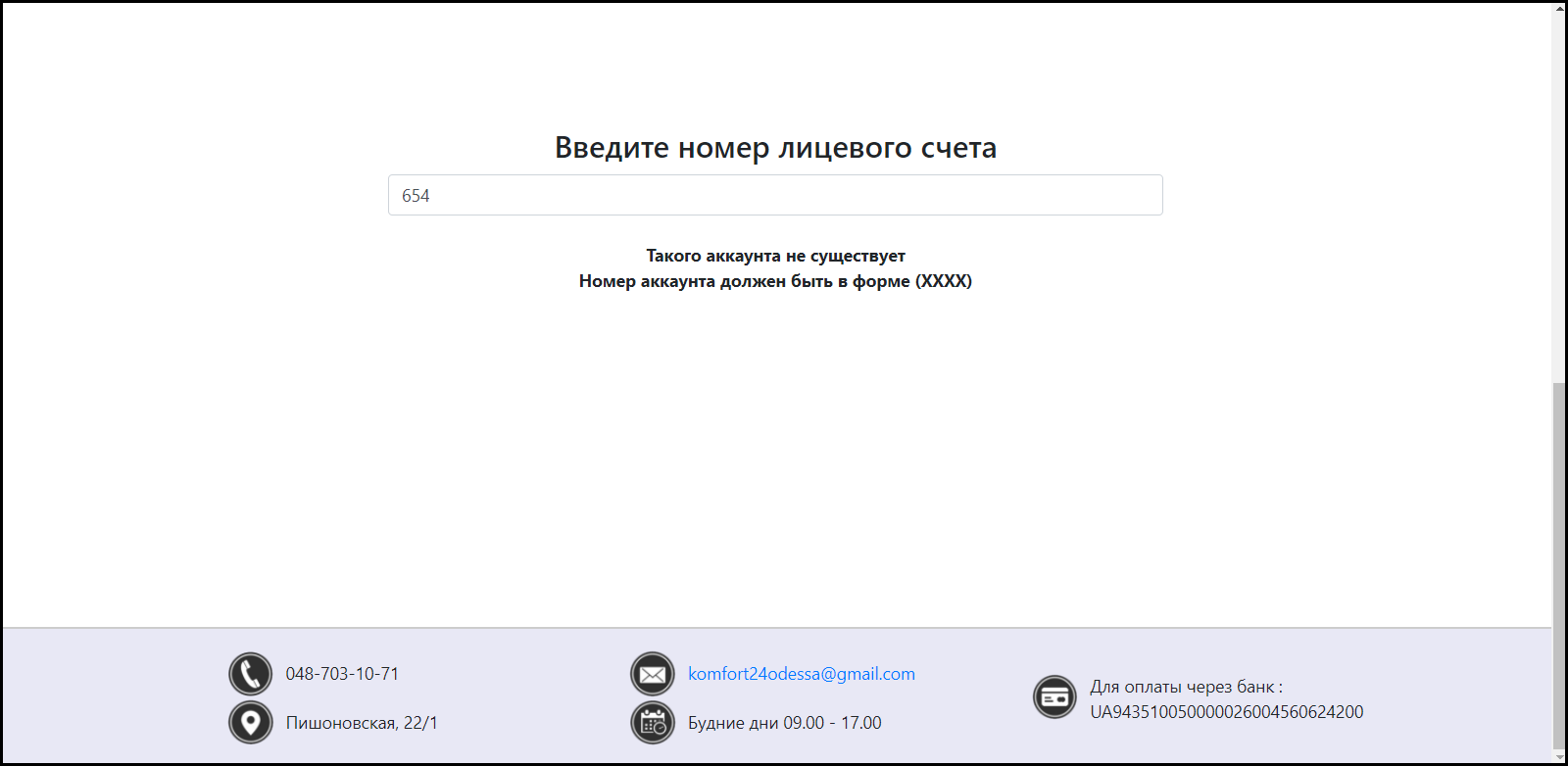


Рисунок 6.19 –Введено некоректний ідентифікатор

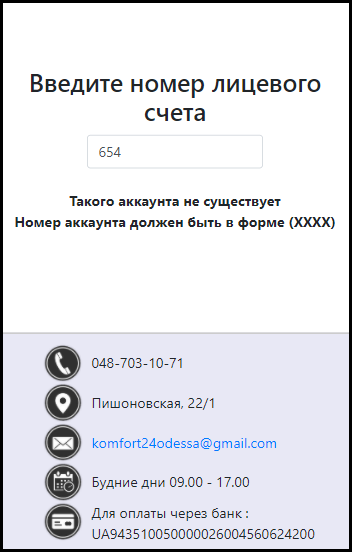


Рисунок 6.20 – Введено некоректний ідентифікатор (на мобільному пристрої)

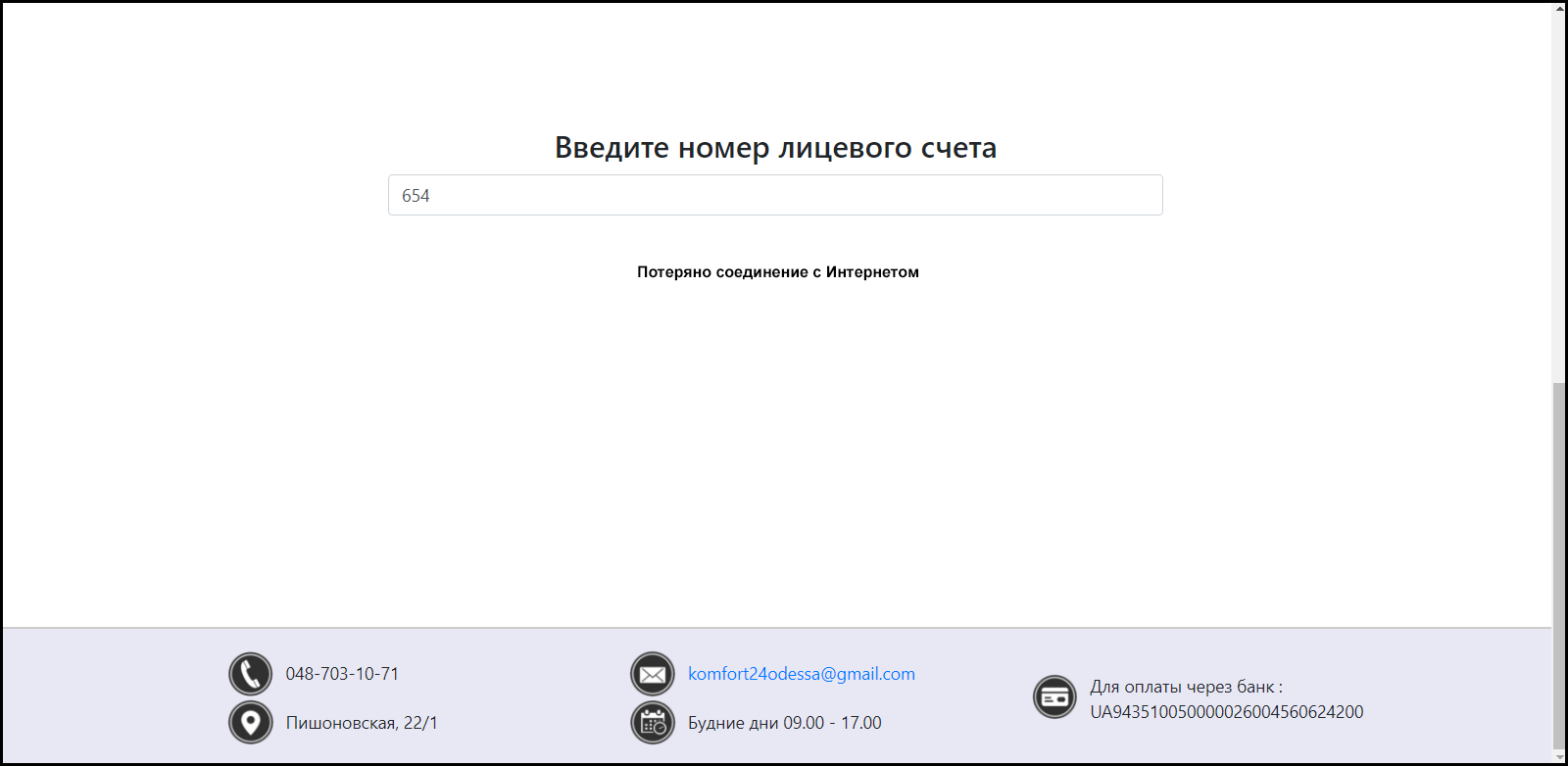


Рисунок 6.21 –Втрачено зв’язок з Інтернетом

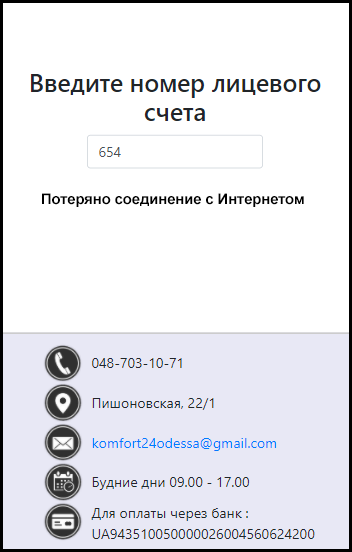


Рисунок 6.22 – Втрачено зв’язок з Інтернетом (на мобільному пристрої)

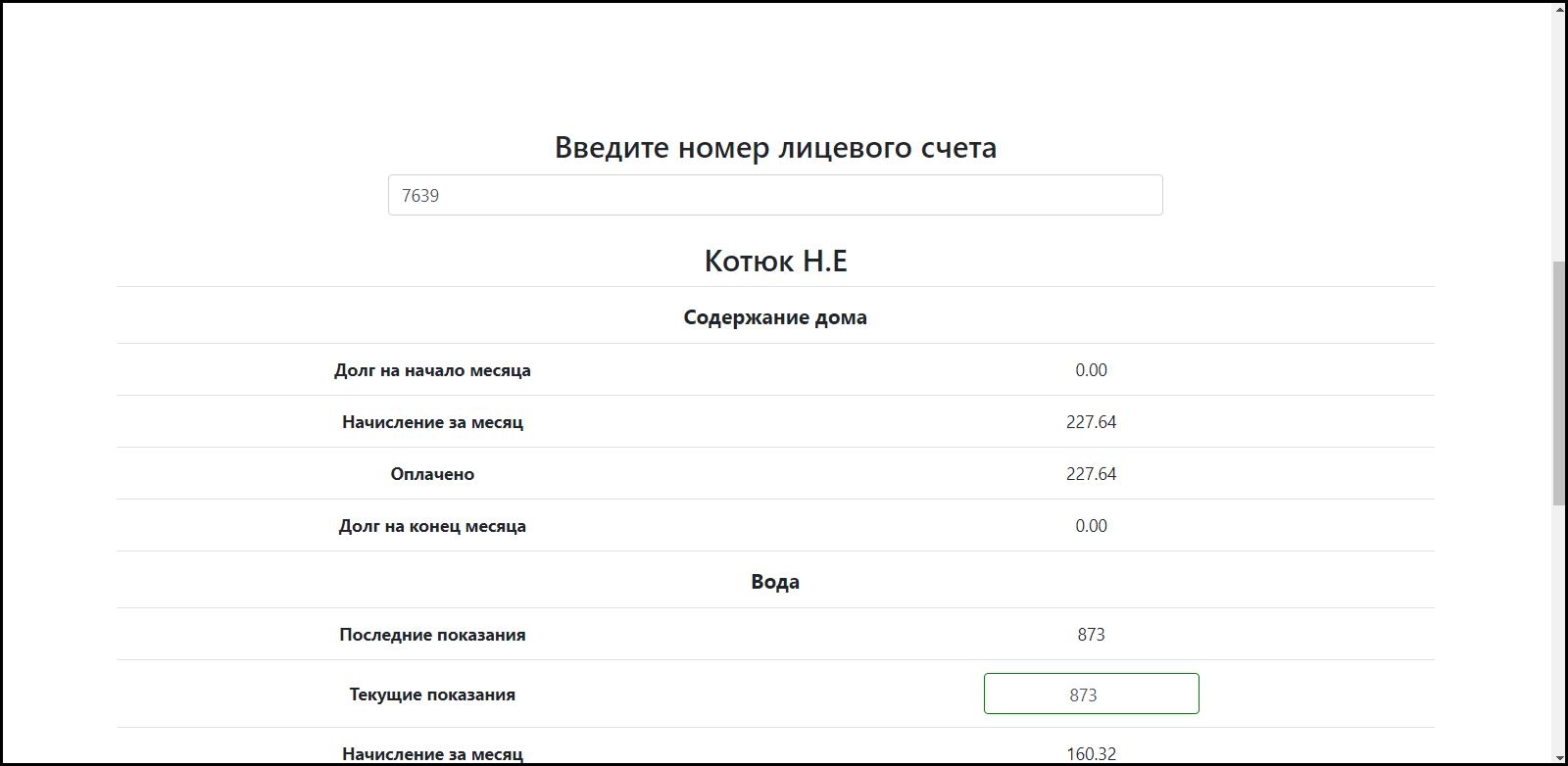


Рисунок 6.23 – Успішний пошук



Рисунок 6.24 – Успішний пошук (на мобільному пристрої)

У табл. 6.10 показано функціональний тест варіанта використання «Огляд службової інформації підприємства».

Таблиця 6.10 – Функціональний тест варіанта використання «Огляд службової інформації підприємства»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| У  м  о  в  а | Втрачено зв’язок з Інтернетом | T | T | F | F |
| Інформація відсутня | T | F | T | F |
| Д  і  я | Отримання оновленої службової інформації |  |  | + | + |
| Вивести стандартну інформацію | + | + | + |  |
| Вивести оновлену інформацію |  |  |  | + |

Виконаємо склейку функціонального тесту варіанта використання «Огляд службової інформації підприємства» (табл. 6.11).

Таблиця 6.11 – Скорочений функціональний тест варіанта використання «Огляд службової інформації підприємства»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| У  м  о  в  а | Втрачено зв’язок з Інтернетом | T | F | F |
| Інформація відсутня | х | T | F |
| Д  і  я | Отримання оновленої службової інформації |  | + | + |
| Вивести стандартну інформацію | + | + |  |
| Вивести оновлену інформацію |  |  | + |

Перевіримо виконання скороченого функціонального тесту у програмній системі (рис. 6.25 – 6.26).

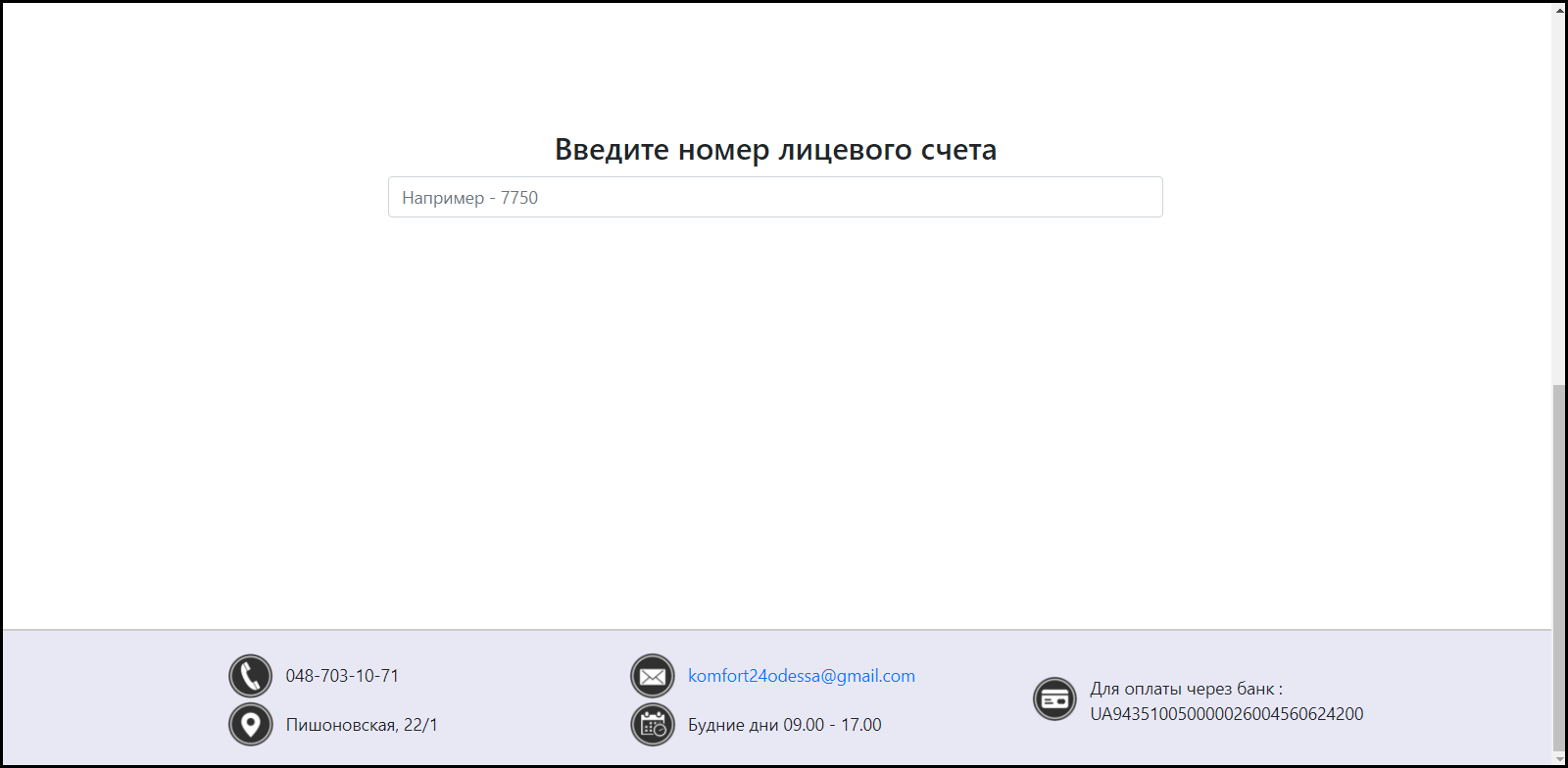


Рисунок 6.25 – Стандартна інформація

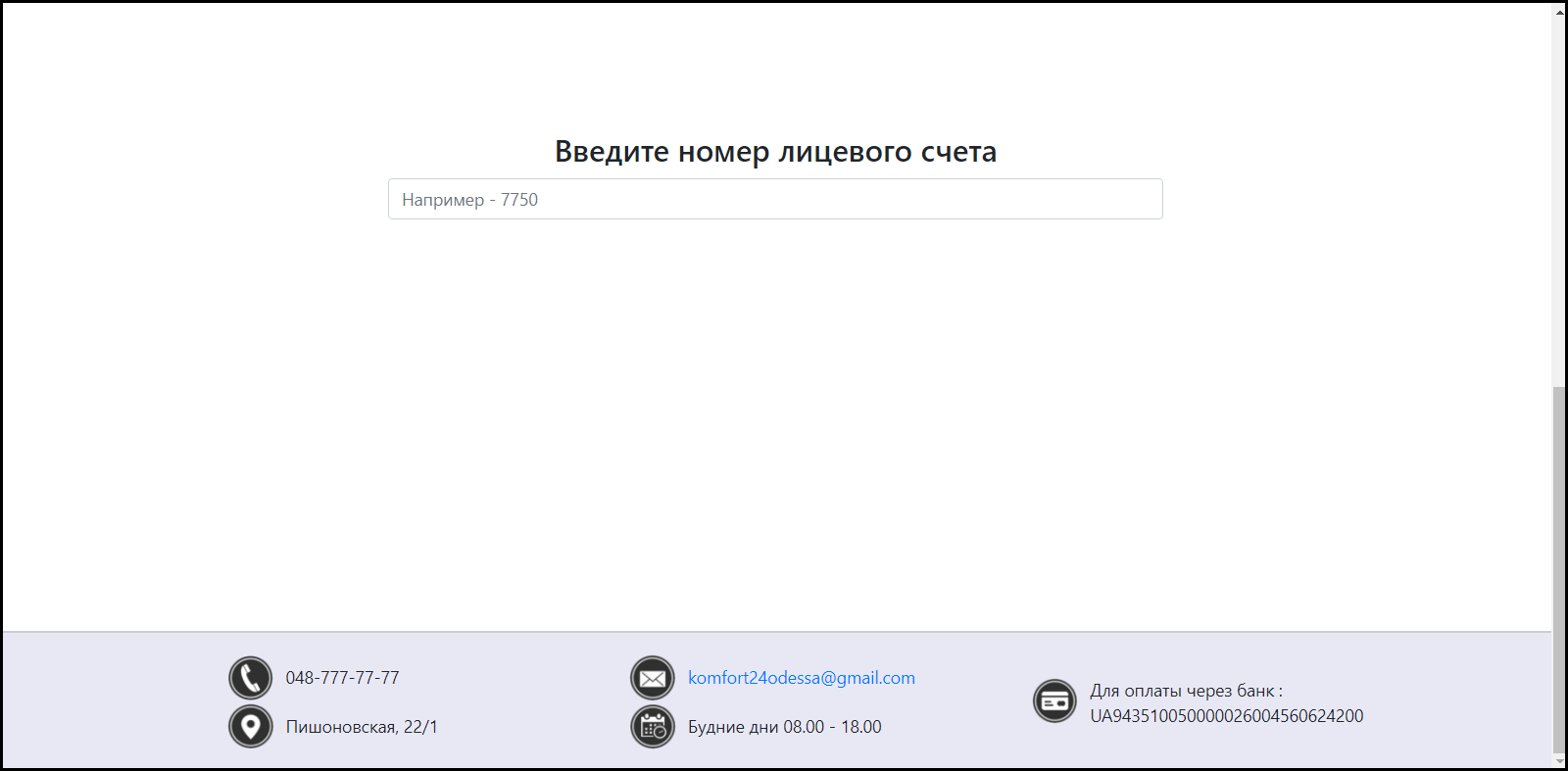


Рисунок 6.26 – Оновлена інформація

У табл. 6.12 показано функціональний тест варіанта використання «Огляд останніх новин підприємства».

Таблиця 6.12 – Функціональний тест варіанта використання «огляд останніх новин підприємства»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| У  м  о  в  а | Втрачено зв’язок з Інтернетом | T | T | F | F |
| Інформація відсутня | T | F | T | F |
| Д  і  я | Отримання інформації щодо новин |  |  | + | + |
| Вивести «На сайті буде публікуватися актуальна інформація» | + | + | + |  |
| Вивести отриману інформацію |  |  |  | + |

Виконаємо склейку функціонального тесту варіанта використання «Огляд останніх новин підприємства» (табл. 6.13).

Таблиця 6.13 – Скорочений функціональний тест варіанта використання «Огляд останніх новин підприємства»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| У  м  о  в  а | Втрачено зв’язок з Інтернетом | T | F | F |
| Інформація відсутня | х | T | F |
| Д  і  я | Отримання інформації щодо новин |  | + | + |
| Вивести «На сайті буде публікуватися актуальна інформація» | + | + |  |
| Вивести отриману інформацію |  |  | + |

Перевіримо виконання скороченого функціонального тесту у програмній системі (рис. 6.27 – 6.29).

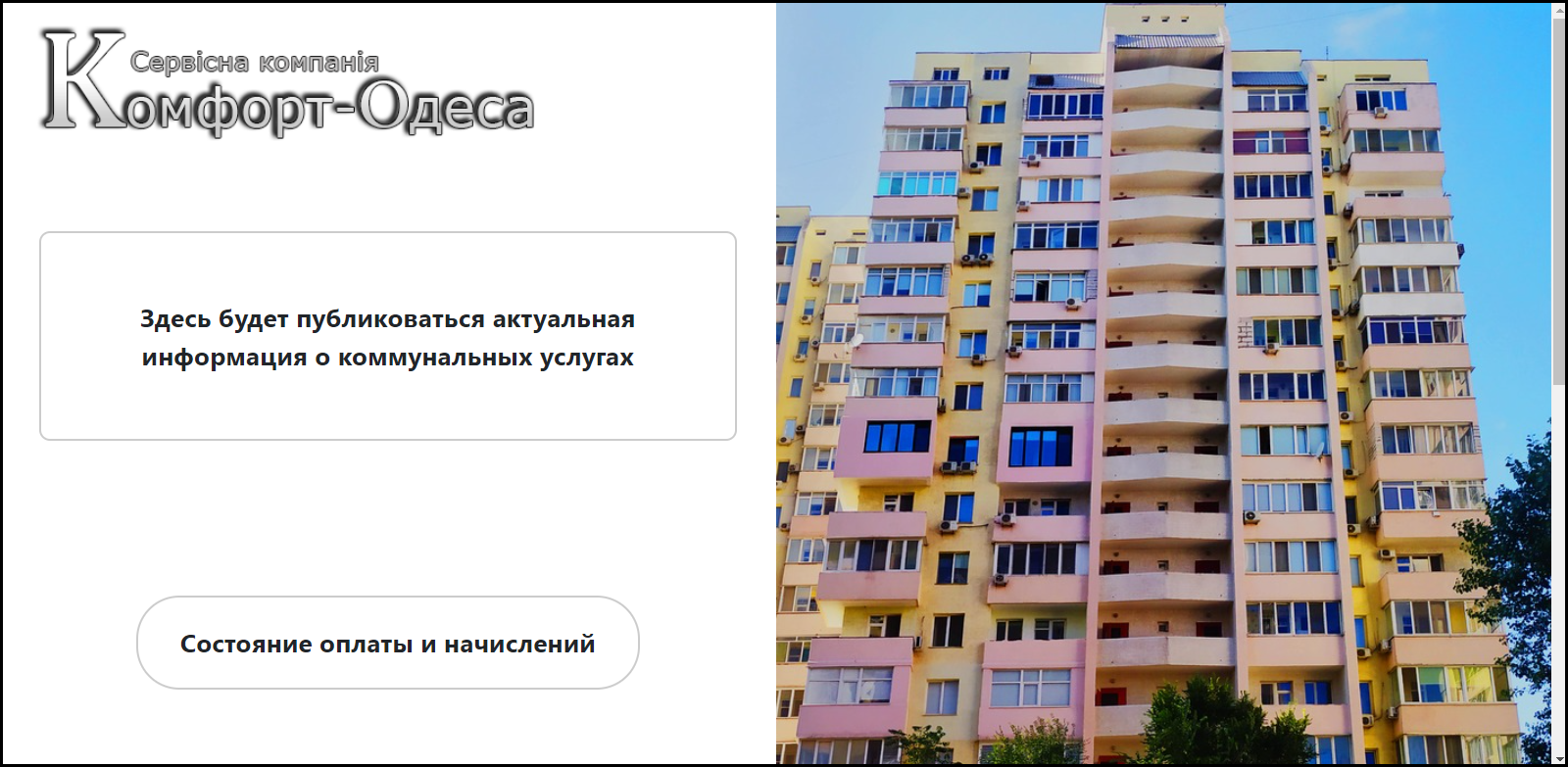


Рисунок 6.27 – Стандартна інформація



Рисунок 6.28 – Стандартна інформація (на мобільному пристрої)

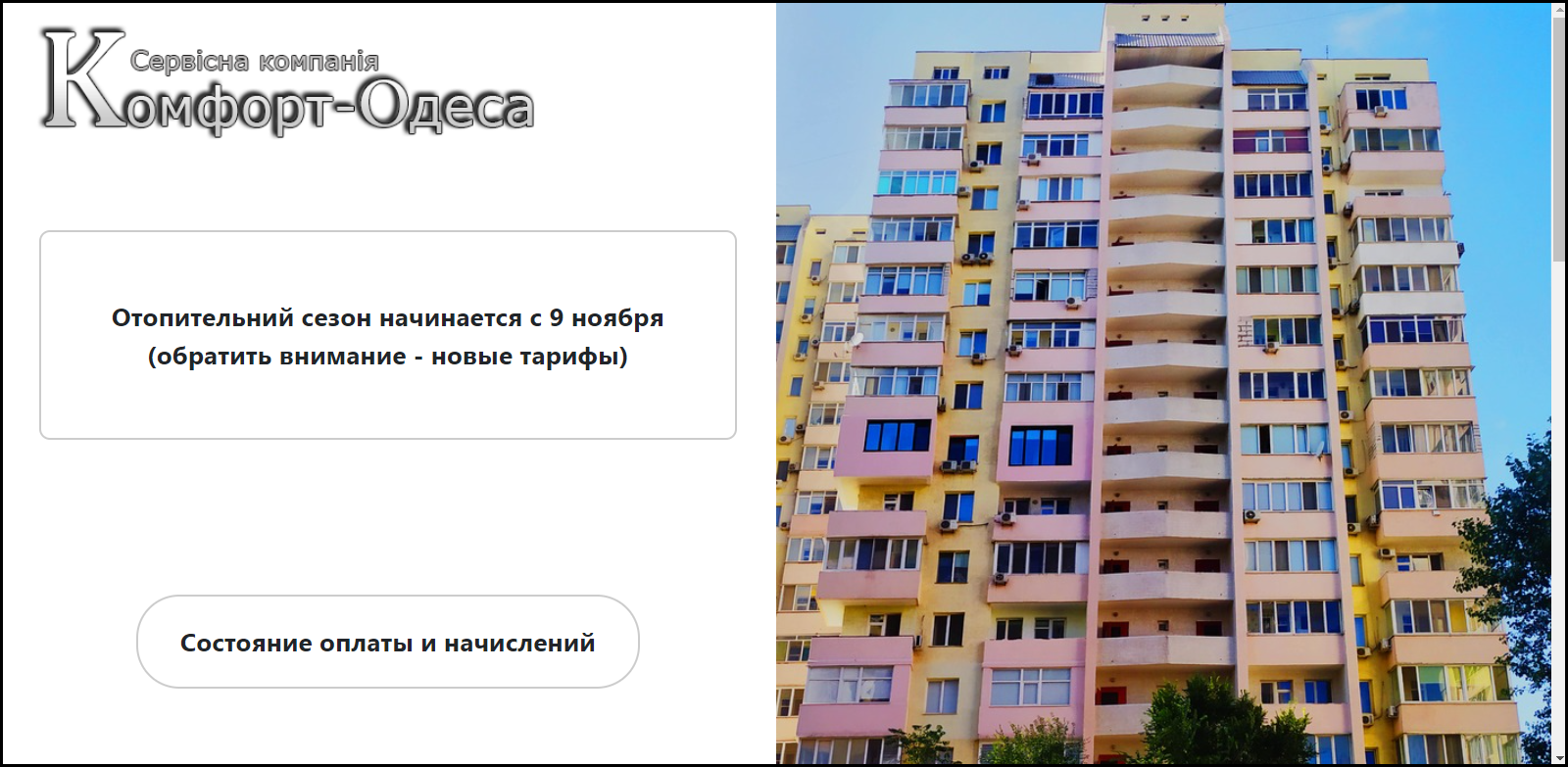


Рисунок 6.28 – Оновлена інформація

## 6.3 Розгортання системи

Для розгортання програмної системи за допомогою Docker необхідно написати набір команд та зберегти його із назвою DockerFile без зазначення формату. Розглянемо набір команд для розгортання вебзастосування CRM системи:

FROMmcr.microsoft.com/dotnet/core/aspnet:3.1-buster-slimASbase

RUN apt-get update -yq \

&& apt-get install curl gnupg -yq \

&& curl -sL https://deb.nodesource.com/setup\_14.x | bash \

&& apt-get installnodejs -yq

WORKDIR /app

EXPOSE 80

EXPOSE 443

FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster AS build

WORKDIR /src

COPY ["Web/Web.csproj", "Web/"]

COPY ["Models/Models.csproj", "Models/"]

COPY ["Tools/Tools.csproj", "Tools/"]

COPY ["SheetsEF/SheetsEF.csproj", "SheetsEF/"]

COPY ["GoogleLib/GoogleLib.csproj", "GoogleLib/"]

RUN dotnet restore "Web/Web.csproj"

COPY . .

WORKDIR "/src/Web"

RUN dotnet build "Web.csproj" -c Release -o /app/build

FROM build AS publish

RUN dotnet publish "Web.csproj" -c Release -o /app/publish

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "Web.dll"]

У даному фрагменті виконуються всі необхідні дії для створення образу, а саме:

1. завантаження легковажних аналогів необхідних для запуску компонентів;
2. копіювання всіх необхідних проектів;
3. збирання проектів.

Наступною дією йде публікація образу на сайті DockerHub.com для того, щоб ми змогли запустити даний образ в контейнері на віддаленій віртуальній машині, як на своєму комп’ютері, використовуючи всі переваги Docker.

# 7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Охорона праці – система правових соціально-економічних, санітарно-гігієнічних, організаційно-технічні та лікувально-профілактичних заходів та засобів спрямованих на збереження життя співробітника у процесі трудової діяльності.

Соціальною сутністю охорони праці є підтримка здоров'я і працездатності економічно активного населення на максимально можливому рівні, а також соціальний захист потерпілих на виробництві та членів їх сімей.

Економічною сутністю охорони праці є мінімізація втрат суспільства при здійсненні ним виробничої діяльності шляхом запобігання випадків виробничого травматизму та професійної захворюваності.

У дипломному проекті були розглянуті питаннястворення CRMсистеми для сервісної компанії, вирішення яких здійснюються ІТ відділом сервісної компанії Komfort-Odessa, а саме їх виконання здійснюєтьсяпрограмістом-розробником.

Отже, розглянемо питання охорони праці та безпеку у надзвичайних ситуаціях програміста-розробника, яке розташоване втриповерховій будівлі, в офісному приміщенні на 2 поверсі, на територіїжитлового комплексу.

Організація заходів із забезпечення безпечних умов праці та поведінки у надзвичайних ситуаціях засновується на наступних нормативних документах:

– ДСТУ 2293-99. Охорона праці. Терміни та визначення [14];

– НАПБ А.01.001-2015. Правила пожежної безпеки в Україні [15];

– ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку;

– НПАОП 0.00-1.28-10. Правила охорони праці при експлуатації ЕОМ;

– ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень;

– НАПБ Б.03.001-2204. Типові норми належності вогнегасників;

– ДБН В.2.5-28-2006. Природне і штучне освітлення.

Вивчення найважливіших заходів з охорони праці та безпеки у надзвичайних ситуаціях для працівників галузі інженерія програмного забезпечення наведено у Додатку В та висвітлює наступні питання:

* організація та управління охороною праці на робочому місті програміста-розробника;
* визначення основних параметрів умов праці робочого місця;
* розрахунок заземлення;
* безпека під час надзвичайних ситуації техногенного характеру;
* заходи з охорони праці та безпеки у надзвичайних ситуаціях;
* безпека у суспільстві в умовах загрози COVID-19.

В ході роботи проаналізовано робоче місце інженера-програміста, вимоги до інтервалів робочих змін, необхідний рівень освітленості при умовах роботи на моніторі високої точності, та режим відпочинку.

Спираючись на проаналізовані дані, було розраховано заземлення та визначення категорії пожежонебезпечних офіснихустанов.

Було запропоновано заходи, які спрямовані на покрашення умов праці,зменшення кількості електротравм робітників сервісної компанії, а також заходи з пожежної безпеки офісної будівлі.

# ВИСНОВКИ

Основним завданням даної роботи було розробка CRMсистеми для компанії у сфері житлово-комунальних послуг (ЖКГ).В ході виконання кваліфікаційної роботи було зменшено витрати на зберігання даних шляхом розробки системи з безкоштовним варіантом зберігання даних за допомогою електронних таблиць компаніїGoogle (GoogleSheets).

Програмна система дозволила виконати генерацію квитанцій приблизно у 87 раз швидше, а оновлення рахунків – приблизно у 53 рази. Також було розширено функціональність шляхом додавання можливості пошуку платежів та друкування історії платежів. З CRMсистемою тепер мешканці можуть не виходячи з дому дізнатися стан оплати комунальних послуг, що є дуже актуальним у період пандемії COVID-19.

В процесі розробки системи було виконано аналіз існуючих рішень, розглянуто теоретичну частину предметної області, проаналізовано функціональні та нефункціональні вимоги,спроектовано програмну систему та виконано її реалізацію, тестування та розгортання.

У результатіаналізупроблем користувачів та аналізуіснуючих рішень, було прийнято рішення розробки системи, так як жодна з існуючих систем не вирішувала всі поставлені задачі.

Під час теоретичної частини було розглянуто структуру даних, з якими працює компанія, а також поточний процес обробки платежів.

Завимогами компанії до програмної системи та аналізу існуючих аналогів системи було складено діаграму варіантів використання системи та описано прецеденти.

При проектуванні системи було обрано загальну архітектуру «клієнт-сервер». Також було створено діаграми класів для всі проектів системи для наочної візуалізації відношень між класами.

Усі поставлені вимоги вдалося реалізувати завдяки ефективному використанню та надійній інтеграції технологій і програмних засобів розробки. Кодуваннянастільного застосування і вебзастосування було реалізовано на сучаснихмовах програмування C# та Typescript.

Система була протестована за допомогою складання функціональних тестівта їх ручної перевірки.Після цього систему було розгорнуто на віддаленій віртуальній машині за допомогою Docker.

CRM система має перспективи розвитку, такі як: покращення дизайну інтерфейсів користувачів, розробку мобільних застосувань для популярних операційних системAndroidта iOS.

# СПИСОК ВИКОРИСТАННОЇ ЛІТЕРАТУРИ

1.Управління відносинами з клієнтами [Електронний ресурс] – Режимдоступу: URL: https://uk.wikipedia.org/wiki.

2.KomlevaN.O., VoroniukD. S. Google Sheets as an alternative way to organize storage of relational databases // XIII Міжнародна науково-практична конференція «Інформаційні технології і автоматизація – 2020», Одеса, 22-23 жовтня 2020. С. 195-197.

3.Олександр Остервальдер. Value Proposition Design: How to Create Products and Services Customers Want / ОлександрОстервальдер. – Москва: Альпина, 2015. – 604 с.

4.Кавіцька В.С. Конспект лекцій з дисципліни «Основи програмної інженерії» для студентів напряму 6.050103 – Програмна інженерія / Кавіцька В.С., Любченко В.В. – Одеса: ОНПУ,2016. – 87 с.

5. Фаулер М. UML. Основы. Третье издание. Краткое руководство по стандартному языку объектного моделирования / Мартин Фаулер. – К.: Символ-Плюс,2018. – 192 с.

6.Оніщенко Т.В. Конспект лекцій з дисципліни «Людино-машинна взаємодія» для студентів напряму підготовки 6.050403 «Програмна інженерія» / Оніщенко Т.В., Городнича Е.А. Одеса: ОНПУ, 2017. - 95 с.

7.Глобальна статистика 2020 [Електронний ресурс] – Режим доступу: https://www.web-canape.ru/business/internet-2020-globalnaya-statistika-i-trendy.

8.GoogleDeveloperConsole [Електронний ресурс] – Режимдоступу: https://console.developers.google.com.

9.GoogleDeveloperAPIGuides [Електронний ресурс] – Режимдоступу: https://developers.google.com/sheets/api/guides/concepts.

10.METANIT.COM Сайт о программировании [Електронний ресурс] – Режим доступу: URL: https://metanit.com/.

11.Microsoft – офіційна домашня сторінка [Електронний ресурс] – Режим доступу: URL: https://www.microsoft.com/uk-ua/.

12.Docker [Електронний ресурс] – Режим доступу: URL: https://www.docker.com/.

13.ГленфордДж. Майерс. The Art of Software Testing 3rd Edition / Гленфорд Дж. Майерс. – Wiley&Sons, Inc., Hoboken, NewJersey. – 151 с.

14. ДНАОП Законодавча база [Електронний ресурс] – Режим доступу: URL:https://dnaop.com/.

15. Про затвердження Правил пожежної безпеки в Україні : наказ Міністерства внутрішніх справ України від 30.12.2014 р. № 1417. Офіційний вісник України. 2015 р. № 26. С. 91, Ст. 767.