МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп’ютерної інженерії

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Жуков І.А.

« ____ » _____ 2021 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ “БАКАЛАВР”

Тема: Засоби оповіщення ініціаторів угод з постачальниками в інформаційній мережі авікомпанії

Виконавець: Гуменюк О.А.

Керівник: Сураєв В.Ф.

Нормоконтролер з ЄСКД: Журавель С.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерної інженерії

Освітній ступінь бакалавр

Напрям 121 "Комп'ютерні системи та мережі "

(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____Жуков І.А.

« ____ » _____ 2021 р.

З А В Д А Н Н Я
на виконання дипломного проекту

Гуменюк Олег Анатолійович
(прізвище, ім'я, по батькові)

1.Тема проекту (роботи): " Засоби оповіщення ініціаторів угод з постачальниками в інформаційній мережі авікомпанії "

затверджена наказом ректора від " 26 " квітня 2021 року № 648/ст.

2.Термін виконання проекту (роботи): з 24.05.2021 до 20.06.2021

3. Вхідні дані до роботи (проекту*): вимоги до системи*

4. Зміст пояснювальної записки:
Аналіз предметної області
Аналіз вимог до програмного забезпечення
Структура серверу

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:
ілюстративний матеріал в Power Point.

6. Календарний план

№ п/п	Етапи виконання дипломного проєкту	Термін виконання етапів	Примітка
1	Ознайомитись із завданням на виконання дипломного проекту.	24.05.21 – 25.05.21	
2	Провести аналіз принципів оповіщення ініціаторів угод з постачальниками.	24.05.21 – 25.05.21	
3	Визначити етапи для виконання поставленої мети.	24.05.21 – 25.05.21	
4	Вибрати елементу базу для розробки системи оповіщення ініціаторів угод.	24.05.21 – 25.05.21	
5	Ознайомитися з засобами програмування системи.	24.05.21 – 25.05.21	
6	Описати етапи проектування системи.	24.05.21 – 25.05.21	
7	Описати алгоритм тестування системи.	24.05.21 – 25.05.21	
8	Проаналізувати можливості системи на факт навантаження	24.05.21 – 25.05.21	
9	Оформити пояснювальну записку та графічний матеріал	24.05.21 – 25.05.21	
10	Захистити дипломний проект		

7. Дата отримання завдання « » 2021 р.

Керівник дипломного проєкту Сураєв В.Ф.
(підпис)

Завдання прийняв до виконання Гуменюк О.А.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Засоби оповіщення ініціаторів угод з постачальниками в інформаційній мережі авікомпанії»: 41 сторінок, 10 рисунків, 9 використаних джерел.

УГОДА, ПОСТАЧАЛЬНИК, СЕРВЕР, ОПОВІЩЕННЯ.

Об'єкт дослідження – засіб оповіщення ініціаторів угод з постачальниками в інформаційній мережі авікомпанії.

Предмет дослідження – оповіщення ініціаторів угод з постачальниками в інформаційній мережі авікомпанії.

Мета дипломної роботи – створення засобу оповіщення ініціаторів угод з постачальниками в інформаційній мережі авікомпанії.

Методи дослідження – методи роботи з електронними документами, методи пошуку, обробки та зберігання інформації, методи автоматизації вирішення різних функцій.

Технічні та програмні засоби – SQL Server.

Результати роботи рекомендується використовувати в інформаційних мережах авіакомпаній.

ABSTRACT

Explanatory note to the thesis "Means of notifying the initiators of agreements with suppliers in the information network of the airline": 41 pages, 10 figures, 9 sources used.

AGREEMENT, SUPPLIER, SERVER, ALERTS.

The object of study – a means of notifying the initiators of agreements with suppliers in the airline's information network..

The subject of study – оповіщення ініціаторів угод з постачальниками в інформаційній мережі авікомпанії.

The aim of the thesis – creation of a means of notifying the initiators of agreements with suppliers in the airline's information network..

Method development – methods of working with electronic documents, methods of searching, processing and storing information, methods of automating the solution of various functions.

Technical and software – SQL Server.

Results of the works are recommended for use in airline information networks.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ІС	Інформаційна систма
ІГ	Ініціатор угоди
БІ	Безпека інформації
АІ	Авіаційна індустрія
СУБД	Система управління базами даних
API	Application programing interface
SQL	Structured Query Language

ВСТУП

Автоматизація відділів технічної підтримки в інформаційній мережі авіакомпанії – задача на сьогоднішній день відома і поширена. Можна говорити про величезну кількість вдало впроваджених проектів в різних галузях, в різних за масштабом компаніях, із застосуванням різних засобів автоматизації. Коли ми говоримо про автоматизацію, ми завжди маємо на увазі впровадження автоматизованого засобу з групи програмного забезпечення Help Desk. Фактично поняття HelpDesk стало аналогом поняття автоматизації процесів, що говорить про те, що в даний час більшість авіакомпаній в тій чи іншій мірі переглянули свій підхід до структури і процесів управління.

Служба підтримки відіграє важливу роль у допомозі клієнтам. Сучасна, всеосяжна консультативна служба є фронт-офісом для всієї авіаіндрустрії і може обробляти більшість потреб і запитів користувачів без допомоги професіоналів. Підтримка замовників є єдиною точкою контакту з авіакомпанією, яка забезпечує своєчасне вирішення їх проблем. Іншими словами, якщо сервісна служба допомоги користувачам не витрачати час на нескінченний пошук фахівців, які можуть вирішити свої проблеми.

Часто служба допомоги не тільки обробляє зовнішні запити від користувачів, але ці скарги були ініційовані в рамках авіакомпанії, наприклад, інциденти, вирішені автоматично або вручну персоналом, виявили або отримали сервісні запити від інших відділів авіакомпанії. Наукова проблема полягає у тому, що служби Help Desk, які є аналогами програмного, являються лише способом передачі загальної інформації від авіакомпанії до клієнтів.

Жодний програмний продукт-аналог не має функціоналу передачі системної інформації, яка надала б змогу ідентифікувати проблему без необхідності присутності спеціаліста біля комп’ютера користувача. Наявність подібного функціоналу надала б змогу у середньому пришвидшити швидкість обробки заявок та зекономити час як спеціаліста, так і користувача.

Об’єктом дослідження виступає процес підтримки користувачів.

Предмет дослідження представляє собою процес оповіщення ініціаторів угод з урахуванням отримання системних та користувацьких даних.

Актуальність даної роботи полягає в тому, що рішення проблеми нераціональної витрати робочого часу спеціалістів і задовільної якості обслуговування дозволяє знизити навантаження на робочий персонал відділу технічної підтримки авіакомпанії, скоротити час очікування, підвищити довіру співробітників до відділу.

Мета дипломної роботи є розробка системи оповіщення ініціаторів угод з постачальниками в інформаційній мережі авіакомпаніх з метою оптимізації діяльності відділу технічної підтримки для підвищення якості його роботи і оптимізації витрат робочого часу співробітників. Для досягнення мети проекту були визначені наступні задачі:

- аналіз даних та бізнес-процесів предметної області;
- розроблення бази даних;
- тестування розробленої системи.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Принципи SMS-оповіщення в інформаційних мережах

SMS-оповіщення в інформаційних мережах - це набір практик, що дозволяє організаціям спілкуватися та взаємодіяти зі своїми замовниками через будь-який мобільний пристрій чи мережу. SMS-оповіщення в інформаційних мережах широко відомий як бездротовий маркетинг. Однак бездротовий зв'язок не обов'язково мобільний. Наприклад, спілкування споживача з веб-сайтом із настільного комп’ютера вдома, із передачею сигналів через бездротову локальну мережу (WLAN) або через супутникову мережу, може кваліфікуватися як бездротовий, але не мобільний зв’язок.

Оповіщення на мобільному телефоні стає дедалі популярнішим з часів появи Служби коротких повідомлень (SMS) на початку 2000-х років у Європі та деяких районах Азії, коли підприємства почали збирати номери мобільних телефонів та надсилати бажаний (або небажаний) вміст. Минуле десятиліття стало свідком революції у використанні ІКТ у країнах, що розвиваються. Багато людей та офіси, а також сільські фермери мають ІКТ-засоби, такі як персональні комп'ютери та мобільні телефони.

Найбільший приріст використання ІКТ спостерігається у мобільній телефонії, де кількість абонементів у країнах, що розвиваються, зросла з приблизно 30 відсотків усього світу у 2000 р. До понад 50 відсотків у 2004 р. Та майже до 70 відсотків у 2007 р. . Хоча використання Інтернету зросло не так швидко, як мобільний зв’язок, за той самий період воно зросло в десятки разів у країнах, що розвиваються. Інші засоби ІКТ, такі як телетрансляція, радіо FM та інформаційні центри, також помітно збільшились за той самий період.

Текстові повідомлення - це широко використовуваний метод сповіщення різними програмами та службами. Найпоширеніша мета SMS-сповіщення в сучасному контексті програми для смартфона включає перевірку номера, повідомлення про певні події, масову рекламу, невеликий спосіб оплати тощо. Текстове сповіщення є не дуже економічним у порівнянні зі сповіщеннями електронною поштою, хоча дуже ефективний спосіб повідомити користувача про делікатну в часі інформацію. Текстове сповіщення популярне серед логістичного бізнесу, 10 постачальників освітніх послуг, охорони здоров'я, банківської справи тощо. Це надійний засіб масової обізнаності у багатьох країнах, що розвиваються, а також засіб виправлення різноманітних послуг у розвинених країнах. Наприклад, під час спалаху Еболи в 2014 році Сенегал надіслав чотири мільйони SMS-повідомлень широкій громадськості, намагаючись підвищити обізнаність громадськості про цю хворобу. Багато екстрених служб створені таким чином, щоб людям, які перебувають неподалік, де сталася надзвичайна ситуація, надсилали SMS-повідомлення про інцидент та інструкції, яких слід виконувати. На великих підприємствах, таких як DHL, SMS - це зручний спосіб для користувачів відстежувати хід виконання замовлення або отримувати інформацію про те, звідки брати замовлення. SMS зручний, коли мобільна мережа перевантажена через наявність в осередку мережі більше, ніж очікувана кількість пристроїв. Шлюз SMS є кращим у ситуаціях, коли текстові повідомлення автоматично генеруються на основі події. Шлюз SMS - це веб-служба, як правило, веб-сайт, що дозволяє користувачам надсилати SMS за допомогою комп’ютера. Наразі Amazon-повідомлення SMS підтримуються для телефонних номерів у Сполучених Штатах, таким чином, замість SMS-повідомлення, push-повідомлення надсилається мобільним користувачам. Подібним чином надсилання повідомлень електронною поштою за допомогою SMTP, шлюз SMS може бути інтегрований в існуючий сервер CASS для реалізації надсилання SMS.

Шлюз SMS працює як інтерфейс між різними протоколами для надсилання SMS, створеного з веб-сайту або настільної програми. Він перетворює та

надсилає повідомлення в Центр SMS, що є місцем, встановленим провайдерами мережі для збору SMS та відправлення його на вказаний номер стільникового телефону.

1.2. Email оповіщення ініціаторів угод в інформаційних мережах.

Електронна пошта є важливим інструментом співпраці для успішного функціонування практично будь-якої організації в індустріальному світі. Електронна пошта як інструмент спілкування займає більше часу, ніж обмін миттєвими повідомленнями, соціальні медіа та телефон разом узяті. Що робить використання електронної пошти настільки поширеним, це простота використання для надсилання та отримання файлів та повідомлень. Отже, це основна інформаційна транспортна система для багатьох підприємств. Застосовується для надсилання автоматизованих повідомлень, таких як звіт про транзакції після того, як користувач придбав товар через Інтернет-магазин, для масового маркетингу інтернет-магазинами у формі інформаційних бюлетенів, для відправки щомісячних рахунків постачальниками послуг своїм клієнтам тощо. Проста пошта Сервер протоколу передачі (SMTP) необхідний для надсилання електронного листа, тоді як для отримання електронного листа використовується протокол Post Office 3 (POP3) або протокол доступу до Інтернет-повідомлень (IMAP). Програма під назвою поштовий клієнт, напр. Microsoft Outlook використовується для надсилання та отримання електронного листа. Поштовий клієнт необхідний для отримання електронних листів, але CASS надсилає лише електронні листи. Отже, поштовий клієнт для CASS не потрібен.

У контексті CASS, коли використовувався Amazon SNS, два типи електронних листів надсилалися суб’єктам, які брали участь у дослідженнях. Як тільки адміністратор дослідження ввів електронну адресу суб'єкта, йому було надіслано електронне повідомлення з повідомленням про свою участь у дослідженні. Цей електронний лист містив посилання від SNS, яке інформує користувача про подальшу підписку на електронні листи з повідомленнями. Усі

інші електронні листи надсилаються точно в ту ж дату та час, коли на новий запит готові відповісти. Якщо користувач не зміг підписатись, то в майбутньому темі не надсилатимуться електронні листи.

Цей крок - надсилання та отримання електронної пошти. Клієнт електронної пошти клієнта підключається для надсилання нової електронної пошти. SMTP SERVER сприяє отриманню електронної пошти користувачами на машині через поштові клієнти POP3 IMAP, необхідну для забезпечення надсилання електронних листів дійсній темі. Отже, було б неможливо надіслати сповіщення електронною поштою, якщо користувач провалив цей перший крок. Це допомагає забезпечити отримання потрібної особи електронною поштою.

Використання надійного постачальника, такого як Amazon, необхідне для успіху роботи системи електронної пошти, що використовується будь-якою організацією. Успіх системи електронної пошти вимірюється чотирма показниками, а саме коефіцієнтом відмов, частотою скарг, проблемами з вмістом і швидкістю доставки. Відмова - це стан, коли електронного листа неможливо надіслати на призначену адресу призначення, оскільки адреса поштової скриньки не існує або повна пошта. Скарга виникає, коли одержувач позначає електронний лист як спам або небажаний. Проблема із вмістом виникає, коли вміст електронного листа є, наприклад, зловмисним, оманливим, шахрайським тощо. Швидкість доставки - це кількість успішно надісланих електронних листів одержувачу без таких проблем, як відмова, скарги та оманливий вміст.

1.3. Мобільні системи оповіщень

Програми для смартфонів стали важливою частиною повсякденної діяльності в промислово розвинутих країнах. Вони використовуються для легкого доступу до таких послуг, як охорона здоров’я, освіта, транспорт, навігація, покупки тощо. Подібним чином вони також використовуються для подій, що враховують час, наприклад, для повідомлення водія автомобіля про ДТП, яка сталася на його / її маршруті, та пропонуючи інформацію про альтернативний

маршрут або затримку через аварію. Додатки настільки розумні, що спілкуються із сервером майже в реальному часі. Як вони це роблять? У абстракції вищого рівня це робиться шляхом підтримання постійного зв’язку між програмою та сервером. Мобільні системи push-сповіщень розроблені, щоб заповнити цю прогалину у підтримці цього спілкування розробниками додатків, щоб програма отримувала повідомлення, як тільки нова інформація стане доступною на сервері. Кожна платформа має власну архітектуру для впровадження системи сповіщення, хоча основна мета однакова.

Google Cloud Messaging (GCM) для Android - це служба сповіщень, яка полегшує програму для надсилання нижчих повідомлень, що походять від сервера і призначені для програми, а також попередніх повідомлень, які походять від програми GCM призначені для сервера. GCM обробляє всі аспекти черги повідомлень і доставки до цільового додатка Android. Існує два типи повідомлень, які можна відправити або отримати за допомогою GCM, полегшене повідомлення, що інформує програму про нові дані, доступні на сервері, наприклад завантажено нове відео. Інший тип повідомлень може містити до 4 кб даних про корисне навантаження, що дозволяє таким програмам, як обмін миттєвими повідомленнями, надсилати та отримувати корисне навантаження.

Реалізація GCM складається із сервера підключення, наданого Google, стороннього сервера додатків та самого додатка. На пристрої користувача є одне активне з'єднання з хмарним сервером Google, яке не вимагає енергії, оскільки через нього не надсилається трафік, поки немає даних. Усі програми з підтримкою GCM використовують це з'єднання для отримання даних від свого сервера. Це розумний спосіб обміну даними між програмами та їх сервером, оскільки він не вимагає традиційної дії опитування для регулярної перевірки нових даних. Оскільки всі програми використовують одне і те ж підключення, це економить і енергію, і дані в смартфоні. Щоразу, коли будуть доступні будь-які нові дані для будь-якої програми, що підтримує GCM, це з'єднання буде використано. Сервер додатків надсилає дані в GCM, а GCM надсилає дані на пристрій. Якщо для кожного додатка на пристрої Android буде потрібно окреме

підключення, то це розрядить акумулятор, а також буде дорогим у ситуації, коли мобільні дані обмежені. Отже, GCM виступає мостом між сервером додатків та самим додатком на пристрої. Користувачі мають вибір між тим, чи отримувати сповіщення, чи блокувати сповіщення для певної програми. Розробники повинні стежити за вподобаннями користувачів. Якщо пристрій недоступний, оскільки він не підключений до Інтернету, тоді натискання буде збережено на сервері GCM для доставки, коли пристрій буде в мережі. Такі push-повідомлення, що зберігаються в GCM, закінчуються через певний час.

Apple оповіщення - це надійна та високоефективна послуга для надсилання інформації на пристрої iOS та OS X. Це постійне з'єднання дозволяє пристроям встановлювати зашифроване IP-з'єднання, яке використовується для отримання сповіщення. Коли сповіщення про програму надходить, коли програма не запущена, операційна система iOS, встановлена на пристрої, попередить користувача, повідомляючи про нові готові дані. Повідомлення складається з двох основних фрагментів даних: маркера пристрою та корисного навантаження. Токен пристрою однозначно ідентифікує пристрій, який готовий отримати сповіщення, тоді як корисне навантаження є фактичним повідомленням, призначеним для цього пристрою.

1.4. Веб-системи як сучасний засіб оповіщення в інформаційних мережах

Служба сповіщень Windows Push (WNS) полегшує розробникам надсилати сповіщення, невелику функцію інтерфейсу, яка з’являється на екрані телефону, плиточці, значку та необроблених оновленнях із їх хмарної служби на телефон Windows. Це ефективний механізм доставки нових оновлень для користувачів, не витрачаючи потужність пристрою. Створення функцій push-сповіщень для телефону з ОС Windows включає багато кроків.

Перший крок надсилання push-сповіщення включає надсилання запиту на канал push-сповіщення на платформу сповіщень клієнтів (NCP). NCP просить WNS створити канал сповіщення. Уніфікований ідентифікатор ресурсу (URI) надсилається назад на пристрій-запитувач для надсилання сповіщення. Windows повертає цей канал сповіщень до програми. Тепер програма надсилає цей URI до своєї хмарної служби. Цей URI діє як інтерфейс зворотного виклику для програми та відповідної хмарної служби. Розробник несе відповідальність за безпечну реалізацію механізму зворотного дзвінка. Коли нові дані готові на сервері, сервер додатків повідомляє WNS за допомогою каналу, який він надав заздалегідь у вигляді URI. Для WNS видається HTTP POST, який містить корисне навантаження сповіщення, яке аутентифікується. WNS направляє це повідомлення на відповідний пристрій. Кожен раз, коли програма запускається, потрібно запитувати URI каналу сповіщення для програми, щоб переконатися, що раніше створений канал не закінчився.

Повідомлення зберігається у WNS до закінчення часу, який може бути встановлений хмарною службою програми. З точки зору розробника систему сповіщення про телефон на Windows не найпростіше впоратись із-за складних кроків. Незважаючи на те, що всі сповіщення обробляються через WNS, окремий URI для окремої програми не дуже зручний у порівнянні з Android.

Сторонні постачальники послуг push-сповіщень популярні серед розробників мобільних пристроїв завдяки простоті управління push-сповіщеннями за допомогою своїх консолей. Деякі з відомих постачальників включають Urban Airship, Parse та Pushwoosh, Amazon Simple Notification Service тощо. Розробникам не потрібно витрачати час і енергію на підтримку системи сповіщень про програму, яка, як правило, є бекендом при використанні таких сторонніх служб . Надсилання push-сповіщення на пристрій включає відстеження ідентифікації пристрою та збереження кожного ідентифікатора пристрою та налаштувань у базі даних, щоб надіслати на нього цільовий push. Розробник повинен відслідковувати, якщо пристрій дозволив отримувати сповіщення чи ні, і якщо пристрій перебуває в режимі он-лайн чи офлайн тощо. Аналогічно, якщо

програма видаляється з пристрою, розробник повинен очистити дані, пов’язані з пристроєм, щоб щоб зменшити перевантаження на задній частині бази даних. Push-сповіщення працює подібно до моделі публікації передплати, коли клієнт підписується на інформацію, а інформація публікується серед тих клієнтів, які підписалися. Тому важливо відстежувати інтереси користувача, щоб отримувати сповіщення. Якщо програма надсилає непотрібне push-повідомлення на пристрій, який явно відмовився від підписки на push-сповіщення через консоль свого менеджера програм на пристрої, дуже ймовірно, що користувач видалить програму або використає альтернативну програму. Взаємодія з користувачем є визначальним фактором в системі push-сповіщень, оскільки всі програми, встановлені на користувацькому пристрої, змагаються за те, щоб змусити користувача проводити в них якомога більше часу. Ці фактори сприяють використанню сторонньої системи push-повідомлень.

Клієнтська програма використовує бібліотеку, надану постачальниками послуг сповіщення, для оновлення уподобань користувача щодо отримання сповіщень та обробки корисного навантаження після отримання push-повідомлення. Розробники можуть використовувати це корисне навантаження для створення власних повідомлень за допомогою тостів, значків або повідомлень у рядку стану на основі платформи. Коли новий сервер буде готовий на сервері програми, буде здійснено виклик до служби сповіщень для обробки корисного навантаження та групи пристроїв, які погодились отримати цей певний push на різних платформах. Служба сповіщень працює як середній чоловік, щоб обробити цей поштовх, використовуючи власну службу сповіщення кожної платформи, і надсилає корисне навантаження на кожен пристрій, а потім, нарешті, до цільової програми. Такий інтерфейс між власними системами сповіщень полегшує розробникам управління push-сповіщеннями. Розробник може налаштувати надсилання розширених текстових повідомлень. Подібним чином багато поведінки push-сповіщення можуть бути змінені безпосередньо з цієї консолі, наприклад відкриття браузера для переходу користувача на певний веб-сайт, коли користувач натискає сповіщення. Це полегшує підприємствам

можливість легко складати повідомлення, щоб охопити свою аудиторію.

Усі обговорювані технології push-сповіщень можуть надсилати push-повідомлення до власних додатків для відповідної платформи. У міру розвитку HTML5, JavaScript та CSS веб-додатки, широко відомі як веб-додатки, набирають популярності серед веб-розробників. jQuery Mobile, AngularJS тощо - популярні фреймворки JavaScript для створення веб-додатків, які мають розширені користувальницькі інтерфейси. Оскільки веб-додатки працюють у веб-браузері, їм не вистачає можливості отримувати нативні сповіщення від сервера, оскільки сервер не може зв’язатись із клієнтом, якщо його не ініціює клієнт. Необхідно підтримувати постійний зв’язок між веб-додатками та їх відповідними серверами для отримання будь-якої нової інформації, доступної на сервері, яка обмежена лише сферою застосування веб-додатків. Існують такі технології, як опитування та тривале опитування, коли сервер постійно контактує з метою отримання будь-якої нової інформації протягом певного періоду або встановлює WebSocket, який має повну двонаправлену можливість зв'язку через TCP (протокол управління передачею). Пізніше, зв’язок WebSocket, поводиться як справжня система сповіщення для веб-додатків. Розробляється нова специфікація для відображення повідомлень поза контекстом веб-сторінки. W3C, Консорціум Всесвітньої павутини, розробляє Інтерфейс прикладного програмування (API) для сповіщень. Такий API дозволить розробникам використовувати нативну систему push-сповіщень, яка може попереджати користувачів поза контекстом веб-сторінок. Опитування - це метод псевдо-push-сповіщень, оскільки основна концепція push-сповіщення - це сервер, який може передавати дані клієнту, але не клієнт, який запитує будь-які нові дані на сервері.

WebSocket - це майбутнє push-сповіщень для веб-додатків. IETF (Internet Engineering Task Force) створив першу специфікацію веб-сокета в 2011 році. Відповідно до специфікації, протокол WebSocket забезпечує двосторонній зв'язок між клієнтом, який запускає надійний код в контрольованому середовищі, для використання моделі віддаленого хосту для цього. модель безпеки на основі походження, яка зазвичай використовується веб-браузерами. Довге опитування

HTTP має великі накладні витрати, оскільки сервер змушений використовувати різні базові TCP-з'єднання для кожного клієнта. Клієнт ініціює запит рукостискання на сервер, а сервер повертає у відповідь рукостискання WebSocket. Після цього рукостискання встановлюється стійкий і відкритий двонаправлений зв’язок для обміну даними. Цей зв’язок може бути закритий будь-якою стороною.

Основна відмінність зв'язку WebSocket від традиційної полягає в тому, що після того, як клієнт і сервер погоджуються обмінюватися даними між ними після рукостискання; вони можуть асинхронно надсилати дані один одному. Це не так у випадку з'єднання HTTP, коли клієнт робить запит на кожен новий фрагмент даних, що йому потрібні, збільшуючи розмір трафіку за допомогою метаданих, необхідних для зв'язку клієнт-сервер.

Висновки до першого розділу

На сьогодні безпека інтернет-комунікацій є делікатною темою для приватних осіб та організацій, оскільки наслідки крадіжки даних є дуже дорогими як у фінансовому, так і в соціальному плані. Зростає занепокоєння щодо того, як хакери можуть використовувати системи push-сповіщень, реалізовані за допомогою WebSocket, GCM тощо, для надсилання зловмисного коду та викрадення даних користувачів. Оскільки було проведено мало досліджень щодо вразливості push-служб, навіть авторитетні служби, такі як GCM та Amazon Device Messaging (ADM), можуть бути використані для викрадення конфіденційних повідомлень з цільового пристрою, неправильної інсталяції чи видалення програми, блокування законного користувача для запобігання користувачеві не використовувати пристрій, а також витирати весь пристрій. У GCM розробники можуть охоче програмувати програму, яка викрадає інформацію про користувача, контролює поведінку користувачів, встановлює небажані програми на пристрій користувача тощо. Звіт, опублікований провідним технологічним журналом Computer World,

посилаючись на дослідження, проведене "Лабораторією Касперського" заявляє, що кіберзлочинці контролюють зловмисні програми, скорочуючи зловмисні програми, встановлені на телефоні Android за допомогою GCM для крадіжки даних користувачів. Розглядається зловмисна програма Trojan-SMS.AndroidOS.FakeInst.a, яка здатна надсилати текстові повідомлення на номери преміум-класу, видаляти вхідні повідомлення без відома власника телефону, генерувати ярлики до шкідливих сайтів та відображати сповіщення про рекламу інших зловмисних програм програми як корисний додаток або ігри. Такі троянські програми торкнулися мільйони мобільних пристроїв Android. Незважаючи на загрози безпеці, спамери використовують службу push-сповіщень для надсилання реклами, фальшивих посилань тощо.

ЧАСТИНА 2
АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Функціональні вимоги

Бізнес-аналіз - це структуроване вивчення проблеми, що має відношення до бізнесу. Бізнес-аналіз проводиться, щоб краще зрозуміти проблему, а потім оцінити, що потрібно для її усунення.

Бізнес-кейс. Комерційна мета або вигода є причиною виконання проекту і може офіційно фіксуватися в документі, що називається бізнес-кейсом, або комерційною пропозицією. Цей документ зазвичай включає фінансові показники (наприклад, приріст виручки, зниження витрат і т.п.) або інші параметри (наприклад, підвищення рівня обслуговування споживачів, підвищення вмотивованості персоналу і т.д.).

Перш ніж починати проект, обов'язково потрібно знати, який результат (продукт) ви хочете отримати. І часом цей продукт необхідно описати найретельнішим чином. Іншими словами, потрібно знати, які вимоги замовник пред'являє до продукту. Повний набір цих вимог називають каталогом вимог, або специфікацією.

Великі і складні проекти зазвичай налічують тисячі вимог. Бізнес-аналіз як раз і дозволяє виявляти проблеми і визначати, що потрібно для їх подолання. У великих проектах, таких, як розробка програмного забезпечення, збір вимог є одним з найважливіших етапів життєвого циклу проекту, котоориий може зайняти кілька тижнів / місяців.

Для виявлення вимог проводиться серія структурованих інтерв'ю з замовниками, які дозволяють точно визначити їх побажання до готового продукту. Спроба безпосередньо дізнатися у замовника, які результати йому потрібні, може закінчиться крахом: замовник стане висувати все нові і нові вимоги, так що ви просто будете не в силах їх задовольнити. Слід пам’ятати вимога впливає на тривалість і вартість проекту. Відповідно, отримуючи докладний список вимог, вам потрібно знати, чи є вони:

- обов'язковими, тобто без них проект буде вважатися незавершеним, а замовник залишиться незадоволеним. Якщо готовий продукт не відповідає всім обов'язковим вимогам, це - провал;

- бажаними. Ці вимоги не обов'язкові, але важливі для замовника, тому їх намагаються дотримуватися, за винятком тих випадків, коли вони тягнуть за собою неприйнятне збільшення вартості або тривалості проекту;

- необов'язковими - це ті вимоги, без яких замовник цілком може обійтися і які задовольняються лише по можливості.

Атестація повинна продемонструвати, що вимоги дійсно визначають ту систему, яку хоче мати замовник. Перевірка вимог важлива, так як помилка в специфікації вимог можуть призвести до переробки системи і великих витрат, якщо будуть виявлені під час процесу розробки системи або після введення її в експлуатацію. Вартість внесення в систему змін, необхідних для усунення помилок у вимогах, набагато вище, ніж виправлення помилок проектування або кодування. Причина в тому, що зміна вимог зазвичай тягне за собою значні зміни в системі, після внесення яких вона повинна пройти повторне тестування.

Відсутність чіткості викладу. Іноді нелегко викласти будь-яку думку природною мовою чітко і недвозначно, що не зробивши при цьому текст багатослівним і важко читається.

Змішання вимог. У призначених для користувача вимогах відсутня чіткий поділ на функціональні вимоги, на системні цілі і проектну інформацію.

Об'єднання вимог. Кілька різних вимог до системи можуть описуватися як єдине призначене для користувача вимога.

Системні вимоги описують властивості і методи всіх об'єктів системи. Програмування - це розробка і реалізація структур даних і алгоритмів. Для розробки системи програмісту необхідно знати структури даних, необхідні для реалізації системи, і алгоритми (бізнес-правила / процедури / пакети обробки даних), які ними маніпулюють. Системні вимоги - деталізований опис системних функцій і обмежень, яке іноді називають функціональної специфікації. Вона служить основою для укладення контракту між покупцем системи і розробниками ПЗ.

Системні вимоги - це більш деталізований опис користувальницьких вимог.

Вони зазвичай служать основою для укладення контракту на розробку програмної системи і тому повинні представляти максимально повну специфікацію системи в цілому. Системні вимоги також використовуються в якості відправної точки на етапі проектування системи. Специфікація вимог може будуватися на основі різних системних моделей, таких, як об'єктна модель або модель потоків даних.

Функціональні вимоги - це перелік сервісів, які повинна виконувати система, причому має бути вказано, як система реагує на ті чи інші вхідні дані, як вона поводиться в певних ситуаціях і т.д. У деяких випадках вказується, що система не повинна робити.

Стандартні форми для функціональних вимог:

- опис функції або об'єкта;
- опис вхідних даних і їх джерела;
- опис вихідних даних із зазначенням пункту їх призначення;
- вказівка, що необхідно для виконання функції;

Якщо це специфікація функції, необхідний опис попередніх умов (передумов), які повинні виконуватися перед викликом функції, і опис заключного умови (постумови), яке повинно бути виконано після завершення виконання функції.

Функціональні вимоги (functional requirements) визначають функціональність ПЗ, яку розробники повинні побудувати, щоб користувачі змогли виконати свої завдання в рамках бізнес-вимог. Іноді вони називаються

вимогами поведінки (behavioral requirements), вони містять положення з традиційним «повинен» або «повинна»: «Система повинна по електронній пошті відправляти користувачеві підтвердження про замовлення».

Провівши аналіз функціональних вимог до системи оповіщення ініціаторів угод з постачальниками в інформаційній мережі авіакомпанії, складено наступний перелік функціональних вимог.

1. Сервер повинен мати авторизацію для користувачів.
2. Сервер повинен мати реєстрацію для користувачів.
3. Сервер повинен авторизовувати лише працівників з інформаційного відділу авіакомпанії.
4. Сервер повинен реєструвати користувачів через підтвердження по електронній пошті.
5. Повідомлення про реєстрацію повинно бути згідно попередньо визначеному шаблону.
6. Під час авторизації слід використати лише логін та пароль.
7. Під час реєстрації слід використати логін, пароль та електронну пошту.
8. Сервер повинен діставати всі угоди з постачальниками кожну добу.
9. Мікросервіс Scheduler повинен тригерити взяття всіх підписаних та зареєстрованих угод с постачальниками.
10. Тригер повинен працювати кожну першу секунду нового дня.
11. Налаштування тригера повинно бути в окремому файлі.
12. Зона тригеру має бути UTC+3.
13. Всі файли угод мають відсилатися на електронну пошту ініціаторів угод.
14. Для відображення змісту угод повинно бути використано визначений шаблон HTML.
15. Сервер повинен записувати всі відправлені угоди до бази даних PostgreSQL.

2.2. Нефункціональні вимоги

Нефункціональні вимоги - Описують характеристики системи і її оточення, а не поведінка системи. Тут також може бути приведений перелік обмежень, що накладаються на дії і функції, виконувані системою.

Вони включають тимчасові обмеження, обмеження на процес розробки системи, стандарти і т.д.

Нефункціональні вимоги не пов'язані безпосередньо з функціями, виконуваними системою. Вони пов'язані з такими інтеграційними властивостями системи, як надійність, час відповіді або розмір системи. Крім того, нефункціональні вимоги можуть визначати обмеження на систему, наприклад на пропускну здатність пристроїв введення-виведення, або формати даних, які використовуються в системному інтерфейсі.

Нефункціональні вимоги ґрунтуються на бюджетних обмеженнях, враховують організаційні можливості компанії-розробника, можливість взаємодії розробляється з іншими програмними і обчислювальними системами, а також такі зовнішні фактори, як правила техніки безпеки, законодавство про захист інтелектуальної власності і т.п.

Нефункціональні вимоги описують цілі і атрибути якості. Атрибути якості (quality attributes) представляють собою додатковий опис функцій продукту, виражене через опис його характеристик, важливих для користувачів або розробників. До таких характеристик відносяться:

- легкість і простота використання;
- легкість переміщення;
- цілісність;
- ефективність і стійкість до збоїв;
- зовнішні взаємодії між системою і зовнішнім світом;
- обмеження дизайну і реалізації. Обмеження (constraints) стосуються вибору можливості розробки зовнішнього вигляду і структури продукту.

Вимоги предметної області характеризують ту предметну область, де буде експлуатуватися система. Ці вимоги можуть бути функціональними і не функціональними. Ці вимоги відображають умови, в яких буде експлуатуватися програмна система. Вони можуть бути представлені у вигляді нових функціональних вимог або у вигляді обмежень на вже сформульовані функціональні вимоги або у вигляді вказівок, як система повинна виконувати обчислення. Невиконання вимог предметної області може призвести до виходу системи з ладу.

Вимоги до продукту описують експлуатаційні властивості програмного продукту. Це вимоги до продуктивності системи, обсягом необхідної пам'яті, надійності (визначає частоту можливих збоїв в системі), переносимості системи на різні комп'ютерні платформи і зручності експлуатації.

Організаційні вимоги відображають політику і організаційні процедури замовника і розробника ПЗ. Включають стандарти розробки програмного продукту, вимоги до реалізації ПО (тобто до мови програмування і методів проектування), вихідні вимоги, які визначають терміни виготовлення програмного продукту, і необхідну документацію.

Вимоги до інтеграції описують низькорівневий інтерфейс взаємодії нової системи з декількома іншими системами компанії. Мета даного документа обгрунтувати і формалізувати вибір методу інтеграції. Документ містить в собі опис методів і способів інтеграції з зовнішніми системами, сервісами.

Інтеграція додатків - це технологічні процеси, які використовуються для організації обміну даними між різними інформаційними системами за допомогою засобів інтеграції, наданими додатками. До засобів інтеграції, наданими додатками відносяться API функції, пакети обробки та експорту / імпорту даних.

В таблиці 3.1 зібрано список нефункціональних вимог до системи оповіщення ініціаторів угод з постачальниками в інформаційній мережі авіакомпанії.

Таблиця 3.1

Предмет вимоги	Аналіз
Мова програмування	Java
Сервер розгортання	Tomcat
Фреймворк	Spring
Бібліотека мапінгу об’єктів	Hibernate
Архітектура	Мікросервіси
HttpClient	FeignClient
Авторизація	Сервер повинен мати авторизацію для користувачів.
Реєстрація	Сервер повинен мати реєстрацію для користувачів.
Авторизація	Сервер повинен авторизовувати лише працівників з інформаційного відділу авіакомпанії
Реєстрація	Сервер повинен реєструвати користувачів через підтвердження по електронній пошті
Реєстрація	Повідомлення про реєстрацію повинно бути згідно попередньо визначеному шаблону.
Авторизація	Під час авторизації слід використати лише логін та пароль.
Реєстрація	Під час реєстрації слід використати логін, пароль та електронну пошту
Сервер	Сервер повинен діставати всі угоди з постачальниками кожну добу.

Мікросервіс Scheduler	Мікросервіс Scheduler повинен тригерити взяття всіх підписаних та зареєстрованих угод с постачальниками.
Тригер	Тригер повинен працювати кожну першу секунду нового дня.
Тригер	Налаштування тригера повинно бути в окремому файлі.
Тригер	Зона тригеру має бути UTC+3.
Оповіщення угод	Всі файли угод мають відсилатися на електронну пошту ініціаторів угод.
Оповіщення угод	Для відображення змісту угод повинно бути використано визначений шаблон HTML.
База даних	Сервер повинен записувати всі відправлені угоди до бази даних PostgreSQL.

Висновки до розділу

Технічне завдання є необхідною частиною будь-якого проекту, воно буде співвідносити побажання замовника та можливості розробників, це дозволить заощадити час та усунути можливі конфлікти. Перш за все, технічне завдання окреслює коло завдань, які необхідно виконати при розробці веб-системи.

Для того, щоб розробники веб-систем змогли повністю реалізувати ідею замовника, необхідно якомога детальніше структурувати її, пояснити своє бачення того, якою має бути остаточна версія..

ЧАСТИНА 3
СТРУКТУРА СЕРВЕРУ

3.1. Архітектура мікросервісів

Архітектура мікросервісу або просто мікросервісів - це особливий метод розробки програмних систем, який намагається зосередитись на побудові однофункціональних модулів з чітко визначеними інтерфейсами та операціями. Ця тенденція набула популярності в останні роки, оскільки підприємства прагнуть стати більш спритними та рухатись до DevOps та постійного тестування.

Мікросервіси мають багато переваг для команд Agile та DevOps - як зазначає Мартін Фаулер, Netflix, eBay, Amazon, Twitter, PayPal та інші технічні зірки еволюціонували від монолітної до архітектури мікросервісів. На відміну від мікросервісів, монолітний додаток побудований як єдиний, автономний блок. Це вносить зміни в додаток повільно, оскільки це впливає на всю систему. Зміни, внесені до невеликого розділу коду, можуть вимагати створення та розгортання абсолютно нової версії програмного забезпечення. Масштабування конкретних функцій програми також означає, що вам потрібно масштабувати всю програму.

Мікросервіси вирішують ці виклики монолітних систем, будучи максимально модульними. У найпростішій формі вони допомагають створити додаток як набір невеликих сервісів, кожна з яких працює в своєму процесі і може бути розгорнута незалежно. Ці послуги можуть бути написані різними мовами програмування та можуть використовувати різні методи зберігання даних. Хоча це призводить до розробки масштабованих та гнучких систем, вона потребує динамічного перетворення.

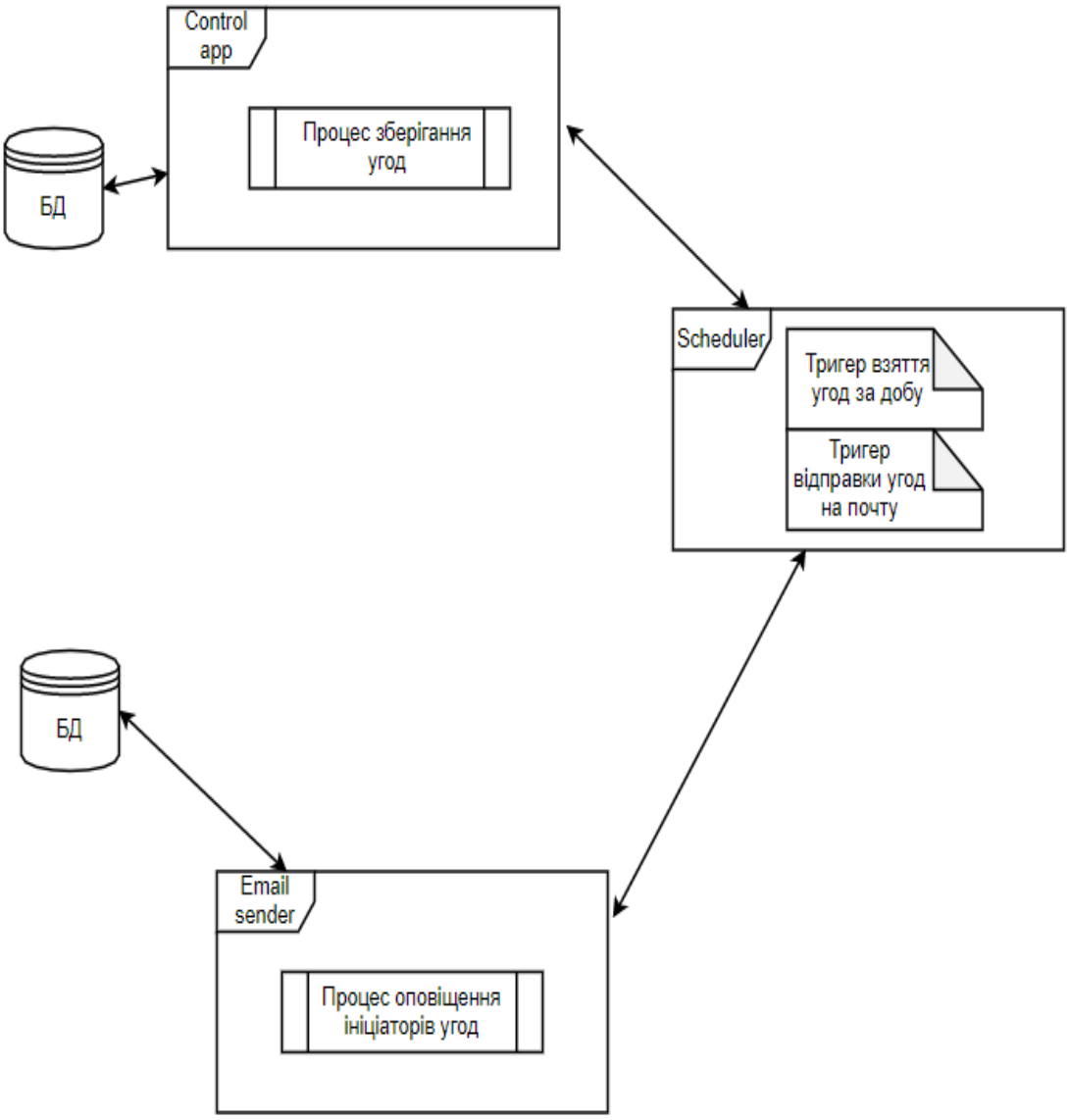
Мікросервіси часто підключаються за допомогою API і можуть використовувати багато однакових інструментів та рішень, які виросли в екосистемі RESTful та веб-сервісу. Тестування цих API може допомогти перевірити потік даних та інформації протягом усього розгортання мікросервісу.

Кілька компонентів. Програмне забезпечення, побудоване як мікросервіс, за визначенням може бути розбито на багатокомпонентні служби. Чому? Таким чином, кожна з цих служб може бути розгорнута, налаштована, а потім передислокована незалежно, не порушуючи цілісність програми. Як результат, вам може знадобитися лише змінити одну або кілька окремих служб, замість того, щоб перерозподіляти цілі програми. Але цей підхід має свої мінуси, включаючи дорогі віддалені дзвінки (замість поточних викликів), більш грубі віддалені API, а також підвищену складність при перерозподілі відповідальності між компонентами.

Створений для бізнесу. Стиль мікропослуг зазвичай організований навколо ділових можливостей та пріоритетів. На відміну від традиційного монолітного підходу до розробки - де різні команди мають особливу увагу, скажімо, на інтерфейси користувача, бази даних, технологічні рівні або логіку на стороні сервера - архітектура мікросервісу використовує міжфункціональні групи. Обов'язки кожної команди полягають у створенні конкретних продуктів на основі однієї або декількох окремих служб, що здійснюють зв'язок через шину повідомлень.

Проста маршрутизація. Мікросервіси діють дещо як класична система UNIX: вони отримують запити, обробляють їх і відповідно генерують відповідь. Це протилежне тому, як працює багато інших продуктів, таких як ESB (Enterprise Service Buses), де використовуються високотехнологічні системи для маршрутизації повідомлень, хореографії та застосування бізнес-правил. Можна сказати, що мікросервіси мають розумні кінцеві точки, які обробляють інформацію та застосовують логіку, і німі канали, по яких інформація протікає.

На схемі 3.1 представлена мікросервісна архітектура системи оповіщення ініціаторів фінансових угод в інформаційній мережі авіакомпанії.



1. Використовуємо мікросервіс зберігання фінансових угод, розроблений раніше одногрупником Андрієвським Денисом.
2. Проектуємо мікросервіс Scheduler, котрий буде тригерити подію взяття всіх угод за останню добу.
3. Проектуємо мікросервіс EmailSender, котрий буде насилати email-нотифікації на почту постачальників.

3.2. Розробка мікросервісу Scheduler.

Створення класу Scheduler для тригерінгу потрібних функцій.

```
package aviation.nau.gumenyuk;

import aviation.nau. gumenyuk.model.Contract;

import java.util.List;

@Scheduler
public class Scheduler {

    @Autowired
    private ScheduleProperties scheduleProperties;

    @Autowired
    private ControlClient controlClient;

    @Autowired
    private EmailSenderClient emailSenderClient;

    @Scheduled(cron = "0 * * * * *")
    public void scheduleGetAllDocumentsFromDB() {

        List<Contract> contracts = controlClient.getAllDocumentsFromDb();

        emailSenderClient.sent(contracts);
    }
}
```

1. Підключаємо проперті scheduleProperties.
2. Підключаємо ControlClient для отримання списку фінансових угод з мікросервісу ControllApp.
3. Підключаємо EmailSenderClient для відправки списку фінансових угод до серверу оповіщення ініціаторів фінансових угод.
4. Реалізуємо метод відправки угод навісивши анотацію @Scheduled , котра вказує, що тригер спрацьовує кожну першу секунду кожного дня.

3.3. Розробка мікросервісу EmailSender

Для початку необхідно підключити бібліотеку відправника електронних листів.

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-mail</artifactId>
<version>2.3.4.RELEASE</version>
</dependency>
```

Інтерфейси та класи для підтримки пошти Java у середовищі Spring організовані таким чином:

Інтерфейс MailSender: інтерфейс верхнього рівня, який забезпечує базову функціональність для надсилання простих електронних листів

Інтерфейс JavaMailSender: підінтерфейс вищезазначеного MailSender. Він підтримує повідомлення MIME і в основному використовується разом із класом MimeMessageHelper для створення MimeMessage. Рекомендується використовувати механізм MimeMessagePreparator із цим інтерфейсом.

Клас JavaMailSenderImpl забезпечує реалізацію інтерфейсу JavaMailSender. Він підтримує MimeMessage та SimpleMailMessage.

Клас SimpleMailMessage: використовується для створення простого поштового повідомлення, що включає поля від, до, копію, тему та текстове поле

Інтерфейс MimeMessagePreparator забезпечує інтерфейс зворотного виклику для підготовки повідомлень MIME.

Клас MimeMessageHelper: допоміжний клас для створення повідомлень MIME. Він пропонує підтримку зображень, типових вкладень до пошти та текстового вмісту в HTML-макеті.

Реалізація HTML щаблону для заповнення деталів фінансових угод з постачальником та відправки на почту.

Далі реалізовано самий клас для оповіщення ініціаторів фінансових угод.

```
@Bean
public JavaMailSender getJavaMailSender() {
    JavaMailSenderImpl mailSender = new JavaMailSenderImpl();
    mailSender.setHost("smtp.gmail.com");
    mailSender.setPort(587);

    mailSender.setUsername("my.gmail@gmail.com");
    mailSender.setPassword("password");

    Properties props = mailSender.getJavaMailProperties();
    props.put("mail.transport.protocol", "smtp");
    props.put("mail.smtp.auth", "true");
    props.put("mail.smtp.starttls.enable", "true");
    props.put("mail.debug", "true");

    return mailSender;
    }
```

Після того, як управління залежностями та конфігурація встановлені, ми можемо використовувати вищезгаданий JavaMailSender для надсилання електронного листа.

Простий фреймворк vanilla Spring, так і його завантажувальна версія обробляють створення та відправлення електронних листів подібним чином, нам не доведеться розрізняти ці два підрозділи.

3.4. Розробка засобу оповіщення ініціаторів угод на SQL Server

Першим способом отримання інформації про угоду є email-робот, котрий відправляє інформацію про угоду за запитом. Постачальник пише лист на конкретну електронну адресу, указуючи в темі листа слово “угода”. Після цього робот слухає вхідні повідомлення кожні 5 хвилин, та якщо знаходить в темі листа слово “угода”, знаходить її в базі даних, викликає функцію збору інформаціїї про угоду, котра в свою чергу конвертує дані в шаблон HTML, відправляє у відповідь лист з інформацією про угоду.

Налаштування e-mail робота перед відправкой e-mail повідомлення ініціаторам угод. За домогою оператора SELECT, функція робить запит, використовуючи ключове слово “угода”.

```
        begin
            SELECT @ParamString=@Subject
            SELECT @Log=@Log+' PARAMETERSTRING: ParamString:'+convert(varchar(40),isnull(@ParamString,''))
            insert TRError(TRLogId,ErrorStr) values (0, @ErrorStr+' '+@Sender+': '+@Subject+' '+@log)
            exec qu_BalanceSaleMailProc @MailDate,@Subject,@Sender,@BodyTxt,@CcRecipients,@Recipients
        end
--=============================================================================================
IF @Subject like 'суточные%' or @Subject like 'добові%'
        begin
            SELECT @ParamString=@Subject
            SELECT @Log=@Log+' PARAMETERSTRING: ParamString:'+convert(varchar(40),isnull(@ParamString,''))
            insert TRError(TRLogId,ErrorStr) values (0, @ErrorStr+' '+@Sender+': '+@Subject+' '+@log)
            exec qu_dailyfoodMailProc 0,@MailDate,@Subject,@Sender,@BodyTxt,@CcRecipients,@Recipients
        end
--=============================================================================================
IF @Subject like 'питание%' or @Subject like 'харчування%'
        begin
            SELECT @ParamString=@Subject
            SELECT @Log=@Log+' PARAMETERSTRING: ParamString:'+convert(varchar(40),isnull(@ParamString,''))
            insert TRError(TRLogId,ErrorStr) values (0, @ErrorStr+' '+@Sender+': '+@Subject+' '+@log)
            exec qu_dailyfoodMailProc 1,@MailDate,@Subject,@Sender,@BodyTxt,@CcRecipients,@Recipients
        end
--=============================================================================================
IF @Subject like 'shipment%'
        begin
            SELECT @ParamString=@Subject
            SELECT @Log=@Log+' PARAMETERSTRING: ParamString:'+convert(varchar(40),isnull(@ParamString,''))
            insert TRError(TRLogId,ErrorStr) values (0, @ErrorStr+' '+@Sender+': '+@Subject+' '+@log)
            exec qu_ShipmentMailProc @MailDate,@Subject,@Sender,@BodyTxt,@CcRecipients,@Recipients
        end
--=============================================================================================
IF @Subject like 'угода%'
        begin
            SELECT @ParamString=@Subject
            SELECT @Log=@Log+' PARAMETERSTRING: ParamString:'+convert(varchar(40),isnull(@ParamString,''))
            insert TRError(TRLogId,ErrorStr) values (0, @ErrorStr+' '+@Sender+': '+@Subject+' '+@log)
            exec qu_AgreementMailProc @Subject,@Sender
        end
--=============================================================================================
IF @Subject like 'payments%'
        begin
            SELECT @ParamString=@Subject
            SELECT @Log=@Log+' PARAMETERSTRING: ParamString:'+convert(varchar(40),isnull(@ParamString,''))
            insert TRError(TRLogId,ErrorStr) values (0, @ErrorStr+' '+@Sender+': '+@Subject+' '+@log)
            exec qu_VendorsPayments_Scan @Sender,@Subject
        end
```

Рис. 3.2. Скрипт робота для запуску функції відправки e-mail повідомлень

Скрипт підготовки запиту відправки оповіщення. Дана процедура формує список необхідних полів для заповнення електронного листа.

```
ALTER PROCEDURE [dbo].[sp_send_dbmail]
    @profile_name                 sysname      = NULL,
    @recipients                   VARCHAR(MAX) = NULL,
    @copy_recipients              VARCHAR(MAX) = NULL,
    @blind_copy_recipients        VARCHAR(MAX) = NULL,
    @subject                      NVARCHAR(255) = NULL,
    @body                         NVARCHAR(MAX) = NULL,
    @body_format                  VARCHAR(20)  = NULL,
    @importance                   VARCHAR(6)   = 'NORMAL',
    @sensitivity                  VARCHAR(12)  = 'NORMAL',
    @file_attachments             NVARCHAR(MAX) = NULL,
    @query                        NVARCHAR(MAX) = NULL,
    @execute_query_database       sysname      = NULL,
    @attach_query_result_as_file BIT           = 0,
    @query_attachment_filename    NVARCHAR(260) = NULL,
    @query_result_header          BIT          = 1,
    @query_result_width           INT          = 256,
    @query_result_separator       CHAR(1)      = ' ',
    @exclude_query_output         BIT          = 0,
    @append_query_error           BIT          = 0,
    @query_no_truncate            BIT          = 0,
    @query_result_no_padding      BIT          = 0,
    @mailitem_id                  INT          = NULL OUTPUT,
    @from_address                 VARCHAR(max) = NULL,
    @reply_to                     VARCHAR(max) = NULL,
          --Changed by Vadim/Suraev@flyuia.com 12 jun 2018
          -- 0 is standard value, 1 - attachment encoded in ANSI
    @ANSI_Attachment BIT                       = 0
  WITH EXECUTE AS 'dbo'
AS
```

Рис. 3.3. Скрипт підготовки запиту відправки оповіщення

Перевірка функції запиту отримання інформації про угоду та конвертування даних в шаблон HTML.

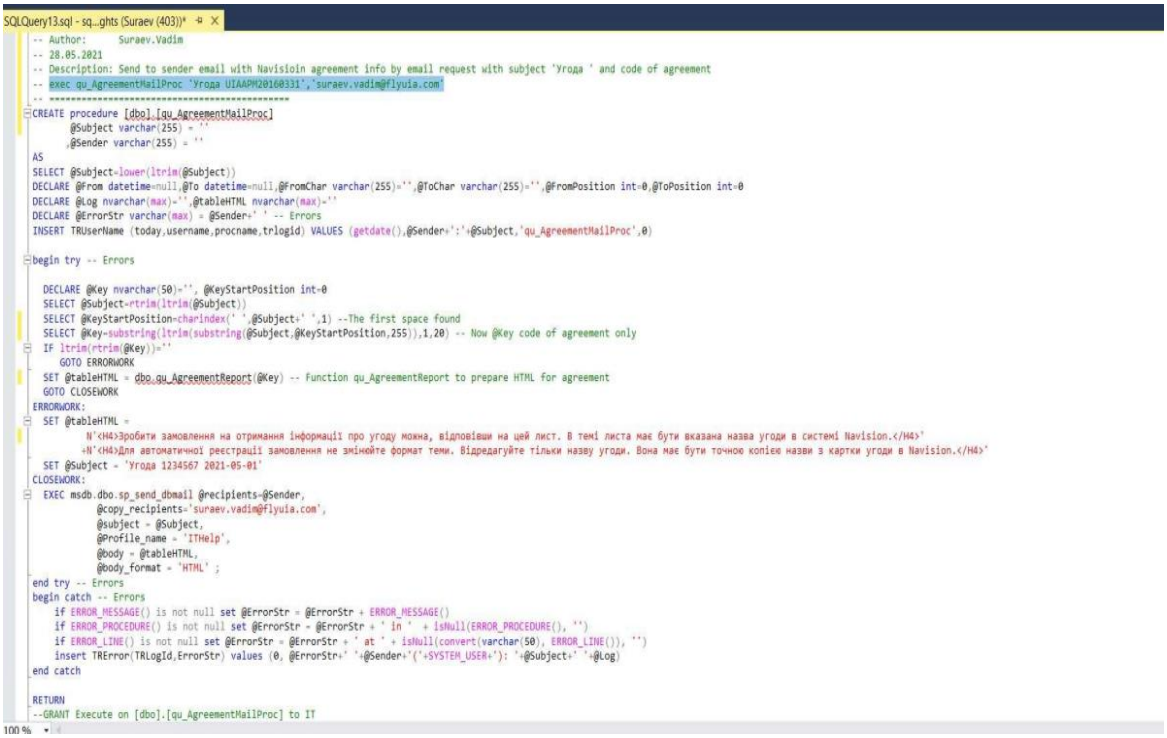


Рис. 3.4. Функція вибірки даних про угоду

Перевірка отримання листа за вказаною угодою. Після того, як ініціатор надіслав листа на корпоративну пошту інформаційної мережі авіакомпанії, робот виявив ключове слово “угода”, дістав з листа номер угоди, зібрав інформацію про потрібну угоду та відправив лист у відповідь.

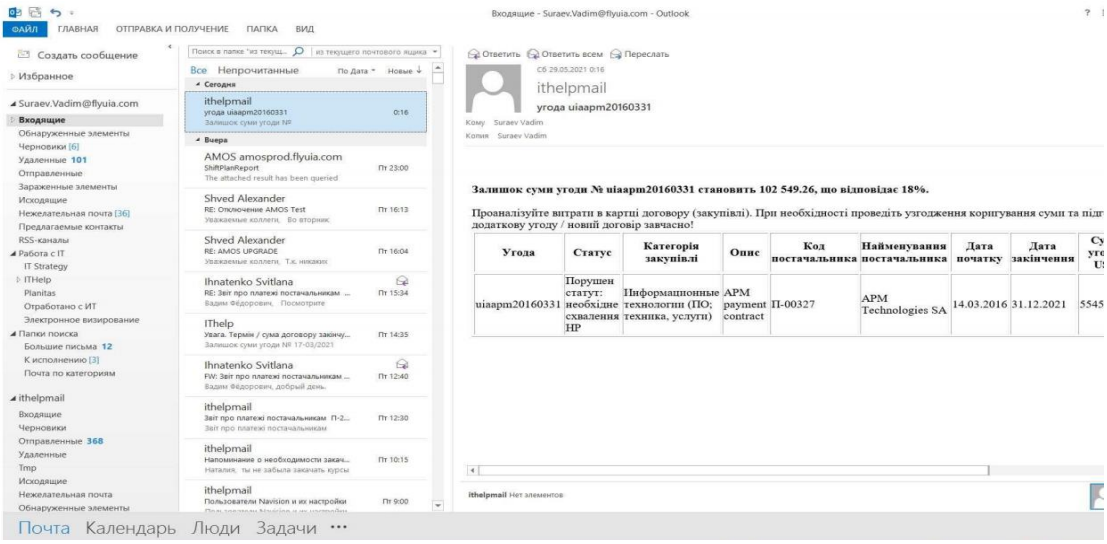


Рис. 3.5. Перевірка вхідних повідомлень

Другим способом оповіщення ініціаторів угод с постачальниками є спосіб проходження по всім актуальним угодам в базі даних, та перевірка інформації на предмет закінчення сроку дії угоди. В разі виявлення такої угоди, робот відправляє оповіщення постачальнику на предмет закінчення терміну угоди. Першим кроков є функція перебору всіх користувачів системи, ітератор проходить за списков всіх користувачів системи.

Функція перебору всіх користувачів системи.

```
CREATE PROCEDURE [dbo].[qu_AgreementCloseUserAlert]
as
DECLARE @UserId nvarchar(50)=''
DECLARE User_Cursor CURSOR FOR SELECT [User ID]
                                from [UIA_MSNAV_PrePROD].[dbo].[UIA$User Setup]
                               where [E-Mail] like '%@%'
OPEN User_Cursor
FETCH NEXT FROM User_Cursor into @userid
WHILE @@FETCH_STATUS = 0
BEGIN
   EXEC qu_AgreementCloseAlert @userid --Новая версия 10.09.2020
   FETCH NEXT FROM User_Cursor into @userid
END
CLOSE User_Cursor
DEALLOCATE User_Cursor

--GRANT EXECUTE ON qu_AgreementCloseUserAlert to IT
```

Рис. 3.6. Скрипт перебору всіх користувачів

Налаштування списку рахунків на оплату закріплених за конкретним користувачем.

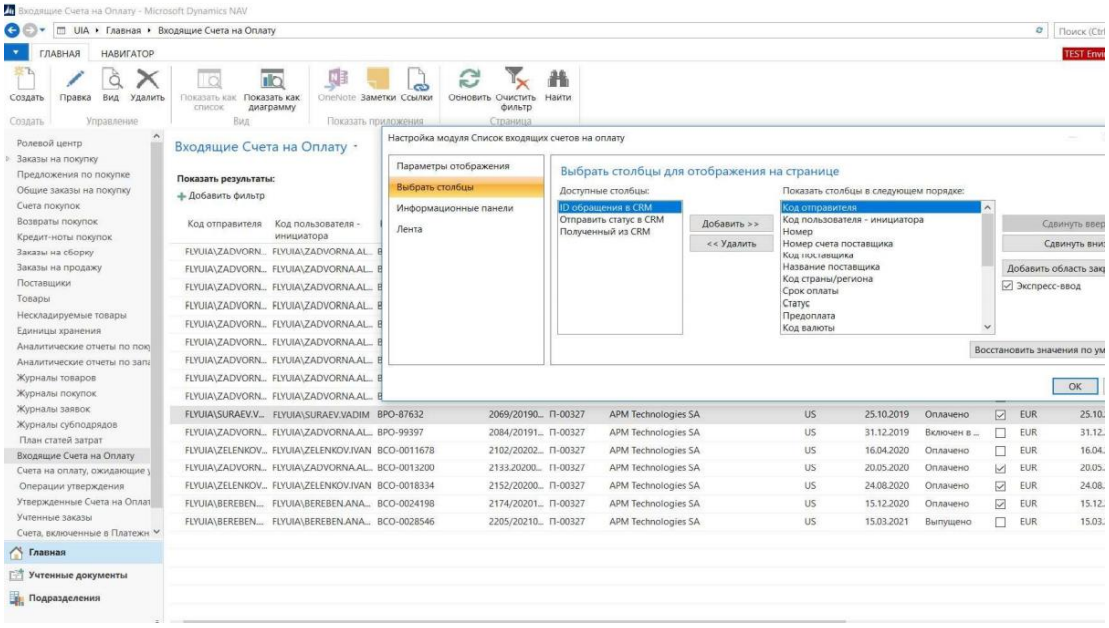


Рис. 3.7. Список рахунків на оплату

Перевірка дійсності угоди перед відправкой повідомлення про термін завершення угоди. Функція проходить по всім угодам конкретного користувача та перевіряє якщо термін дії підходить до завершення то викликає фунцію конвертування інформації про угоду в HTML шаблон та відправляє лист на пошту ініціатора.

```
-- ============================================
CREATE PROCEDURE [dbo].[qu_AgreementCloseAlert] @UserId nvarchar(50)=''
as
DECLARE @code nvarchar(20)='',@description nvarchar(250)='',@expiredate varchar(10)='',@sourcename nvarchar(50)='',@email nvarchar(100)=''
select @email=lower([E-Mail]) from [UIA_MSNAV_PrePROD].[dbo].[UIA$User Setup]
 where [UIA$User Setup].[User ID]=@UserId
DECLARE @tableHTML  NVARCHAR(MAX)='' ;
  SET @tableHTML=''
  DECLARE Agreement_Cursor CURSOR FOR
   SELECT convert(varchar(10),[Expire Date],104),[Code],[Description],[Source Name]
     from [UIA_MSNAV_PrePROD].[dbo].UIA$Agreement
    where (Code in (SELECT [Agreement No_] FROM [UIA_MSNAV_PrePROD].[dbo].[UIA$Incoming Payment Inv_ Header] where [User-Initiator Code] = @UserId)
          or ([Source Type]=0 and @UserId in (UIA$Agreement.[Created by User ID]))
          )
      and (datediff(dd,getdate(),[UIA$Agreement].[Expire Date]) between 0 and 40)

  OPEN Agreement_Cursor
  FETCH NEXT FROM Agreement_Cursor into @expiredate,@code,@description,@sourcename
  WHILE @@FETCH_STATUS = 0
  BEGIN
    SET @tableHTML = @tableHTML+
        N'</br></br>Термін дії договору <B>'+isnull(@code,'')+'</B> ('+isnull(@sourcename,'')+', '+isnull(@description,'')+') закінчується '+isnull(@expiredate,'')+'.'
    FETCH NEXT FROM Agreement_Cursor into @expiredate,@code,@description,@sourcename
  END
  CLOSE Agreement_Cursor
  DEALLOCATE Agreement_Cursor
  IF isnull(@tableHTML,'') != ''
  EXEC msdb.dbo.sp_send_dbmail @recipients=@email,
    @blind_copy_recipients=
    'Suraev.Vadim@flyuia.com' ,
    @subject = 'Увага - термін дії договору закінчується.',
    @Profile_name = 'ITHelpReturn',
    @from_address='ithelp@flyuia.com',
    @body = @tableHTML,
    @body_format = 'HTML' ;

--GRANT EXECUTE ON qu_AgreementCloseAlert to IT
```

Рис. 3.8. Перевірка дійсності угоди

Перевірка вхідних повідомлень на предмет надходження повідомлення про завершення терміну дії угоди.

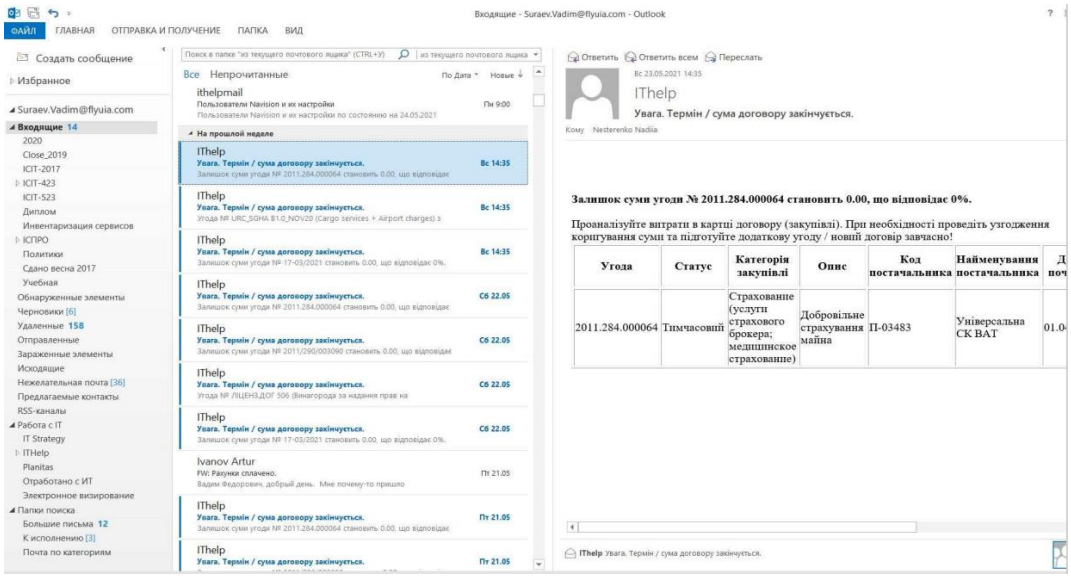


Рис. 3.9. Вхідне повідомлення про завершення угоди

Висновок до третього розділу

У третьому розділі розроблено веб-систему оповіщення ініціаторів угод з постачальниками. Використано мікросервісну архітектуру для побудування компонентів програми. На основі розробленого раніше модуля для контролю та збереження інформації про фінансові угоди з постачальниками, розроблено два мікросервіси. Мікросервіс Scheduler тригерить всі угоди в базі даних на предмет терміну дії, якщо він підходить до завершення, функція отримує інформацію про угоду та надсилає до мікросервісу EmailSender, котрий в свою чергу робить оповіщення за вказаною email-адресою.

Також розгорнуто систему на SQL сервері. Першим способом отримання інформації про угоду був розроблений email-робот, котрий відправляє інформацію про угоду за запитом.

Другим способом оповіщення ініціаторів угод с постачальниками був розроблений метод проходження по всім актуальним угодам в базі даних, та перевірка інформації на предмет закінчення сроку дії угоди.

ВИСНОВКИ

Інформаційна мережа авіакомпанії відіграє важливу роль у допомозі клієнтам. Сучасна, всеосяжна консультативна служба є фронт-офісом для всієї авіаіндрустрії і може обробляти більшість потреб і запитів користувачів без допомоги професіоналів. Підтримка замовників є єдиною точкою контакту з авіакомпанією, яка забезпечує своєчасне вирішення їх проблем. Іншими словами, якщо сервісна служба допомоги користувачам не витрачати час на нескінченний пошук фахівців, які можуть вирішити свої проблеми.

В рамках другого розділу виділено всі основні вимоги до майбутньої системи оповіщення ініціаторів угод.

У третьому розділі розроблено веб-систему оповіщення ініціаторів угод з постачальниками. Використано мікросервісну архітектуру для побудування компонентів програми. На основі розробленого раніше модуля для контролю та збереження інформації про фінансові угоди з постачальниками, розроблено два мікросервіси. Мікросервіс Scheduler тригерить всі угоди в базі даних на предмет терміну дії, якщо він підходить до завершення, функція отримує інформацію про угоду та надсилає до мікросервісу EmailSender, котрий в свою чергу робить оповіщення за вказаною email-адресою.

Також розгорнуто систему в іншому варіанті, на SQL сервері. Першим способом отримання інформації про угоду був розроблений email-робот, котрий відправляє інформацію про угоду за запитом.

Другим способом оповіщення ініціаторів угод с постачальниками був розроблений метод проходження по всім актуальним угодам в базі даних, та перевірка інформації на предмет закінчення сроку дії угоди.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жуков А.Ф., Калінський В.Д. Основи створення і використовування електронних засобів для оповіщення електронною поштою // Технічні засоби. — №3 . — 2011. С. 2-7.. 2.

2. Андреев П. П. Рассылка сообщений с помощью роботв / Андреев П. П. — М.: Международная асоциация, 2009. — 123 с.

3. Авраменко А.А., Мишина К.С., Левченко П.Р. та ін. Проектування баз даних в SQL Server " Проектування баз даних в SQL Server ": Навч.- метод, посібник / За ред. Т.О. Мешниковой. — К.: АВПВВ, 2007.-23 с.

4. Алексейчук І.С. Про способи створення системи оповіщення / І.С. Алексейчук // Нові технології навчання: Науково-методичний збірник. — К.: НМЦВД, 2000. — С.43-92.

5. Комарова Н. В. Оценка качества SQL запросов / Н. В. Комарова // Познавательный журнал "Технические советы преподавателей П. Ф. Комун". — 2004. — №23(11). — С. 131-134.

6. Алюнева, В. В. Подходы к оценке качества деятельности преподавателя вуза // Университетское управление: практика и анализ. — 2006. — № 2 (11). — С. 74-78.

7. What is MySQL? [Електронний ресурс]. — Режим доступу: http://www.restapitutorial.com/lessons/whatismysql.html.

8. Markus Egger — MVVM Survival Guide for Enterprise Architectures in Silverlight and WPF [Електронний ресурс]. — 2012. — Режим доступу: https://www.packtpub.com/application-development/mvvm-survival-guide-enterprisearchitectures-silverlight-and-wpf.