**Пояснювальна записка  
до курсової роботи**

на тему: інвойсова система “Billimo”

КПІ.ІП-14.045440.02.81

Київ – 2023

Зміст

[Перелік умовних позначень 4](#_Toc115992741)

[Вступ 5](#_Toc115992742)

[1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 6](#_Toc115992743)

[1.1 Загальні положення 6](#_Toc115992744)

[1.2 Змістовний опис і аналіз предметної області 6](#_Toc115992745)

[1.3 Аналіз існуючих технологій та успішних IT-проєктів 6](#_Toc115992746)

[1.3.1 Аналіз відомих алгоритмічних та технічних рішень 6](#_Toc115992747)

[1.3.2 Аналіз допоміжних програмних засобів та засобів розробки 7](#_Toc115992748)

[1.3.3 Аналіз відомих програмних продуктів 7](#_Toc115992749)

[1.4 Аналіз вимог до програмного забезпечення 7](#_Toc115992750)

[1.4.1 Розроблення функціональних вимог 8](#_Toc115992751)

[1.4.2 Розроблення нефункціональних вимог 9](#_Toc115992752)

[1.5 Постановка задачі 9](#_Toc115992753)

[Висновки до розділу 10](#_Toc115992754)

[2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 11](#_Toc115992755)

[2.1 Моделювання та аналіз програмного забезпечення 11](#_Toc115992756)

[2.2 Архітектура програмного забезпечення 11](#_Toc115992757)

[2.3 Конструювання програмного забезпечення 11](#_Toc115992758)

[2.4 Аналіз безпеки даних 13](#_Toc115992759)

[Висновки до розділу 13](#_Toc115992760)

[3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 14](#_Toc115992761)

[3.1 Аналіз якості ПЗ 14](#_Toc115992762)

[3.2 Опис процесів тестування 14](#_Toc115992763)

[3.3 Опис контрольного прикладу 15](#_Toc115992764)

[Висновки до розділу 15](#_Toc115992765)

[4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 16](#_Toc115992766)

[4.1 Розгортання програмного забезпечення 16](#_Toc115992767)

[4.2 Підтримка програмного забезпечення 16](#_Toc115992768)

[Висновки до розділу 17](#_Toc115992769)

[ВИСНОВКИ 18](#_Toc115992770)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 19](#_Toc115992771)

Перелік умовних позначень

|  |  |  |
| --- | --- | --- |
| IDE | – | Integrated Development Environment – інтегроване середовище розробки. |
| API | – | Application programming interface, прикладний програмний Інтерфейс |
| SDK | – | Software development kit |
| IT | – | Інформаційні технології |
| ER | – | Entity-Relation diagram |
| OC | – | Операційна система. |
| БД | – | База даних. |

Вступ

Сучасна динаміка бізнес-середовища невпинно вимагає ефективних інструментів для автоматизації та оптимізації фінансових процесів. У зв'язку з цим створено інвойсову систему Billimo з метою надання рішення для автоматизації процесу створення, відправлення та обробки інвойсів, а також для ведення обліку фінансових операцій.

Актуальність розробки Billimo зумовлена стрімким розвитком фінансових технологій та необхідністю уніфікації та оптимізації фінансових процесів для підприємств, фрілансерів та бухгалтерів. Система пропонує розв’язання у сферах, де основний акцент складається на швидкості, точності та простоті управління фінансами.

Моя робота спирається на світові тенденції розвитку програмних засобів для фінансових послуг та здобуті досягнення провідних наукових установ, організацій та фахівців галузі, а також враховує важливість інтеграції з сучасними інструментами фінансового управління.

Billimo має потенціал застосування в різних галузях та сферах бізнесу, створюючи сприятливі умови для оптимізації фінансових процесів та підвищення продуктивності роботи.

# АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Загальні положення

Звідування інтернет-простору часто відбувається завдяки веб-розробці, що включає створення веб-контенту, дизайн, програмування клієнтської/серверної частин, а також забезпечення мережевої безпеки. Ця сфера також може включати процеси, пов'язані з пошуковою оптимізацією, що впливає на рейтинг сайту у пошукових системах.

Основним артефактом веб-розробки є веб-сайти, а її фахівців часто називають веб-розробниками. В залежності від потреб власника, це може включати кодування або програмування, що забезпечує функціональність веб-сайту.

HTML, CSS і JavaScript - це основні мови, які використовуються для створення сайтів. HTML формує структуру сайту, CSS стилізує вміст, а JavaScript забезпечує взаємодію та інтерактивність.

Веб-розробники поділяються на інтерфейсних розробників (відповідальних за візуальні компоненти сайту), бекенд-розробників (управляють сервісами хостингу, базами даних) та розробників повного стеку технологій (які відповідають за всі аспекти веб-сайту).

Хоча це лише огляд веб-розробки, ця сфера - це складна система процесів і технологій, що забезпечують належне функціонування і взаємодію з користувачем у великому інтернет-просторі.

## Змістовний опис і аналіз предметної області

Вимоги до сучасних веб-додатків сьогодні стали набагато вищими, з урахуванням зростаючих очікувань користувачів. Зазвичай вони мають бути максимально доступними з будь-якого місця та на різних пристроях з різними розмірами екранів. Щоб впоратися зі зростаючим попитом, веб-додатки мають бути безпечними, гнучкими та здатними масштабуватись.

Для ефективної обробки складних сценаріїв використовуються розширені клієнтські інтерфейси, реалізовані через JavaScript та веб-API. Ці технології суттєво впливають на розробку та загальну структуру веб-додатків. Масштабованість та хмарний хостинг стали загальноприйнятими особливостями, оскільки більшість фреймворків для розробки пристосовані для роботи у хмарі – будь то загальнодоступна або приватна.

Стандартом стало кросплатформенне використання основних мов програмування в Linux, MacOS і Windows. Такі сервіси стають більш доступними та мають різноманітні можливості для розробки та використання програм.

У сучасній розробці важливо застосовувати спрощені та зрозумілі процеси розробки та розгортання. Можна використовувати прості текстові редактори, командний рядок або потужні середовища розробки для створення програм. Автоматизація розгортання за допомогою безперервної інтеграції (CI) та безперервної доставки (CD) стала стандартом.

Оскільки вимоги до розробки веб-додатків на сьогоднішній день дуже широкі, важливо обрати зрілий та розвинутий фреймворк, такий як ASP.NET Core, який надає достатню підтримку та можливості для ефективної розробки та розгортання веб-застосунків. ASP.NET Core працює оптимально з онлайн-програмами та сервісами хмарного хостингу, використовуючи лише необхідні ресурси для підвищення швидкості та безпеки програми.

## Аналіз існуючих технологій та успішних IT-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації Billimo. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

### Аналіз відомих алгоритмічних та технічних рішень

Серед веб-продуктів виділяють два основні види архітектури:моноліт та мікросервіси.

Монолітна архітектура забезпечує простоту у розробці та розгортанні ,але може бути складною для масштабування та модифікації в майбутньому. З іншого боку, мікросервіси дозволяють розбити додаток на невеликі компоненти, які можна розвивати, масштабувати та підтримувати окремо. Також, мікросервіси дають можливість використовувати різні технології для розробки одного додатку. Однак ця архітектура потребує складнішого управління та координації між сервісами.

Для монолітної архітектури характерно просте управління та швидкість розгортання, але вона може стати складною у великих і складних проектах. Мікросервіси забезпечують гнучкість у розвитку і масштабуванні, але потребують більше зусиль для управління та синхронізації між сервісами.

З моєї сторони, я вибрав мікросервісну архітектуру для моєї інвойсової системи, оскільки вона надасть мені можливість розвивати та масштабувати систему у майбутньому, навіть якщо це потребуватиме більше ресурсів.

Начало формы

### Аналіз допоміжних програмних засобів та засобів розробки

Розробка програмного забезпечення має базуватися на екосистемі з широким спектром можливостей для веб-застосувань. Однією з платформ, що відповідає цим вимогам, є Microsoft .NET Framework. Для опису функціоналу веб-застосування використовуватимемо ASP.NET Framework, який є одним з найбільших та розвинених рішень для клієнт-серверних додатків на платформі Microsoft .NET Framework.

У Microsoft.NET існує багато мов програмування, таких як C#, Visual Basic і F#. Ці мови компілюються до Common Intermediate Language (CIL), що подальше перетворюється в об’єктний код для виконання програми. Обравши C#, найбільш підтримувану та популярну мову Microsoft, ми можемо користуватися всіма перевагами Microsoft .NET Framework для розробки нашої системи.

Начало формы

### Аналіз відомих програмних продуктів

Для порівняння курсової роботи з аналогом можна скористатись таблицею 1.1

Таблиця 1.1 – Порівняння з аналогом

|  |  |  |
| --- | --- | --- |
| Функціонал | Billimo | PayPulse |
| 2fa | Так | Так |
| Надсилання інвойсів | Так | Так, обмежено |
| Оплата рахунків | Так | Так |
| Архівування інвойсів | Так | Так, обмежено |
| Інтеграція з Plaid | Так | Ні |
| Спосіб оплати через Plaid | Так | Ні |
| Апрув інвойсів | Так | Так |
| Групова оплата рахункіів | Ні | Так |

## Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є надсилання інвойсів іншим, зареєстрованим в системі компаніям, більше функцій можна побачити на рисунку 1.3.

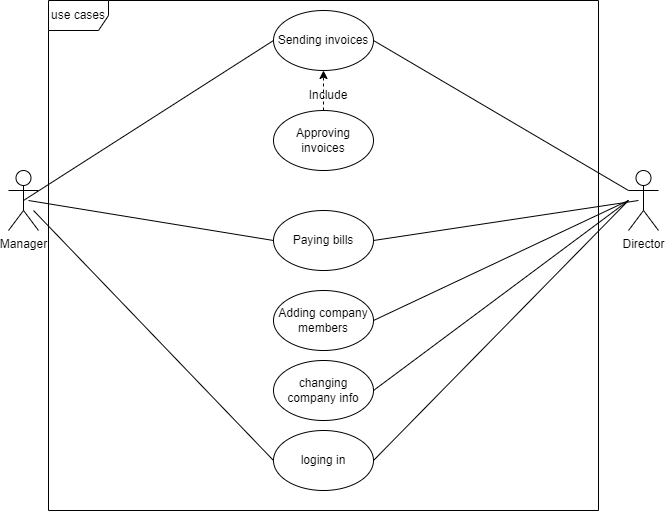


Рисунок 1.3 – Діаграма варіантів використання

В таблицях 1.2 – 1.17 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

|  |  |
| --- | --- |
| Use case name | Реєстрація користувача |
| Use case ID | UC-1 |
| Goals | Реєстрація нового користувача в системі |
| Actors | Гість (незареєстрований користувач) |
| Trigger | Користувач бажає зареєструватися |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: пошта користувача, пароль в системі, після цього користувач вводить данні про свої компанію(назва,пошта, адреса, zip-код) |
| Extension | В випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом. |
| Post-Condition | Створення кабінету користувача та реєстрація компанії в системі |

Таблиця 1.3 - Варіант використання UC-2

|  |  |
| --- | --- |
| Use case name | Надсилання інвойсу |
| Use case ID | UC-2 |
| Goals | Надсилання інвойсу отримувачу |
| Actors | Зареєстрований користувач, представник компанії(менеджер, директор) |
| Trigger | Користувач бажає відправити інвойс |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку Invoices та натискає кнопку Create invoice,у новому вікні заповнює форму,вибирає компанію отримувача,також вводить колекцію послуг та їх ціну,  В кінці натискає кнопку Send invoice |
| Extension | В випадку введення не коректних даних, під полем з’явиться помилка червоного кольору |
| Post-Condition | Надісланий інвойс який буде перетворено у рахунок для отримувача |

Таблиця 1.4 - Варіант використання UC-3

|  |  |
| --- | --- |
| Use case name | Додавання методу оплати |
| Use case ID | UC-3 |
| Goals | Додати метод оплати |
| Actors | Зареєстрований користувач, представник компанії(менеджер, директор) |
| Trigger | Користувач бажає обрати метод оплати |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку Company account,натискає кнопку Add payment method, його перенаправляють на сторінку Plaid де із списку він вибирає банк,авторизується на ньому та обирає картку з якої буде оплачувати рахунки. |
| Extension | В випадку обрання не розрахункової картки користувача буде повідомлено на сторінці Plaid |
| Post-Condition | Доданий та збережений у системі Billimo метод оплати |

Таблиця 1.5 - Варіант використання UC-4

|  |  |
| --- | --- |
| Use case name | Оплата рахунку |
| Use case ID | UC-4 |
| Goals | Оплатити надісланий користувачу інвойс |
| Actors | Зареєстрований користувач, представник компанії(менеджер, директор) |
| Trigger | Користувач бажає оплатити надісланий його компанії інвойс |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку Bills та обирає рахунок для оплати, потім натискає кнопку Pay the bill, після відкриття користувач ознайомлюється із сумою до оплати,обирає час оплати(Pay now або Pay on specific date) та спосіб оплати,в кінці користувач натискає кнопку Pay the bill |
| Extension | У випадку не обрання користувачем способу або дати оплати,кнопка Pay the bill буде не активна |
| Post-Condition | Оплачений рахунок |

Таблиця 1.6 - Варіант використання UC-5

|  |  |
| --- | --- |
| Use case name | Додавання нового користувача в компанію |
| Use case ID | UC-5 |
| Goals | Додати нового користувача в компанію |
| Actors | Незареєстрований в системі користувач, адмін або директор компанії |
| Trigger | Директор або адмін бажають додати до компанії менеджера |
| Pre-conditions | - |
| Flow of Events | Член компанії переходить на сторінку Company account та вибирає вкладку Members, натискає кнопку Add member та вводить пошту та ім’я нового користувача,новому користувачу на вказану пошту приходить повідомлення з одноразовим паролем та посиланням на форму реєстрації де новий користувач може,використовуючи одноразовий пароль,придумати собі пароль за його бажанням, потім новий користувач натискає кнопку Register |
| Extension | У випадку вказання не правильного однаразового пароля під полем форми буде вказано помилку червоного кольору |
| Post-Condition | Новий користувач у системі,та новий член компанії |

Таблиця 1.6 - Варіант використання UC-5

|  |  |
| --- | --- |
| Use case name | Двох факторна аутентифікація |
| Use case ID | UC-5 |
| Goals | Дозволити користувачам користуватись 2fa |
| Actors | Користувач системи |
| Trigger | Користувач системи бажає ввімкнути 2fa |
| Pre-conditions | - |
| Flow of Events | Користувач ввімкнувши 2fa,при наступному вході в систему буде змушений пройти двох-етапну аутентифікацію, після вводу ним пошти і пароля на пошту користувача буде відправлений лист із кодом доступу,користувач повинен буде ввести цей код у поле |
| Extension | Якщо користувач введе неправильний код три рази,йог |
| Post-Condition | Новий користувач у системі,та новий член компанії |

Таблиця 1.7 - Варіант використання UC-6

|  |  |
| --- | --- |
| Use case name | Відображення інвойсів компанії |
| Use case ID | UC-6 |
| Goals | Відобразити користувачу інвойси його компанії |
| Actors | Користувач системи |
| Trigger | Користувач системи бажає переглянути інвойси компанії |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку Invoices та вибирає вкладку для перегляду (Current, Archived) |
| Extension | При відсутності інвойсів у компанії на сторінці користувач уде повідомлений про це |
| Post-Condition | Список інвойсів компанії |

### Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.4 наведено загальну модель вимог, а в таблицях 1.8 – 1.15 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.5.



Рисунок 1.4 – Модель вимог у загальному вигляді

Таблиця 1.8 – Функціональна вимога FR-1

|  |  |
| --- | --- |
| Назва | Реєстрація користувача |
| Опис | Система повинна надавати можливість реєстрації користувачеві шляхом введення пошти, паролю. |

Таблиця 1.9 – Функціональна вимога FR-2

|  |  |
| --- | --- |
| Назва | Відправлення інвойсу |
| Опис | Система повинна надавати можливість надсилати інвойси компаніям за послуги. |

Таблиця 1.10 – Функціональна вимога FR-3

|  |  |
| --- | --- |
| Назва | Додавання нового користувача |
| Опис | Система повинна надавати можливість додавати нового члена компанії до системи |

Таблиця 1.11 – Функціональна вимога FR-4

|  |  |
| --- | --- |
| Назва | Оплата інвойсу |
| Опис | Система повинна надавати можливість оплачувати надіслані користувачу інвойси |

Таблиця 1.12 – Функціональна вимога FR-5

|  |  |
| --- | --- |
| Назва | Додавання способу оплати |
| Опис | Система повинна надавати можливість додавати спосіб оплати надісланих користувачеві інвойсів |

Таблиця 1.13 – Функціональна вимога FR-6

|  |  |
| --- | --- |
| Назва | Відображення інвойсів компанії |
| Опис | Система повинна надавати можливість перегляду інвойсів компанії |

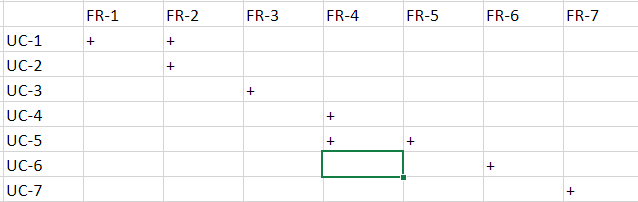


Рисунок 1.11 – Матриця трасування вимог

### Розроблення нефункціональних вимог

Опишемо нефункціональні вимоги до розроблюваного програмного забезпечення.

Реєстрація має бути захищеною, конфіденціальні дані користувача пересилатися лише в криптографічному форматі.

Форма реєстрації має бути простою та зрозумілою для користувачів, знайомою. Рівень помилок користувачів, які надсилають дані свого облікового запису на сторінці оформлення замовлення, не повинен перевищувати 10 відсотків.

## Постановка задачі

Розроблятиметься веб-додаток для системи обліку та створення інвойсів "Billimo", призначений для використання функціоналу системи з відкритим інтерфейсом дляприйняття рішень, спрямованих на оптимізацію процесів бізнесу. Мета розробки полягає у створенні програмного забезпечення для обліку та взаємодії з інвойсами. Для досягнення поставленої мети передбачається реалізація наступних завдань:

Реєстрація та авторизація користувачів.

Робота з інвойсами.

Начало формы

## Висновки до розділу

У цьому розділі "Аналіз вимог до програмного забезпечення" розглядалися питання загальних положень, що стосуються розробки програмного забезпечення для інвойсів системи Billimo. Розробка веб-додатків — складний процес, що включає дизайн, контент, та скрипти як на стороні клієнта, так і на серверному рівні. Новітні мережеві технології розширюють можливості розробки, включаючи сферу ігор та розваг. Сучасні розробки відповідають високим потребам користувачів, що вимагає ефективних рішень та використання передових технологій. Аналізуючи існуючі технології та успішні ІТ-проєкти, було виявлено ключові аспекти та архітектурні рішення, які допоможуть у реалізації системи обліку інвойсів. Також були проаналізовані вимоги до програмного забезпечення враховані у функціональних вимогах розроблюваної системи.

# МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу програмного забезпечення використовується BPMN модель (рисунок 2.1).

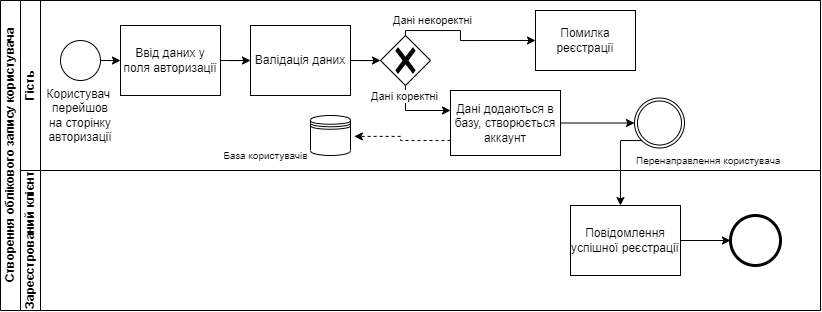


Рисунок 2.1 – BPMN модель створення облікового запису користувача

Опис послідовності створення облікового запису користувача:

* користувач переходить на сторінку авторизації;
* користувач заповнює поля авторизації;
* якщо введені поля, не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою;
* якщо введені поля відповідають шаблону, аккаунт користувача створюється, додається до бази;
* користувач отримує повідомлення про успішну реєстрацію.

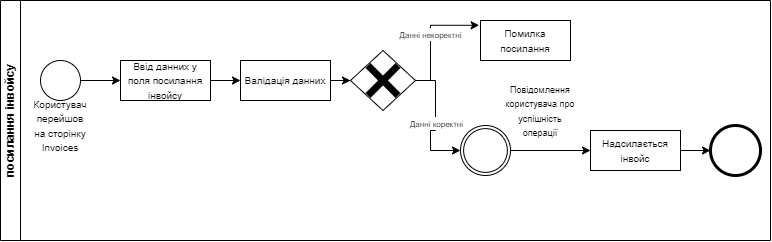


Рисунок 2.2 – BPMN модель надсилання інвойсу

Опис послідовності надсилання інвойсу

* користувач переходить на сторінку Invoices та натискає кнопку Create Invoice
* користувач обирає компанію отримувача інвойсу, вводить список послуг та їх ціни,заповнює інші поля
* якщо всі обов’язкові поля заповнені та коректні відбувається посилання інвойсу
* при помилці користувача буде повідомлено

## Архітектура програмного забезпечення

При розробці програмного забезпечення було використано клієнт-серверний архітектурний патерн.

Клієнтський застосунок використовує схему SPA(single page application).

Для розробки клієнтського застосунку була використана бібліотека Angular,так як ця бібліотека надає розробникам інструменти та практики, які сприяють швидкому розгортанню, підтримці та розширенню різних застосувань.

Для розробки серверного додатку було використано мікросервісну архітектуру.

Серверне рішення складається із таких мікросервісів

1. Invoices сервіс відповідає за обробку та зберігання інвойсів.

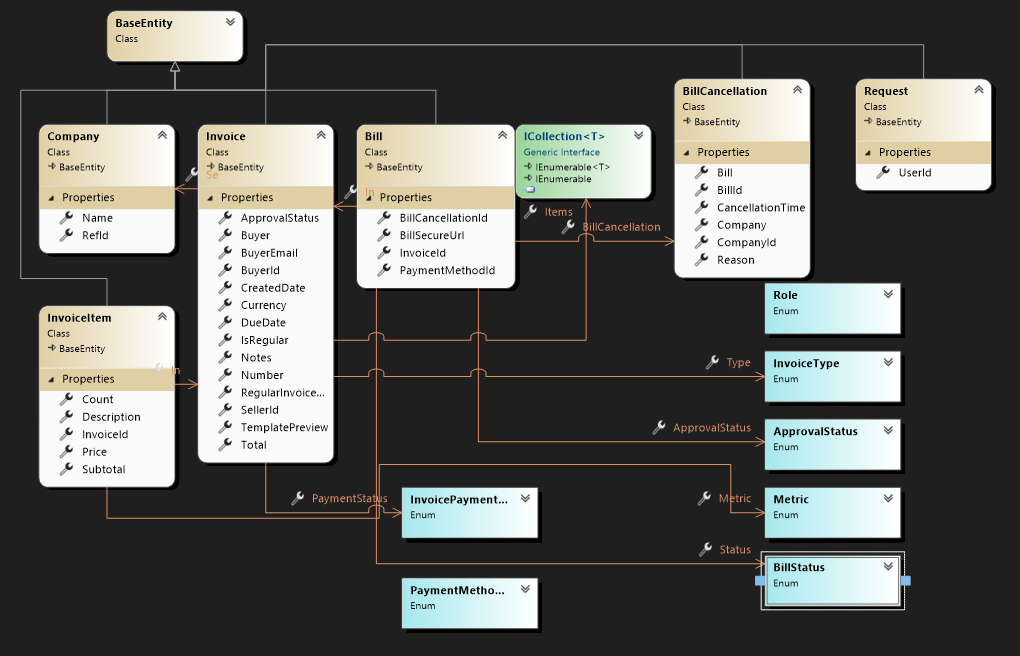


Рисунок 2.3 - Діаграма класів сервісу Invoices

1. Administration сервіс відповідає за реєстрацію користувачів та компаній.

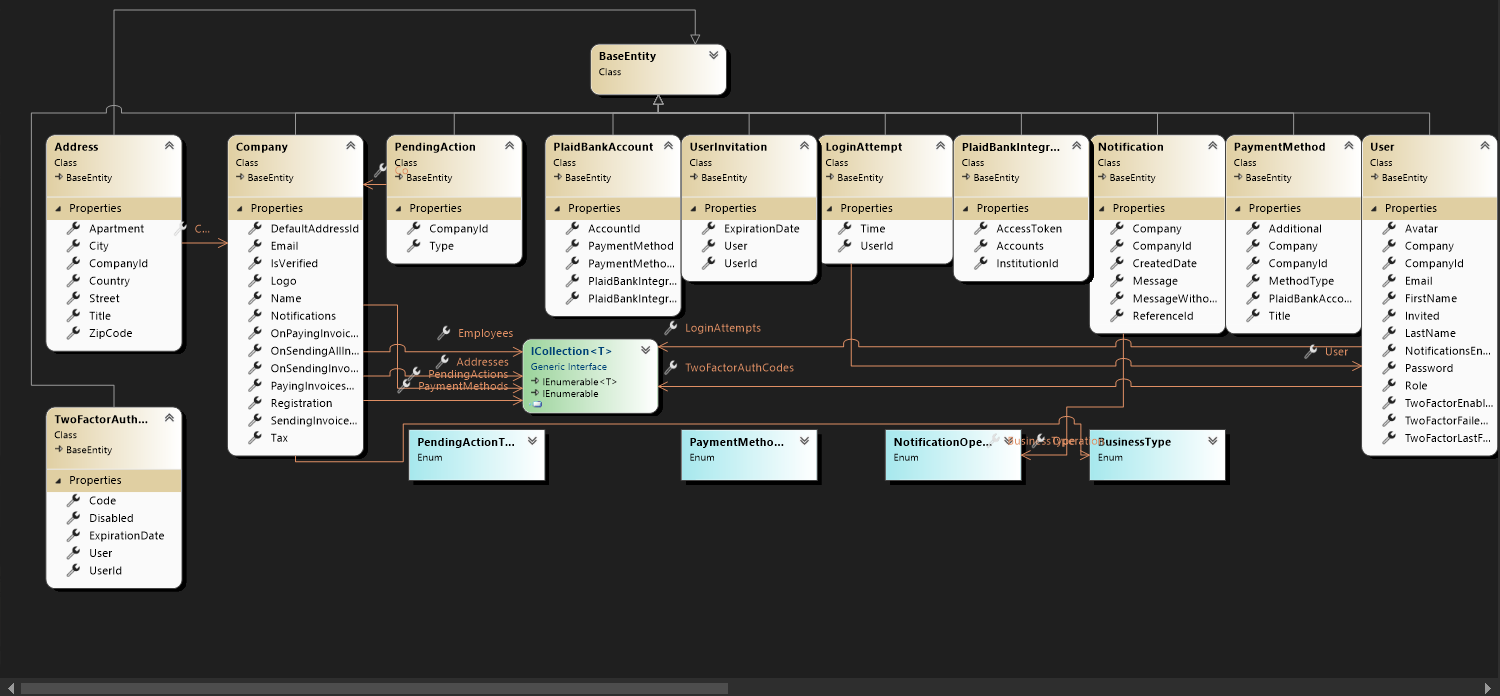


Рисунок 2.4 – Діаграма класів сервісу Administration

1. Payments сервіс відповідає за обробку платежів.

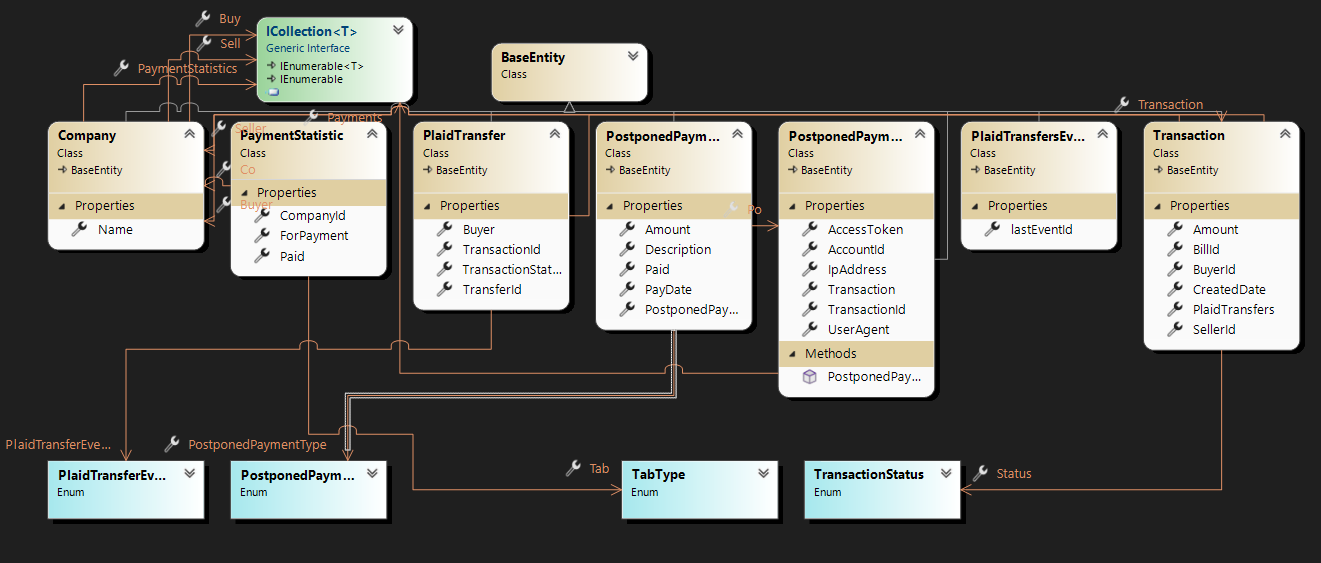


Рисунок 2.5 – Діаграма класів сервісу Payments

1. EmailService відповідає за надсилання повідомлень на пошту.

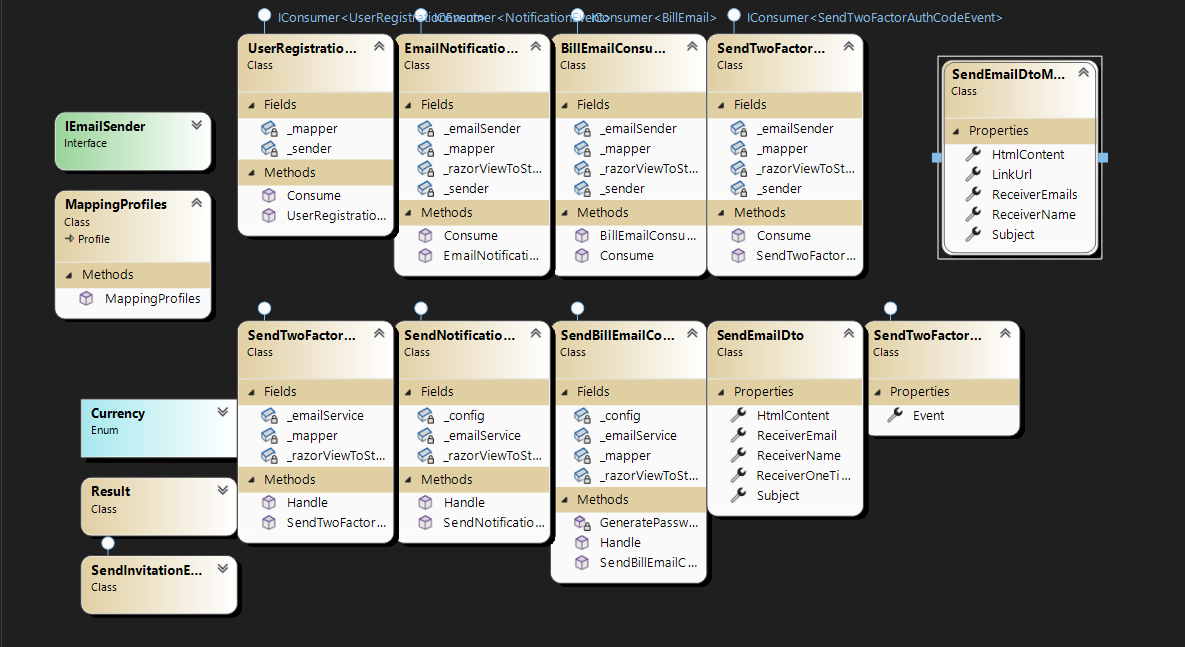


Рисунок 2.6 – Діаграма класів сервісу EmailService

Клієнтський та серверний додатки спілкуються між собою через gateway, який в свою чергу обирає відповідний мікросервіс.

На рисунку 2.6 представлено структуру всього додатку

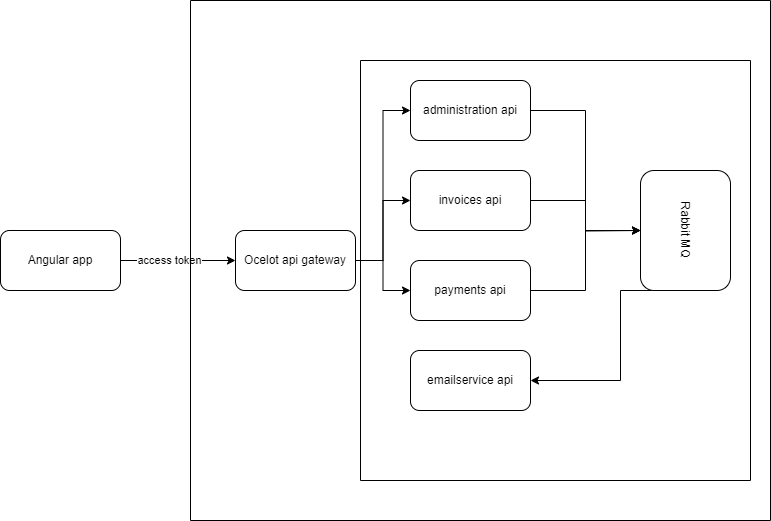


Рисунок 2.7 – Структура всього додатку “Billimo”

## Конструювання програмного забезпечення

Через великий обсяг функціоналу та різноманіття операцій з даними, для розробки бізнес логіки був обраний підхід CQRS. Цей алгоритм надає можливість ефективно впоратися зі складністю системи шляхом розділення логіки для операцій зміни та читання даних. Забезпечуючи окремі моделі для кожного типу операцій, CQRS дозволяє оптимізувати ресурси та підвищує продуктивність, сприяючи більш ефективній роботі з великим обсягом даних та вимогами до операційної ефективності. Такий підхід також полегшує масштабування системи та підтримку нового функціоналу, що дозволяє розширювати можливості системи без значних перешкод.

В якості системи управління базами даних використовується Postgres. База даних серверу призначена для зберігання користувачів, а також даних про інвойси,рахунки та інше. Опис таблиць бази даних наведено у таблицях 2.7 - 2.20. Модель баз даних наведена на рисунку 2.8 – 2.10.

Таблиця 2.7 – Опис таблиці adress

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Address | id | int | Айді адреси |
| title | varchar | Назва адреси |
| country | int | Країна |
| city | varchar | Місто |
| street | varchar | Вулиця |

Таблиця 2.8 – Опис таблиці Company

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Company | id | int | Айді компанії |
| Name | varchar | Назва компанії |
| Email | string | Пошта |
| Logo | blob | Лого |
| BusinessType | int | Тип бізнесу |

Таблиця 2.9 – Опис таблиці LoginAttempt

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| LoginAttempt | id | int | Айді |
| UserId | int | Посилання на таблицю User |
| Time | datetime | Час спроби |

Таблиця 2.10 – Опис таблиці Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Notification | id | int | Айді |
| CompanyId | int | Посилання на таблицю Company |
| Operation | int | Тип операції |
| Message | varchar | Повідомлення |
| CreatedDate | datetime | Час створення нотифіікації |

Таблиця 2.11 – Опис таблиці PaymentMethod

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| PaymentMethod | id | int | Айді |
| Title | varchar | Назва методу |
| MethodType | int | Тип методу |
| CompanyId | int | Посилання на таблицю  Company |
| PlaidBankAccount | int | Посилання на таблицю PlaidBankAcoount |

Таблиця 2.12 – Опис таблиці User

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| User | id | int | Айді |
| FirstName | varchar | Ім’я |
| LastName | varchar | Прізвище |
| Email | varchar | Пошта |
| Role | varchar | Роль |
| Password | varchar | Хешований пароль |

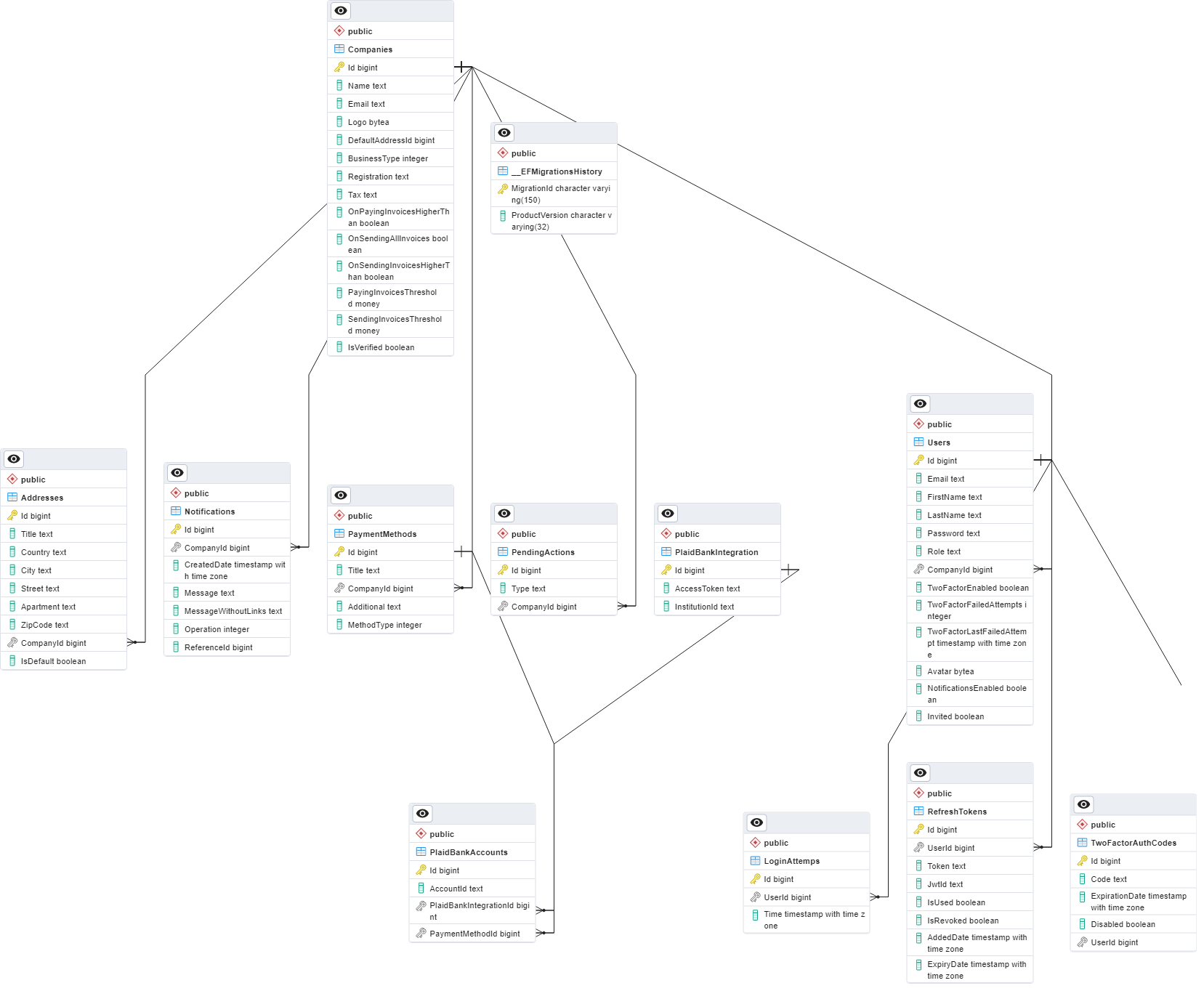


Рисунок 2.8 – Структура бази даних administration

Далі розглянемо базу даних invoices

Таблиця 2.13 – Опис таблиці Bill

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Bill | id | int | Айді |
| InvoiceId | int | Ім’я |
| PaymentMethodId | int | Посилання на таблицю PaymentMethod |
| Status | int | Статус |
|

Таблиця 2.14 – Опис таблиці Invoice

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| Invoice | id | int | Айді |
| Number | int | Номер інвойсу |
| CreatedDate | datetime | Посилання на таблицю PaymentMethod |
| DueDate | datetime | Статус |
| BuyerId | int | Посилання на таблицю  Company, а саме на покупця |
|  | SellerId | int | Посилання на таблицю  Company, а саме на продавця |
|  | Total | float | Ціна оплати |
|  | Type | int | Тип |
|  | PaymentStatus | int | Статус оплати |

Таблиця 2.14 – Опис таблиці InvoiceItem

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| InvoiceItem | id | int | Айді |
| Description | varchar | Опис послуги |
| Count | int | Кількість |
| Metric | int | Метрика,тобто вимір ціни за конкретний ітем |
| InvoiceId | int | Посилання на таблицю  Invoice |
|  | SubTotal | float | Ціна за конкретний ітем |

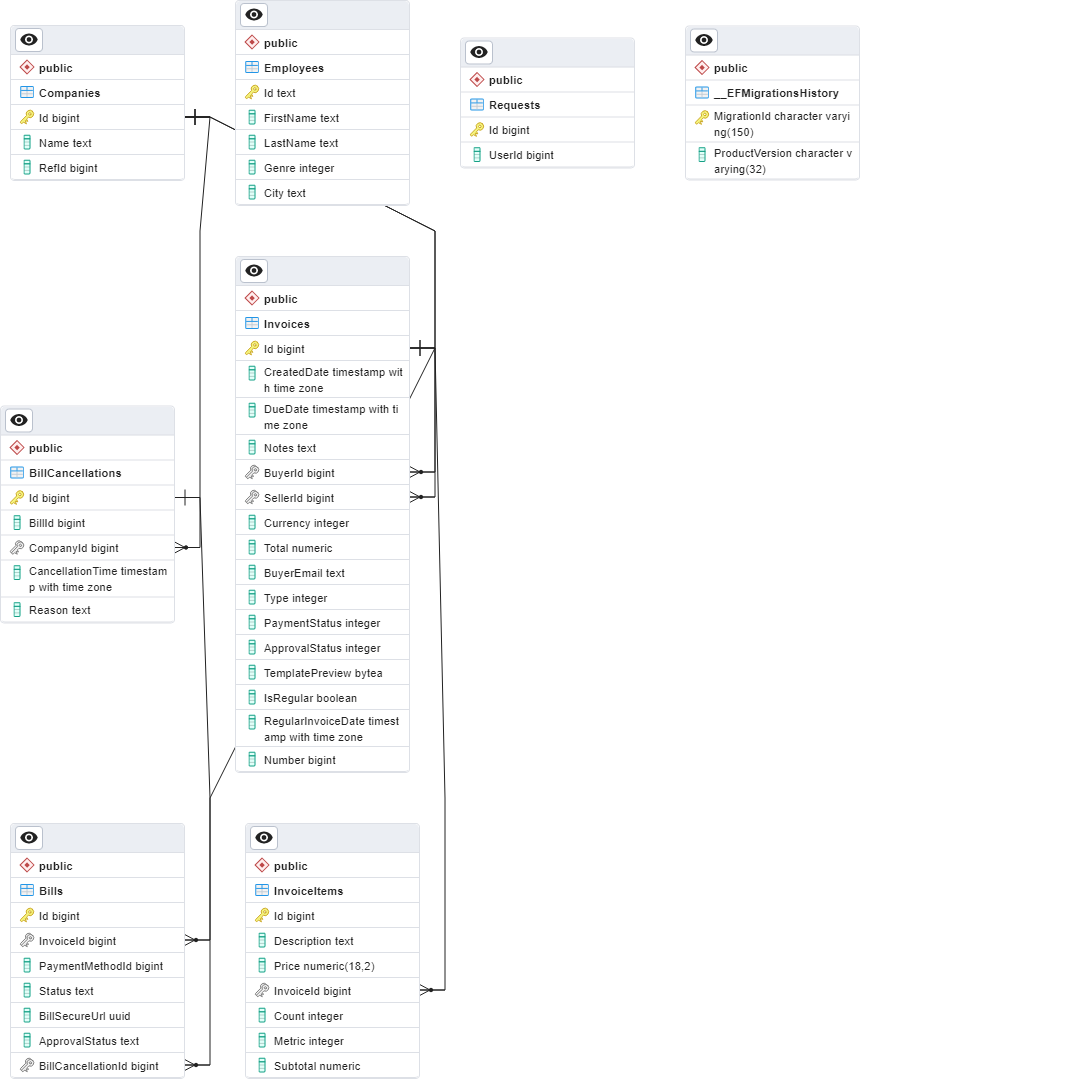


Рисунок 2.9 – Структура бази даних invoices

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.22.

Таблиця 2.3 – Опис утиліт

|  |  |  |
| --- | --- | --- |
| №  п/п | Назва утиліти | Опис застосування |
| 1 | Visual Studio | Головне середовище розробки програмного забезпечення серверної частини курсової роботи. |
| 2 | Postman | Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів. |
| 3 | Microsoft SQL Server Management Studio | Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних, для створення, маніпуляції базами. |
| 4 | Visual Studio Code | Головне середовище розробки програмного забезпечення клієнтської частини курсової роботи. |
| 5 | Rabbit MQ | Програмне забезпечення для взаємодії між сервісами серверного рішення |

Програмне забезпечення повинно функціонувати на Linux-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

* тип процесору: Intel Core i5;
* об‘єм ОЗП: 8 Гб;
* підключення до мережі Інтернет зі швидкістю від 40 мегабіт;

Рекомендована конфігурація технічних засобів:

* тип процесору: Intel Core i7;
* об‘єм ОЗП: 16 Гб;
* підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

## Аналіз безпеки даних

У застосуванні реалізовано систему реєстрації та аутентифікації \ авторизації користувачів. Для убезпечення конфіденціальних даних, які надходять від клієнта до сервера, було використано JWT. JWT, або веб-токен JSON — це Інтернет-стандарт для створення даних із необов’язковим підписом та/або необов’язковим шифруванням, корисне навантаження якого містить JSON, який заявляє певну інформацію про користувача. Токени підписуються за допомогою приватного секрету або відкритого/приватного ключа.

## Висновки до розділу

За поставленою в технічному завданні задачею, було розроблено веб застосування для автоматизації взаємодій між компаніями “Billimo”.

Серед бізнес-процесів, зокрема реєстрація, авторизація користувачів, створення інвойсів, оплата рахунків.

Для розробки було обрано клієнт-серверну архітектуру. Серверна частина застосунку представляє собою мікросервісну архітектуру розроблену з допомогою патерну CQRS.

Клієнтська частина – SPA застосунок на мові Typescript із використанням бібліотеки Angular.

Також були створені та підключені бази даних(для кожного сервісу окремо) для зберігання інформації про користувачів,інвойси та оплати. Використано ряд утиліт та бібліотек для спрощення та пришвидшення розробки.

# АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Аналіз якості ПЗ

Для аналізу якості коду було використано вбудований функціонал IDE Visual Studio 2022. Код було проаналізовано за наступними метриками:

* Індекс супроводжуваності (англ. Maintainability index) – обчислює значення індексу від 0 до 100, яке представляє відносну легкість підтримки коду. Високе значення означає кращу придатність до змін та підртримки.
* Цикломатична складність (англ. Cyclomatic complexity) – вимірює структурну складність коду. Він створюється шляхом обчислення кількості різних шляхів коду в потоці програми. Програма, яка має складний контрольний потік, потребує більше тестів для досягнення гарного покриття коду та є менш зручною для обслуговування
* Зв’язок класів (англ. Class coupling) – вимірює зв’язок з унікальними класами за допомогою параметрів, локальних змінних, типів повернення, викликів методів, загальних або шаблонних екземплярів, базових класів, реалізацій інтерфейсу, полів, визначених у зовнішніх типах, і оформлення атрибутів. Хороший дизайн програмного забезпечення диктує, що типи та методи повинні мати високу когезію та низький зв’язок. Високий зв’язок вказує на те, що проект важко повторно використовувати та підтримувати через його багато взаємозалежностей від інших типів.

На рисунках 3.1 – 3.4 можна побачити результати аналізів коду для кожного сервісу.

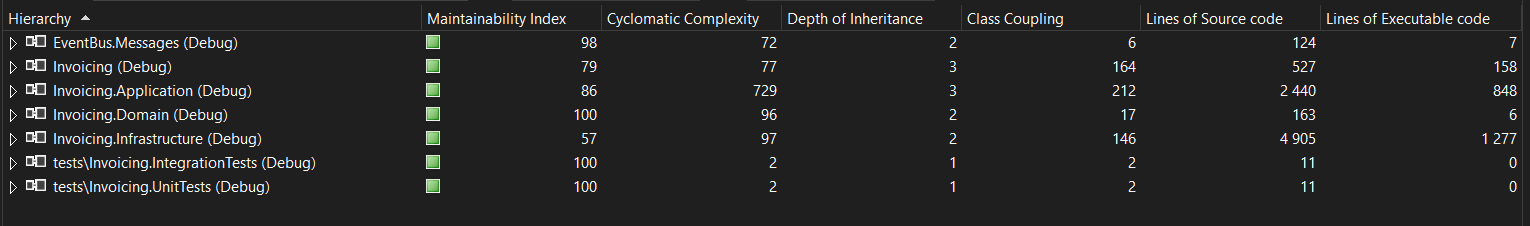


Рисунок 3.1 – Аналіз коду для сервісу Invoices

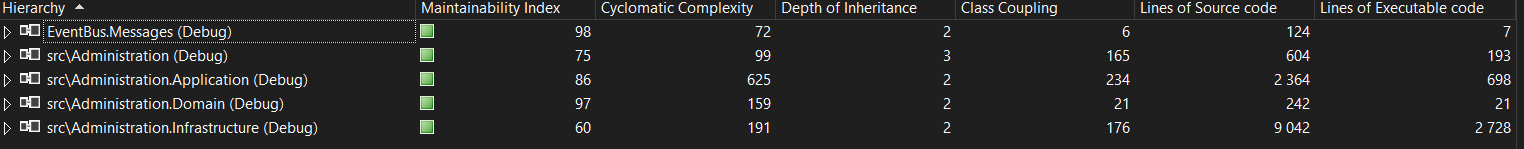


Рисунок 3.2 – Аналіз коду для сервісу Administration

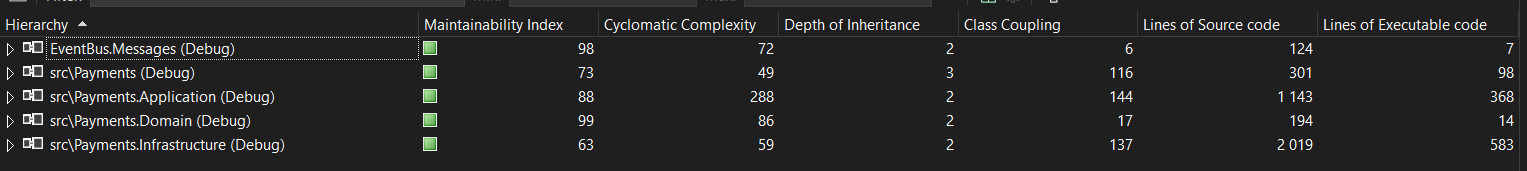


Рисунок 3.3 – Аналіз коду для сервісу Payments

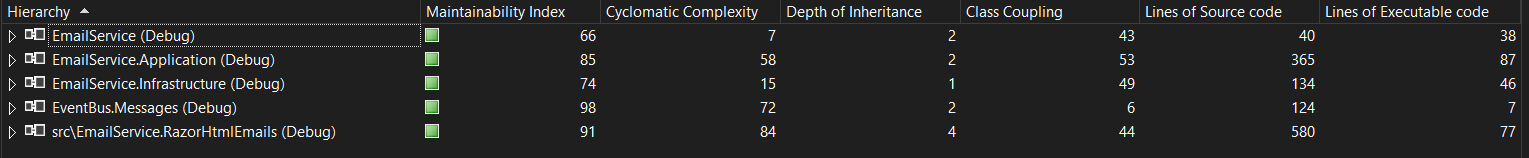


Рисунок 3.4 – Аналіз коду для сервісу EmailService

**Опис процесів тестування**

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.11.

Таблиця 3.1 – Тест 1.1

|  |  |
| --- | --- |
| Тест | Реєстрація користувача |
| Модуль | Авторизація користувача |
| Номер тесту | 1.1 |
| Початковий стан системи | Користувач знаходиться на сторінці реєстрації |
| Вхідні данні | Електронна пошта, пароль, підтвердження паролю |
| Опис проведення тесту | У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 8 до 64 символів, який містить хоча б з одну англійську літеру, одне число і один спеціальний символ,після цього користувач вводить інформацію про його компанію(назва,адреса, zip-код) в кінці користувач натискає на кнопку Register |
| Очікуваний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку. |
| Фактичний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку. |

Таблиця 3.2 – Тест 1.2

|  |  |
| --- | --- |
| Тест | Логін із двох-факторною авторизацією |
| Модуль | Авторизація користувача |
| Номер тесту | 1.2 |
| Початковий стан системи | Користувач знаходиться на сторінці входу в систему |
| Вхідні данні | Електронна пошта, пароль |
| Опис проведення тесту | У відповідні поля вводиться пошта та пароль,після цього на вказану пошту відправляється лист із кодом авторизації в систему, користувач вводить цей код у відповідне поле та натискає на кнопку Login |
| Очікуваний результат | Логін проходить успішно, користувач додається у систему і перенаправляється на головну сторінку. |
| Фактичний результат | Логін проходить успішно, користувач додається у систему і перенаправляється на головну сторінку. |

Таблиця 3.3 – Тест 2.1

|  |  |
| --- | --- |
| Тест | Додавання способу оплати |
| Модуль | Налаштування компанії |
| Номер тесту | 2.1 |
| Початковий стан системи | Користувач знаходиться на сторінці налаштувань компанії |
| Вхідні данні | - |
| Опис проведення тесту | Користувач натискає на кнопку +Add payment method після чого його буде перенаправлено на сторінку plaid де він зможе вибрати підходящий йому банк, авторизуватись у свій аккаунт та обрати картку для виплат |
| Очікуваний результат | Додавання методу оплати проходить успішно, сам метод оплати буде додано та відображено на сторінці |
| Фактичний результат | Додавання методу оплати проходить успішно, сам метод оплати буде додано та відображено на сторінці. |

Таблиця 3.4 – Тест 2.2

|  |  |
| --- | --- |
| Тест | Додавання адреси компанії |
| Модуль | Налаштування компанії |
| Номер тесту | 2.2 |
| Початковий стан системи | Користувач знаходиться на сторінці налаштувань компанії |
| Вхідні данні | - |
| Опис проведення тесту | Користувач натискає на кнопку Add address та у відповідне поле вводить новий адрес компанії,за бажанням його можна встановити як основний |
| Очікуваний результат | Додавання нової адреси проходить успішно, її буде відображено на сторінці налаштувань компанії |
| Фактичний результат | Додавання нової адреси проходить успішно, її буде відображено на сторінці налаштувань компанії |

Таблиця 3.4 – Тест 2.3

|  |  |
| --- | --- |
| Тест | Додавання нового співробітника компанії |
| Модуль | Налаштування компанії |
| Номер тесту | 2.3 |
| Початковий стан системи | Користувач знаходиться на сторінці налаштувань компанії |
| Вхідні данні | - |
| Опис проведення тесту | Користувач натискає на кнопку Add company member та у відповідні поля вводить пошту нового користувача та обирає його роль, після цього на вказану пошту приходить лист із одноразовим паролем та посиланням на сторінку авторизації, новий співробітник вводить своє ім’я пароль та одноразовий пароль та натискає на кнопку Register |
| Очікуваний результат | Додавання нового користувача пройшло успішно, він додається до списку співробітників компанії |
| Фактичний результат | Додавання нового користувача пройшло успішно, він додається до списку співробітників компанії |

Таблиця 3.5 – Тест 2.3

|  |  |
| --- | --- |
| Тест | Додавання нового співробітника компанії |
| Модуль | Налаштування компанії |
| Номер тесту | 2.3 |
| Початковий стан системи | Користувач знаходиться на сторінці налаштувань компанії |
| Вхідні данні | Тимчасовий пароль неправильний |
| Опис проведення тесту | Користувач натискає на кнопку Add company member та у відповідні поля вводить пошту нового користувача та обирає його роль, після цього на вказану пошту приходить лист із одноразовим паролем та посиланням на сторінку авторизації, новий співробітник вводить своє ім’я пароль та одноразовий пароль та натискає на кнопку Register |
| Очікуваний результат | Додавання нового користувача провалилось, користувача не зареєстровано |
| Фактичний результат | Додавання нового користувача провалилось, користувача не зареєстровано |

Таблиця 3.6 – Тест 3.1

|  |  |
| --- | --- |
| Тест | Відправлення інвойсів |
| Модуль | Відправка інвойсів |
| Номер тесту | 3.1 |
| Початковий стан системи | Користувач знаходиться на сторінці Invoices |
| Вхідні данні | Список послуг, їх метрики та одержувач інвойсу |
| Опис проведення тесту | Користувач натискає на кнопку Create Invoice і вводить данні в такі поля: обирає компанію одержувача, вводить послуги у форму(опис послуги, її метрика та кількість), після чого натискає на кнопку SendInvoice |
| Очікуваний результат | Відправка пройшла успішно, створений інвойс додався до категорії current invoices а одержувач отримав інвойс як рахунок на вкладці Bills та зможе його оплатити в майбутьому |
| Фактичний результат | Відправка пройшла успішно, створений інвойс додався до категорії current invoices а одержувач отримав інвойс як рахунок на вкладці Bills та зможе його оплатити в майбутьому |

Таблиця 3.7 – Тест 3.2

|  |  |
| --- | --- |
| Тест | Оплата рахунку |
| Модуль | Відправка інвойсів |
| Номер тесту | 3.1 |
| Початковий стан системи | Користувач знаходиться на сторінці Bills |
| Вхідні данні | Спосіб оплати та час оплати |
| Опис проведення тесту | Користувач обирає який рахунок потрібно оплатити, натискає на кнопку Pay the bill, після чого обирає спосіб оплати(картку для виплат) та час оплати(pay now або pay at specific type) після чого натискає на кнопку Pay |
| Очікуваний результат | Оплата пройшла успішно, з аккаунту компанії були зняті кошти а на компанію відправника інвойсу кошти були нараховані,статусу рахунку тепер Paid |
| Фактичний результат | Оплата пройшла успішно, з аккаунту компанії були зняті кошти а на компанію відправника інвойсу кошти були нараховані,статусу рахунку тепер Paid |

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Розгортання програмного забезпечення

<У підрозділі викладають:

повний опис покрокового розгортання ПЗ. Кроки доповнюють ілюстраціями. Обсяг 2 сторінки, чи більше.>

Клієнтську і серверну частини програмного забезпечення було вирішено розгорнути на платформі Heroku. Для розгортання було використано сервіс GitHub Actions, який надає можливості для постійної інтеграції і розгортання [13].

Розгортання починається коли новий код застосунку доставляється у репозиторій у гілку main. Тоді у середовищі GitHub Actions створюється Docker image за допомогою Dockerfile, що знаходиться у проекті. Цей image розгортається у Heroku за допомогою пакету heroku-deploy. Інформацію про розгортання клієнтської і серверної частини проекту можна побачити на рисунках 4.1 і 4.2.

Рисунок 4.1 - Інформація про розгортання клієнту

## Підтримка програмного забезпечення

<У підрозділі викладають:

опис того, як буде виконуватись підтримка програмного забезпечення. Ілюстрації. Обсяг 1 сторінка, чи більше.>

Користувачі повинні мати можливість отримати нову версію консольного застосунку з кожною версією. До того ж кожна нова версія консольного застосунку повинна бути опублікована в npm. Для автоматизації цього процесу був використаний сервіс GitHub Actions.

Створення нового випуску починається, коли нова версія консольного застосунку доставляється у репозиторій у гілку main, тобто коли commit має tag формату “v\*.\*.\*.”, де замість “\*” знаходиться число. Тоді у середовищі GitHub Actions встановлюється NodeJS. Після цього для проекту встановлюються залежності і проект збирається. Bash скрипт за допомогою бібліотеки pkg генерує виконувані файли (executables) для Linux і для Windows, та пакує файл для Linux у .deb пакет. Після цього .deb пакет і файл для Windows архівуються,

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# ВИСНОВКИ

У висновках викладають найважливіші наукові й практичні результати роботи та наводять:

* оцінку одержаних результатів і їх відповідність сучасному рівню наукових і технічних знань;
* ступінь впровадження та можливі галузі або сфери використання результатів роботи;
* наукову, науково-технічну, соціально-економічну значущість роботи;
* доцільність продовження досліджень за відповідною тематикою тощо.

Також у висновках необхідно відобразити стан вирішення усіх поставлених в курсовій роботі задач.

В результаті виконання курсової роботи було спроєктовано …

В якості середовища розробки обрано …

У якості БД використано …

Після реалізації застосунку він був протестований на пристроях з різними версіями Android, з різними розмірами екранів щоб переконатися, що додаток акуратно відображається на різних пристроях.

Наукова новизна роботи (якщо вона є) полягає в наступному (достатньо вказати щось одне).

Вперше:

* реалізовано можливість запитів від пацієнта до лікаря;
* використано те-то, що дозволило те-то.

Модифіковано:

* те-то, що дозволило те-то.

Набуло подальший розвиток:

* те-то, що дозволило те-то.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. [Оформлення бібліографії ДСТУ 7.1:2006](https://drive.google.com/file/d/1VUr7fwKHOk8t1u8sh-Sv2dlEc_6OBoIS/view?usp=sharing)
2. [Оформлення бібліографії ДСТУ 8302-2015](https://drive.google.com/file/d/17RYEYVnSXolL0S8D1k85VfReYd2yC6WO/view?usp=sharing) з прикладами