|  |  |
| --- | --- |
|  | WYŻSZA SZKOŁA EKONOMII  I INFORMATYKI W KRAKOWIE |

Piotr Dziadkowiec

nr albumu 13638

**PROJEKT I IMPLEMENTACJA USŁUGI SYSTEMOWEM „PC-MARKET-SERVICE”**

Praca inżynierska

napisana pod kierunkiem

Prof. …………

**Kraków 2024 r.**

**Streszczenie**

Celem niniejszej pracy inżynierskiej jest przedstawienie projektu zaawansowanego systemu do automatyzacji kontroli płatności, wykorzystującego technologie .NET i język C# zgodnie z zasadami programowania obiektowego i zasadami SOLID, które zapewniają wysoką jakość kodu i jego łatwość w utrzymaniu. Praca została zmotywowana potrzebą usprawnienia procesów finansowych w przedsiębiorstwach poprzez zminimalizowanie błędów ludzkich i przyspieszenie operacji płatniczych. System integruje się z istniejącymi bazami danych przedsiębiorstwa, co pozwala na automatyczne przetwarzanie danych o płatnościach i ich terminach.

Analiza projektu obejmowała identyfikację wymagań funkcjonalnych i pozafunkcjonalnych, opracowanie architektury systemu, a także implementację poszczególnych modułów systemu, w tym modułów konfiguracyjnych, emailowych, logowania i zarządzania bazą danych. Testowanie systemu odbyło się na kilku poziomach: testy jednostkowe, integracyjne oraz testy akceptacyjne użytkownika, które potwierdziły zgodność systemu z wymaganiami biznesowymi i jego gotowość do wdrożenia.

System został przygotowany do wdrożenia w realnym środowisku biznesowym, co obejmowało przygotowanie środowiska produkcyjnego, migrację danych, oraz szereg testów integracyjnych potwierdzających współpracę z zewnętrznymi systemami ERP. Wdrożenie systemu uwzględniało również przeszkolenie użytkowników końcowych oraz przygotowanie wsparcia technicznego. Monitoring i utrzymanie systemu zaplanowano jako ciągłe procesy, mające na celu zapewnienie stabilności i optymalizacji działania systemu w przyszłości.

**Słowa kluczowe**

Usługa serwisowa, C#, .Net, Sql-Server, Pc-Market

**Abstract**

This engineering thesis is focused on the design and implementation of an advanced payment control automation system using .NET technologies and the C# language, adhering to object-oriented programming principles and SOLID principles for high code quality and maintainability. The motivation for this work stemmed from the need to improve financial processes in enterprises by reducing human errors and speeding up payment operations. The system integrates with existing company databases, allowing for the automatic processing of payment data and their deadlines.

The project analysis included identifying functional and non-functional requirements, developing the system architecture, and implementing various system modules, including configuration, email, logging, and database management modules. The system underwent several levels of testing: unit tests, integration tests, and user acceptance tests, which confirmed that the system met business requirements and was ready for deployment.

The system was prepared for deployment in a real business environment, which involved setting up the production environment, data migration, and a series of integration tests confirming cooperation with external ERP systems. The deployment of the system also included training end-users and preparing technical support. Continuous monitoring and maintenance of the system were planned to ensure its stability and optimize its operation in the future.

**Keywords**

Usługa serwicowa, C#, .Net, Sql-Server, Pc-Market

# Spis treści

[Spis treści 3](#_Toc164875875)

[Wstęp 5](#_Toc164875876)

[1. Tworzenie usługi windowsowej - analiza zagadnień teoretycznych 7](#_Toc164875877)

[1.1 C# najważniejsze aspekty języka oraz słabości 7](#_Toc164875878)

[1.2 Rust najlepsze zastosowania oraz braki 8](#_Toc164875879)

[1.3 JavaScript użyteczność z zastosowaniem 9](#_Toc164875880)

[1.4 Platforma programistyczna Visual Studio 2022 10](#_Toc164875881)

[1.5 Platforma programistyczna Visual Studio Code 15](#_Toc164875882)

[1.6 Podsumowanie wyboru Visual Studio 2022 jako preferowanej platformy programistycznej 18](#_Toc164875883)

[2. Cele i założenia projektowe 19](#_Toc164875884)

[2.1 Cele projektowe 19](#_Toc164875885)

[2.2 Wymagania funkcjonalne i pozafunkcjonalne 20](#_Toc164875886)

[2.3 Wymagania funkcjonalne: 20](#_Toc164875887)

[2.4 Wymagania pozafunkcjonalne: 20](#_Toc164875888)

[2.5 Analiza profilu użytkownika 20](#_Toc164875889)

[2.6 Analiza projektu i budowy usługi 21](#_Toc164875890)

[2.7 Schemat działania aplikacji – Diagram UML 23](#_Toc164875891)

[2.8 Schemat analizy bazy danych - Diagramy UML 24](#_Toc164875892)

[2.9 Koncepcja konfiguracji serwisu 26](#_Toc164875893)

[2.10 Podział struktury projektu na części 27](#_Toc164875894)

[2.11 Testowanie i Walidacja Systemu 29](#_Toc164875895)

[2.11.1 Strategia testowania 30](#_Toc164875896)

[2.11.2 Walidacja systemu 30](#_Toc164875897)

[2.11.3 Proces walidacji 30](#_Toc164875898)

[2.12 Wdrożenie i Eksploatacja 31](#_Toc164875899)

[Inicjacja wdrożenia systemu wymaga dokładnego planowania i koordynacji: 31](#_Toc164875900)

[2.12.1 Przygotowanie środowiska produkcyjnego: 31](#_Toc164875901)

[2.12.2 Migracja danych i testy integracyjne: 31](#_Toc164875902)

[2.12.3 Wdrożenie usługi: 31](#_Toc164875903)

[2.12.4 Szkolenie użytkowników i wsparcie techniczne: 31](#_Toc164875904)

[2.12.5 Monitorowanie i utrzymanie: 31](#_Toc164875905)

[2.12.6 Zapewnienie ciągłości działania: 31](#_Toc164875906)

[2.13 Podsumowanie rozdziału drugiego 32](#_Toc164875907)

[3. Proces implementacji i dokumentacja techniczna 33](#_Toc164875908)

[3.1 Konfiguracja środowiska pod usługę oraz instalacja 34](#_Toc164875909)

[3.1.1 Krok 1 - Przygotowanie systemu : 34](#_Toc164875910)

[3.1.2 Krok 2 - Ustawienie serwera baz danych : 34](#_Toc164875911)

[3.1.3 Krok 3 - Instalacja i konfiguracja usługi : 35](#_Toc164875912)

[3.1.4 Krok 4 - Testowanie i weryfikacja : 35](#_Toc164875913)

[3.2 Metodyka zaciągania danych 36](#_Toc164875914)

[3.3 Metodyka obróbki danych oraz mapowania 38](#_Toc164875915)

[3.4 System przebudzający usługę 40](#_Toc164875916)

[Wnioski końcowe 42](#_Toc164875917)

[4. Podsumowanie 42](#_Toc164875918)

[Bibliografia 43](#_Toc164875919)

# Wstęp

W erze powszechnej cyfryzacji procesy biznesowe podlegają nieustannym przemianom, wymuszając na przedsiębiorstwach adaptację do szybko zmieniających się warunków rynkowych. Jednym z kluczowych obszarów, gdzie automatyzacja przynosi znaczące korzyści, jest zarządzanie finansami, a w szczególności kontrola płatności. W odpowiedzi na te potrzeby, niniejsza praca inżynierska koncentruje się na projektowaniu i implementacji zaawansowanego systemu do automatyzacji kontroli płatności, wykorzystującego nowoczesne technologie programistyczne i najlepsze praktyki inżynieryjne.

Głównym celem pracy jest stworzenie usługi, która nie tylko usprawni procesy kontrolowania i zarządzania płatnościami, ale także zminimalizuje ryzyko błędów ludzkich, przyspieszy operacje finansowe i zwiększy ogólną transparentność finansową przedsiębiorstwa. Projekt ma na celu wykorzystanie technologii .NET oraz języka programowania C#, które oferują zarówno elastyczność w rozwoju aplikacji, jak i solidne wsparcie dla zasad SOLID[[1]](#footnote-1), co jest niezbędne dla utrzymania wysokiej jakości kodu i jego długoterminowego utrzymania.

Automatyzacja płatności staje się standardem w wielu branżach, jednak wiele firm wciąż polega na manualnych lub półautomatycznych systemach, które są podatne na błędy i nieefektywne. Wprowadzenie kompleksowego systemu automatyzacji, który integruje się z istniejącymi bazami danych i systemami ERP (Enterprise Resource Planning), otwiera nowe możliwości dla optymalizacji procesów i zwiększenia konkurencyjności firmy na rynku.

Praca została podzielona na kilka głównych części, zaczynając od dogłębnej analizy wymagań funkcjonalnych i niefunkcjonalnych, które są niezbędne do zrozumienia potrzeb i oczekiwań użytkowników końcowych. Następnie przedstawiono projekt architektury systemu, który uwzględnia zarówno modułowość, jak i skalowalność przyszłego rozwiązania. Implementacja systemu pokazuje praktyczne zastosowanie wybranych technologii i metod, podczas gdy etap testowania i walidacji koncentruje się na zapewnieniu niezawodności i bezpieczeństwa aplikacji.

W pracy zastosowano metodykę Agile, która umożliwia elastyczne zarządzanie projektem i adaptację do zmieniających się wymagań w trakcie jego realizacji. Testy jednostkowe, integracyjne oraz akceptacyjne (UAT) pozwalają na bieżącą weryfikację funkcjonalności i bezpieczeństwa systemu, zapewniając, że finalny produkt spełni zarówno techniczne, jak i biznesowe kryteria jakości.

Ostatecznym celem jest dostarczenie narzędzia, które poprawi efektywność operacji finansowych, zredukuje koszty operacyjne i zwiększy satysfakcję klientów dzięki szybszemu i bardziej transparentnemu procesowi rozliczeń. System ma potencjał do dalszego rozwoju i adaptacji w różnych środowiskach biznesowych, co może być przedmiotem przyszłych badań i rozwoju.

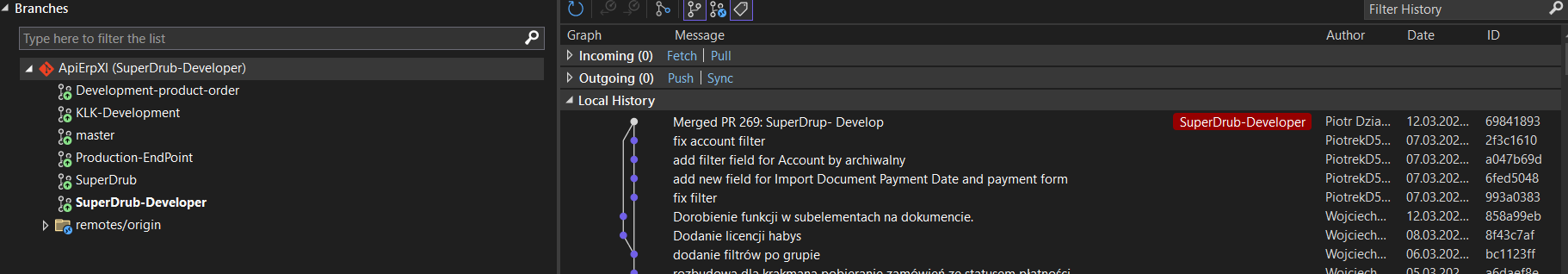
# Tworzenie usługi windowsowej - analiza zagadnień teoretycznych

W niniejszym rozdziale przedstawione zostaną kluczowe zagadnienia związane z projektowaniem i tworzeniem usług dla systemu Windows. Analiza obejmie porównanie trzech technologii backendowych: C#, Rust oraz JavaScript, pod kątem ich struktury, funkcjonalności oraz dojrzałości technologicznej. Każdy z języków zostanie oceniony ze względu na oferowane możliwości, co umożliwi zrozumienie ich potencjału w kontekście przyszłego rozwoju i możliwości adaptacji projektu. Szczególna uwaga zostanie zwrócona na dokładność i szybkość działania każdej z technologii.

## C# najważniejsze aspekty języka oraz słabości

C# jest jednym z bardziej popularnych języków obiektowych, stworzony przez Microsoft w 2002 roku. Pierwsze oficjalne wydanie, C# 1.0, miało na celu dostarczenie uniwersalnego narzędzia dla programistów, przypominającego strukturą język Java, lecz dedykowanego dla platformy Windows. Mimo początkowych sukcesów, C# miał szereg ograniczeń, które były stopniowo eliminowane w nowszych wersjach, wprowadzanych na rynek w odpowiedzi na rosnące potrzeby użytkowników. Kluczową platformą uruchamiającą w tamtym czasie było Visual Studio .NET 2002, które również ewoluowało, oferując coraz to bardziej zaawansowane funkcje dla użytkowników[[2]](#footnote-2).

W obecnym projekcie został zastosowany C# w wersji 12 i .NET 8, najnowsze wydanie frameworka Microsoftu. Wybór ten podyktowany był znacznymi ulepszeniami wprowadzonymi od wersji 1.0, które zwiększyły funkcjonalność języka oraz jego kompatybilność z innymi platformami. Visual Studio 2022, które używam, oferuje wygodny interfejs użytkownika, ułatwiający zarządzanie gałęziami projektu na platformie GitHub, tworzenie pull requestów oraz integrację z zewnętrznymi projektami. Rozbudowane biblioteki i dodatki dostępne dla C# i jego frameworka pomagają zwiększyć klarowność kodu i optymalizują cały projekt[[3]](#footnote-3).



Screen shoot – zakładki branch do Manage github  
 źródło: projekt

Mimo że C# jest językiem wysokiego poziomu, posiada również swoje wady. Nie jest tak szybki jak niektóre inne języki programowania, co może wpływać na czas oczekiwania podczas walidacji danych. Ponadto, jego dokumentacja, choć obszerna, jest czasem skomplikowana, co może stanowić wyzwanie dla programistów.

## Rust najlepsze zastosowania oraz braki

Rust, nowoczesny język programowania, został oficjalnie wprowadzony do użytku w 2010 roku. Jest to wieloparadygmatowy język, który ciągle jest rozwijany i wspierany przez Mozillę. Język ten został zaprojektowany z myślą o kluczowych aspektach takich jak bezpieczeństwo, równoległość i wydajność, co czyni go szczególnie cenionym wśród języków niskopoziomowych. Jego mechanizm zarządzania pamięcią, który nie wymaga użycia garbage collector, jest często podkreślany jako jedna z jego największych zalet.

Kluczowe zalety Rusta to bezpieczeństwo, które jest wspomagane przez mechanizmy takie jak Borrow Checker i Lifetime. Te elementy języka pomagają w zarządzaniu pamięcią i zabezpieczają referencje do danych, co jest realizowane poprzez:

* &T - umożliwia wielokrotne referencje bez ich modyfikacji,
* &mut T - pozwala na jedną referencję, która może modyfikować dane.

Jedną z najbardziej cenionych cech Rusta jest jego wydajność. Język został zaprojektowany tak, aby był jak najbardziej efektywny, porównywalnie z innymi językami niskopoziomowymi. Niskopoziomowość Rusta pozwala programistom na bezpośredni dostęp do zasobów sprzętowych, co jest kluczowe w aplikacjach systemowych lub wbudowanych, oraz w miejscach, gdzie wymagana jest wysoka wydajność obliczeniowa[[4]](#footnote-4).

Dynamiczny rozwój infrastruktury backendowej to kolejna zaleta Rusta. Język ten jest często wybierany do rozbudowy i rozwoju niezawodnych, wysokowydajnych serwerów oraz rozwiązań backendowych. Dzięki swojej strukturze i wydajności, Rust jest idealnym wyborem dla dużych organizacji lub korporacji, które wymagają tworzenia skalowalnych mikro usług lub API, zdolnych do obsługi dużych obciążeń i szybkiego reagowania na zapytania od użytkowników[[5]](#footnote-5).

Mimo wielu zalet, Rust ma także swoje słabości. Najważniejszą z nich jest stroma krzywa uczenia się, spowodowana jego unikalnymi funkcjami takimi jak system własności i zasady pożyczania, co może być wyzwaniem dla początkujących developerów. Dodatkowo, ograniczone zasoby bibliotek i gotowych rozwiązań mogą wydłużać czas oraz zwiększać koszty projektów. Rust charakteryzuje się również długim czasem kompilacji, co wynika z silnego typowania i szczegółowej analizy czasu kompilacji, co może wpływać na tempo pracy programistycznej.

Podsumowując, Rust to język, który mimo swojej młodości, prosperuje w znaczącym tempie i oferuje wiele zalet związanych z bezpieczeństwem i wydajnością. Warto jednak zwrócić uwagę na jego wady, które mogą wpłynąć na decyzję o jego stosowaniu w konkretnych projektach.

## JavaScript użyteczność z zastosowaniem

JavaScript to język programowania, który zyskał dużą popularność zarówno w obszarze frontendu, jak i backendu, dzięki frameworkowi Node.js. To ułatwia tworzenie kompleksowych aplikacji przez indywidualnych programistów lub zespoły deweloperskie. Język ten oferuje bogaty zasób bibliotek i frameworków, co przyspiesza proces tworzenia produktów i ułatwia zarządzanie projektami. Jest on również wspierany przez szeroką społeczność programistów, co zapewnia bieżące wsparcie i ciągły rozwój, co bezpośrednio przekłada się na szybkość kompilacji w projektach i aplikacjach[[6]](#footnote-6).

Mimo swoich zalet, JavaScript ma również swoje wady. Jego główna słabość to predyspozycje do zastosowań frontendowych, które mogą rodzić problemy z kompatybilnością między różnymi przeglądarkami. Chociaż sytuacja ta uległa poprawie, nadal stanowi wyzwanie dla wielu programistów. Bezpieczeństwo jest innym obszarem obaw, ponieważ kod JavaScript jest podatny na różnego rodzaju ataki, w tym cross-site scripting, który polega na wstrzykiwaniu złośliwego kodu do przeglądarki w celu obejścia zabezpieczeń aplikacji. Dodatkowo, jako język słabo typowany, JavaScript może prowadzić do nieoczekiwanych błędów w czasie kompilacji, co generuje różnorodne problemy programistyczne[[7]](#footnote-7).

W kontekście niniejszego projektu, zdecydowałem się nie korzystać z JavaScriptu, głównie ze względu na wymagania związane z tworzeniem usługi backendowej. C# oferuje tu wystarczające możliwości bez konieczności implementacji dodatkowych frameworków, jak Node.js. To ułatwia proces tworzenia, redukując potrzebę angażowania zasobów systemowych, takich jak pamięć RAM, co z kolei przekłada się na szybsze przetwarzanie danych i poprawia responsywność projektu. Struktura C# dedykowana dla systemów Windows również znacznie usprawnia wydajność, oferując biblioteki specjalnie przystosowane do tego środowiska.

Chociaż JavaScript posiada potężne narzędzia do kompilacji kodu, takie jak Visual Studio Code, platforma Visual Studio 2022 oferuje bardziej zaawansowane narzędzia do debugowania. W Visual Studio 2022 możliwe jest ustawianie punktów przerwania bezpośrednio w trakcie kompilacji, co jest bardziej intuicyjne w porównaniu do konieczności uruchamiania kodu przez przeglądarkę lub zewnętrzne frameworki w przypadku JavaScript.



Zdięcie – Logo programowania w Java Script

źródło: https://www.datocms-assets.com/48401/1627664298-javascript.png?fit=max&fm=webp&w=900

## Platforma programistyczna Visual Studio 2022

Visual Studio 2022 to flagowa platforma programistyczna (IDE) firmy Microsoft, zaprojektowana głównie dla programistów pracujących z technologiami .NET oraz C#. Choć platforma umożliwia również pracę z innymi językami, takimi jak C++, to najnowsza stabilna wersja, 17.9.5[[8]](#footnote-8), szczególnie wspiera środowisko .NET. IDE to zostało opracowane z myślą o zwiększeniu produktywności użytkowników przez zaawansowane funkcje debugowania i ulepszone narzędzia do zarządzania kodem.

Visual Studio 2022 oferuje rozbudowane możliwości zarządzania projektami, w tym efektywne działanie z gałęziami kodu oraz kontrolę zmian, co obejmuje również zarządzanie pull-requestami. Taka funkcjonalność jest kluczowa w nowoczesnym rozwoju oprogramowania, gdzie szybkie i skuteczne zarządzanie zależnościami oraz wersjami jest niezbędne dla utrzymania ciągłości i jakości pracy.

Dodatkowo, Visual Studio 2022 wprowadza rozszerzone wsparcie dla aplikacji 64-bitowych, co pozwala na efektywniejsze wykorzystanie zasobów dostępnych w urządzeniach lub serwerach. To rozszerzenie znacząco poprawia wydajność i możliwości obsługi większych projektów, co jest istotne w przypadku rozwijania skomplikowanych i wymagających aplikacji[[9]](#footnote-9).

Obraz zawierający tekst, zrzut ekranu, Grafika, Czcionka

Opis wygenerowany automatycznieScreen shoot – uruchomienia visual studio 2022

źródło: środowisko

Visual Studio 2022 to zaawansowana platforma programistyczna (IDE) od Microsoft, stworzona głównie dla deweloperów .NET i C#, choć obsługuje również inne języki, takie jak C++. Ta wersja IDE, oznaczona numerem 17.9.5, jest najnowszą i stabilną edycją, która szczególnie skupia się na zwiększeniu produktywności programistów poprzez ulepszone narzędzia debugowania i zarządzania kodem.

Platforma oferuje szerokie możliwości dla programistów, w tym:

* Szybkie tworzenie testów jednostkowych - ułatwia szybką weryfikację funkcjonalności poszczególnych komponentów aplikacji.
* Wdrażanie aplikacji na różnych platformach - wspiera elastyczne rozwiązania cross-platformowe, pozwalając na łatwiejszą adaptację w różnorodnych środowiskach operacyjnych.
* Tworzenie aplikacji mobilnych dla Androida i iOS - umożliwia efektywne projektowanie oprogramowania mobilnego z jednego środowiska.

Jedną z kluczowych cech Visual Studio 2022 jest IntelliCode, system inteligentnego podpowiadania kodu zasilany przez AI. IntelliCode znacząco przyspiesza pisanie kodu, sugerując najbardziej odpowiednie metody i właściwości. Platforma umożliwia również łatwą integrację z pakietami NuGet, co pozwala programistom na implementację dodatkowych bibliotek tworzonych przez zewnętrznych deweloperów, zapewniając kompatybilne i efektywne rozwiązania wspomagające rozwój aplikacji

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Screen shoot – zakładki Tool do Manage NuGet  
 źródło: środowisko

Manager NuGet w Visual Studio 2022 jest podzielony na pięć głównych sekcji:

* Browse – Zakładka ta służy do odnajdywanie dostępnych oraz wspieranych bibliotek zewnętrznych wspieranych przez twórcę Visual Studio 2022 oraz opublikowane przez rzetelnych twórców. Powoduje to systematyczne nowe wersje danych bibliotek dostępnych przy nowszych wersjach podczas udostępniania platformy programistycznej.
* Installed – Druga zakładka służy do kontroli bibliotek zaimplementowanych w projekcie. Rozwiązanie to powoduje informowanie użytkownika o wystąpieniu zmian w projekcie między innymi dostępność nowszej wersji nuget, ostrzeżenie o braku wsparcia dla danej biblioteki oraz stan faktyczny po instalacji nowej biblioteki.
* Updates – Trzecia zakładka służy do odseparowania możliwych bibliotek do aktualizacji od głównego projektu aby skrócić okres wyszukiwania wyznaczonych nugetów. Również dodaje nam ikonkę informującą o możliwości aktualizacji na nowszą wersję.
* Consolidate – Ostatnia zakładka służy do ujednolicenia wersji pakietów nuget wykorzystywanych przy tworzonych projektach lub rozwiązaniach. Jego działanie polega na wykrywaniu różnic pakietowych w tworzonych rozwiązaniach i daje możliwość użytkownikowi poprawy bibliotek do jednej współkompatybilnej oraz ujednoliconej wersji.
* Project Manager – Wyświetlone okno po prawej stronie służy do zarządzania instalacją danych pakietów do danej części projektu. Segmentacja projektu polega na przekazanie kodu do osobnych bibliotek klas, co znacznie strukturyzuje oraz ogranicza instalacje projektów co powoduje mniejsze zapotrzebowanie na ogólną zajmowaną pamięć projektów.

Obraz zawierający tekst, oprogramowanie, Oprogramowanie multimedialne, Oprogramowanie graficzne

Opis wygenerowany automatycznie

Screen Shoot – interfejsu manage nugets

źródło: środowisko

Dodatkowo, Visual Studio 2022 wspiera zarządzanie wersjami przez Git, oferując takie funkcje jak[[10]](#footnote-10):

* Tracking Changes – System pozwalający użytkownik, kontrolowanie zmian poprzez drzewo zdarzeń. Powoduje to znaczne ułatwienie przy pracy z większymi projektami, podczas segmentacji zadań na osobne branche oraz poprzez możliwość zatwierdzania jedynie odpowiednich zmian wraz z możliwością sprawdzenia kodu.
* History Viewing – Funkcja dająca użytkownikowi możliwości podglądu ostatnich zmian projektowych. Jest to znacząca funkcja działania programu ponieważ daje możliwości sprawdzania historii projektu podczas jego rozwoju czy także przebudowy. Dostarcza programiście między innymi informacji o ostatnich zmianach wraz z możliwymi połączeniami inaczej pull-request, co daje pełną swobodę w sprawdzaniu zmian oraz konsultacji z osobą, która tworzyła kod.
* Branching and Merging – Methoda prowadząca do podziały pracy przy danym rozwoju projektu, poprzez utworzenie własnych branchy co daje swobodę programiście przy zleceniu danej rozbudowy. Również daje możliwości scalenia wielu brachy do jednego projektu bez zakłócenia jego struktury oraz działania co powoduje, że dany użytkownik zawsze może pracować na aktualnych projekcie.
* Conflict Resolution – Kluczowa metoda sprawdzająca. Zawarta wewnątrz programu programistycznego kontrolująca poprawność zmian projektowych. Funkcja ta powoduje wystąpienie wyjątku w momencie wysłania zmian do zatwierdzenia do głównego pnia projektowego, daje to programiście informacje o wystąpieniu konfliktu podczas próby scalenia zmian oraz prosi o wybór poprawnych zmian aby zakończyć scalanie projektu.
* Collaboration – System ten daje możliwość współpracy wielu programistów na raz, powoduje to większą możliwość porozumienia się przy bardziej wymagających projektach. Wewnętrzna metoda pozwala dodania wielu różnych użytkowników do jednego projektu aby każda z osób powiązanych miała do niego dostęp, nawet w momencie gdy dane repozytorium jest prywatne.
* Rollbacks – Funkcja programu pozwalająca na przywrócenie projektu do poprzedniej wersji poprzez historie zmian. Najczęściej używane przez osoby zarządzające projektem podczas scalenia projektowego z niechcianymi skutkami, dający możliwość zachowania kontroli wersji.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Screen Shoot – interfejsu Git Changes

źródło: środowisko

Podsumowując, Visual Studio 2022 dostarcza kompleksowe środowisko, które wspiera programistów w efektywnym tworzeniu, testowaniu i zarządzaniu oprogramowaniem na różnych platformach i w różnorodnych językach programowania.

## Platforma programistyczna Visual Studio Code

Visual Studio Code (VS Code) to lekka, ale potężna platforma programistyczna typu open source, rozwijana przez Microsoft. Dzięki swojej wszechstronności i elastyczności, zdobyła ona szerokie uznanie wśród programistów pracujących w różnych językach programowania, w tym w C#, JavaScript, a także wielu innych. Kluczowym aspektem, który wyróżnia VS Code na tle innych środowisk programistycznych, jest jej modułowość oraz możliwość dostosowywania przez liczne rozszerzenia, co pozwala programistom na stworzenie idealnie dopasowanego środowiska pracy[[11]](#footnote-11).

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Screen Shoot – interfejsu Visual Studio Code

źródło: środowisko

Kluczowe cechy i funkcjonalności :

VS Code oferuje wszechstronne wsparcie dla debugowania, inteligentne podpowiedzi kodu (IntelliSense), szybkie przeglądanie kodu, a także wbudowane narzędzia do zarządzania wersjami, które integrują się z popularnymi systemami kontroli wersji, takimi jak Git. To, co jednak czyni VS Code niezwykle popularnym, to jego lekkość i wydajność. W przeciwieństwie do pełnej wersji Visual Studio, VS Code może być uruchamiany na znacznie słabszych maszynach, co czyni go doskonałym wyborem dla programistów korzystających z różnych urządzeń, w tym także z laptopów o ograniczonych zasobach.

Integracja z technologiami .NET i C# :

Dla programistów pracujących z .NET i C#, VS Code jest szczególnie atrakcyjny dzięki możliwości korzystania z oficjalnego rozszerzenia C# od Microsoft, które zapewnia zaawansowane funkcje IntelliSense, debugowanie, a także inne narzędzia specyficzne dla .NET takie jak generowanie kodu, refaktoryzacja, i nawigacja. Dodatkowo, VS Code wspiera projektowanie i debugowanie aplikacji konsolowych, webowych oraz aplikacji mobilnych z wykorzystaniem .NET Core, co sprawia, że jest to wyjątkowo uniwersalne narzędzie[[12]](#footnote-12).

Rozszerzenia i personalizacja :

Jedną z najważniejszych zalet VS Code jest jego system rozszerzeń. Użytkownicy mogą dostosować edytor do własnych potrzeb, instalując rozszerzenia dodające nowe funkcje, tematy, debuggery i inne narzędzia. Dostępność tych dodatków sprawia, że każdy programista może skonfigurować VS Code dokładnie pod kątem własnych preferencji i wymagań projektowych.

Chociaż Visual Studio Code jest niezwykle popularnym i wszechstronnym narzędziem, posiada również pewne wady, które mogą wpływać na doświadczenia niektórych użytkowników:

Zasobożerność przy wielu rozszerzeniach :

Mimo swojej bazowej lekkości, VS Code może stać się znacznie bardziej zasobożerny, gdy użytkownik zainstaluje dużą liczbę rozszerzeń. Każde dodatkowe rozszerzenie, szczególnie te bardziej skomplikowane, może zwiększać obciążenie systemu, co prowadzi do spowolnienia działania edytora, szczególnie na słabszych maszynach. To może być problematyczne w środowiskach, gdzie szybkość odpowiedzi i wydajność są kluczowe.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Screen Shoot – interfejsu Visual Studio Code dział pakietów

źródło: środowisko

Brak niektórych zaawansowanych funkcji IDE :

VS Code, będąc bardziej edytorem kodu niż pełnoprawnym środowiskiem programistycznym (IDE), nie posiada wszystkich zaawansowanych funkcji, które oferują środowiska takie jak Visual Studio. Na przykład, nie ma natywnego wsparcia dla niektórych specyficznych zadań, jak projektowanie graficzne interfejsów użytkownika (GUI), co wymusza poszukiwanie i instalowanie dodatkowych narzędzi i rozszerzeń.

Wymóg samodzielnej konfiguracji :

VS Code wychodzi naprzeciw oczekiwaniom programistów, oferując im możliwość personalizacji i dostosowywania narzędzia do własnych potrzeb. Jednakże, ta elastyczność niesie za sobą potrzebę spędzenia czasu na konfiguracji i dostosowywaniu środowiska, co może być barierą dla mniej doświadczonych użytkowników, którzy preferują "out-of-the-box" rozwiązania.

## Podsumowanie wyboru Visual Studio 2022 jako preferowanej platformy programistycznej

Decyzja o wyborze Visual Studio 2022 jako głównej platformy programistycznej do realizacji zaawansowanego systemu do automatyzacji kontroli płatności opiera się na kilku kluczowych aspektach, które sprawiają, że jest to narzędzie bardziej odpowiednie do tego projektu niż Visual Studio Code.

Zaawansowane możliwości programistyczne :

Visual Studio 2022 oferuje pełnoprawne zintegrowane środowisko programistyczne (IDE), które zapewnia wsparcie dla zaawansowanego debugowania, profilowania, oraz zarządzania aplikacjami wielojęzycznymi. Dostępność tych narzędzi jest kluczowa w projektach o dużej skali i złożoności, takich jak system automatyzacji kontroli płatności, gdzie precyzja i efektywność są niezbędne.

Integracja z narzędziami Microsoft i technologiami .NET :

Visual Studio 2022 jest szczególnie zintegrowane z ekosystemem Microsoft, oferując optymalne wsparcie dla technologii .NET, co jest istotne dla tego projektu. Zapewnia to płynne połączenie z innymi narzędziami i platformami Microsoft, co ułatwia zarządzanie projektami i integrację z istniejącymi systemami przedsiębiorstwa.

Wsparcie dla zarządzania projektami i zespołami :

Visual Studio 2022 umożliwia skuteczne zarządzanie wielkimi projektami i zespołami programistów, oferując funkcje takie jak kontrola wersji, zarządzanie zależnościami oraz analizę kodu. Te narzędzia są niezbędne w złożonych projektach inżynierskich, gdzie współpraca i kontrola jakości odgrywają kluczową rolę.

Optymalizacja i wydajność :

Visual Studio 2022 jest bardziej wydajne w zarządzaniu zasobami i obsłudze dużych projektów. Zawiera funkcje, które pomagają w optymalizacji wydajności aplikacji, co jest kluczowe przy obsłudze rozbudowanych systemów i dużych ilości danych, jakie generuje system do kontroli płatności.

Podsumowanie :

Chociaż Visual Studio Code jest atrakcyjną opcją ze względu na swoją lekkość i modularność, Visual Studio 2022 przewyższa je w obszarach niezbędnych dla tego projektu. Zaawansowane możliwości programistyczne, głęboka integracja z technologiami .NET, lepsze wsparcie dla zarządzania dużymi projektami i zespołami, a także większa efektywność w zarządzaniu zasobami, czynią Visual Studio 2022 idealnym wyborem dla zaprojektowania i wdrożenia zaawansowanego systemu do automatyzacji kontroli płatności. Wybór ten zapewnia, że system będzie rozwijany w stabilnym, wszechstronnym i wydajnym środowisku, co ma kluczowe znaczenie dla sukcesu projektu.

# Cele i założenia projektowe

W drugim rozdziale niniejszej pracy przedstawiony zostanie szczegółowy koncept projektu usługi, mającej na celu automatyzację kontroli płatności dokumentów. Zaprezentowane zostaną zarówno wymagania funkcjonalne, jak i niefunkcjonalne, które definiują oczekiwane działanie i cechy systemu.

Dodatkowo, dokonana zostanie analiza profilu odbiorcy. Rozdział ten szczegółowo zidentyfikuje potencjalnych użytkowników rozwiązania, co pozwoli na lepsze zrozumienie potrzeb i oczekiwań grupy docelowej. Proces projektowania usługi będzie uwzględniał nie tylko wybrane technologie, ale również metodologie i frameworki, które zapewnią skuteczność i efektywność rozwiązania.

Szczególną uwagę poświęci się także zasadom czystego kodu oraz poprawności struktury projektowej. Omówienie tych elementów pomoże w zrozumieniu, jak kluczowe jest utrzymanie wysokich standardów kodowania i organizacji projektu w celu zapewnienia jego skalowalności, utrzymania i możliwości dalszego rozwoju.

Podsumowując, rozdział ten będzie nie tylko prezentacją wymagań i technologii, ale również refleksją nad najlepszymi praktykami w projektowaniu oprogramowania, które są niezbędne do stworzenia niezawodnego i efektywnego systemu do automatyzacji kontroli płatności.

## Cele projektowe

Projekt ten został zainicjowany w odpowiedzi na rosnące zapotrzebowanie w różnych sektorach gospodarki na automatyzację codziennych operacji, takich jak weryfikacja płatności faktur. Rozwój technologii i oprogramowania, a także zwiększone oczekiwania użytkowników dotyczące efektywności pracy, skłaniają do wprowadzania procesów automatyzacyjnych.

Celem projektu jest zredukowanie czasu oraz zasobów ludzkich, które są obecnie poświęcane na ręczne zarządzanie i kontrolę płatności. Dotychczasowe metody, polegające na manualnym informowaniu firm zewnętrznych lub kontrahentów o nieuregulowanych płatnościach, są czasochłonne i narażone na błędy ludzkie. Proponowane rozwiązanie ma na celu nie tylko przyspieszenie procesu weryfikacji faktur, ale także zwiększenie jego dokładności i skuteczności.

Dzięki automatyzacji monitorowania terminów płatności, projekt ten minimalizuje ryzyko błędów ludzkich, co znacząco usprawnia zarządzanie finansami w firmach i korporacjach. Ostatecznym wynikiem będzie lepsza organizacja procesów płatniczych i wydajniejsze zarządzanie zobowiązaniami finansowymi.

## Wymagania funkcjonalne i pozafunkcjonalne

### Wymagania funkcjonalne:

* Pobieranie listy dokumentów
* Pobieranie kontrahentów
* Pobranie wyszczególnionych pól z listy dokumentów
* Pobranie wyszczególnionych pól z listy kontrahentów
* Walidacja danych o czasie płatności
* Metoda sprawdzająca dane i filtrująca odpowiednie wyniki
* Metoda wysyłająca emaila do płatnika
* Stworzenie systemu wyjątków Logger
* Stworzenie panelu konfiguracyjnego
* Stworzenie modelu pobierania danych dokumentów
* Stworzenie modelu pobierania danych kontrahentów
* Stworzenie workera do działania jako usługa
* Metoda dodająca czas uruchomienia usługi
* Metoda wysyłania wiadomości sms
* Przechwytywanie wyjątków
* Stworzenie logiki wewnętrznej oraz implementacji serwisów

### Wymagania pozafunkcjonalne:

* Konfiguracja pod system Windows
* Konfiguracja pod systemy typu Unix
* Test poprawności pobierania dokumentów
* Test poprawności pobierania kontrahentów
* Przesyłanie danych miedzy metodami
* Wykonanie interwału o podanej godzinie
* Informacje konsolowe przez system log
* Testy integracyjne
* Testy jednostkowe
* Testy wdrożeniowe

## Analiza profilu użytkownika

Charakterystyka grup użytkowników:

* Księgowi – Użytkownicy tej grupy często poszukują rozwiązań automatyzujących procesy płatnicze i fakturacje, co zwiększa wygodę oraz efektywność ich pracy w obliczu rosnącego zapotrzebowania. Automatyzacja przypomnień o zaległych płatnościach, integracja z innymi systemami rozrachunkowymi oraz możliwość personalizacji funkcji pod konkretne potrzeby firmy są dla nich kluczowe. To wszystko przyczynia się do zmniejszenia ryzyka błędów i poprawy płynności finansowej organizacji.
* Dostawcy – Dla tej grupy ważne są funkcje ułatwiające nadzorowanie i zarządzanie płatnościami za dostarczone produkty oraz integracja z systemami magazynowymi, takimi jak ERP bądź Soga. Cenią oni sobie możliwość szybkiego dostępu do informacji, które pozwalają na bieżąco monitorować status zamówienia i płatności.

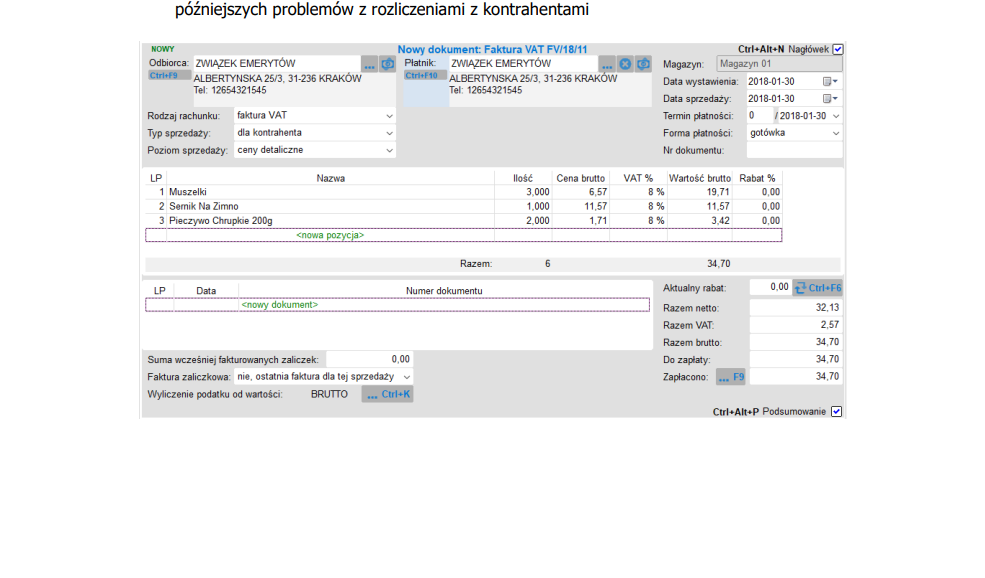
Kompatybilność z różnymi systemami :

System ten oferuje kompatybilność z różnymi platformami ERP, takimi jak ERP XL, ERP Optima, Pc-Market, Soga. Elastyczność ta wynika z modułowej struktury programu, która umożliwia łatwe przepięcie i dostosowanie do różnych środowisk poprzez zmianę bazy danych i odpowiednie konfiguracje pól.

Wybór tych grup użytkowników jako głównych odbiorców projektu podyktowany jest rosnącym zapotrzebowaniem na coraz bardziej zaawansowane i skuteczne rozwiązania automatyzujące. Rynkowa luka w dostarczaniu zmechanizowanych usług przez twórców zewnętrznego oprogramowania stanowi szansę na wprowadzenie nowych, efektywniejszych produktów, zaspokajających te specyficzne potrzeby.

## Analiza projektu i budowy usługi

Podczas projektowania i tworzenia usługi, kluczowe jest dogłębne zrozumienie problemu oraz analiza celów, które mają za zadanie prowadzić do stworzenia skutecznego produktu rozwiązującego konkretne problemy. Jednym z głównych wyzwań było zrozumienie relacji między dokumentami przechowywanymi w bazach danych, na przykładzie systemu Pc-Market, oraz stworzenie interfejsu umożliwiającego efektywne zarządzanie potrzebnymi rekordami.



Screen Shoot oficjalnej dokumentacji – sposób wystawienia od strony interfacu źródło : https://www.insoft.com.pl

Obraz zawierający tekst, numer, Czcionka, linia

Opis wygenerowany automatycznie

Screen Shoot oficjalnej testowej bazy – implementacja pól z bazy danych źródło : <https://www.insoft.com.pl/>

Kluczowym elementem projektu była również decyzja dotycząca wyboru technologii. Na rynku dostępnych jest wiele języków programowania, zarówno wysoko-, jak i nisko-poziomowych, jednak wybór padł na obiektowy język C# z frameworkiem .NET. Wybór ten był motywowany kilkoma czynnikami, takimi jak wysoka szybkość obliczeniowa, prostota implementacji oraz bogata struktura gotowych metod. Dodatkowo, obszerna i szczegółowa dokumentacja dostarczana przez twórców C# ułatwia implementację i skraca czas tworzenia, co przekłada się na redukcję kosztów.

Ostatnim istotnym aspektem projektu była implementacja systemu kontroli błędów, który jest kluczowy dla poprawnego funkcjonowania usługi. System ten oferuje trzy rodzaje komunikatów wyświetlanych w konsoli, co pomaga w identyfikacji i rozróżnieniu błędów:

* Informacyjne – sygnalizują użytkownikowi o uruchomieniu procesu lub o bieżącej aktywności wykonawczej w usłudze.
* Błąd – informują o błędzie występującym podczas przetwarzania danych, z precyzją wskazującą, w której części struktury wystąpił problem.
* Ostrzeżenie – wskazują na potencjalne ryzyko błędu na danym etapie kompilacji, co pozwala na wcześniejsze podjęcie działań korygujących.

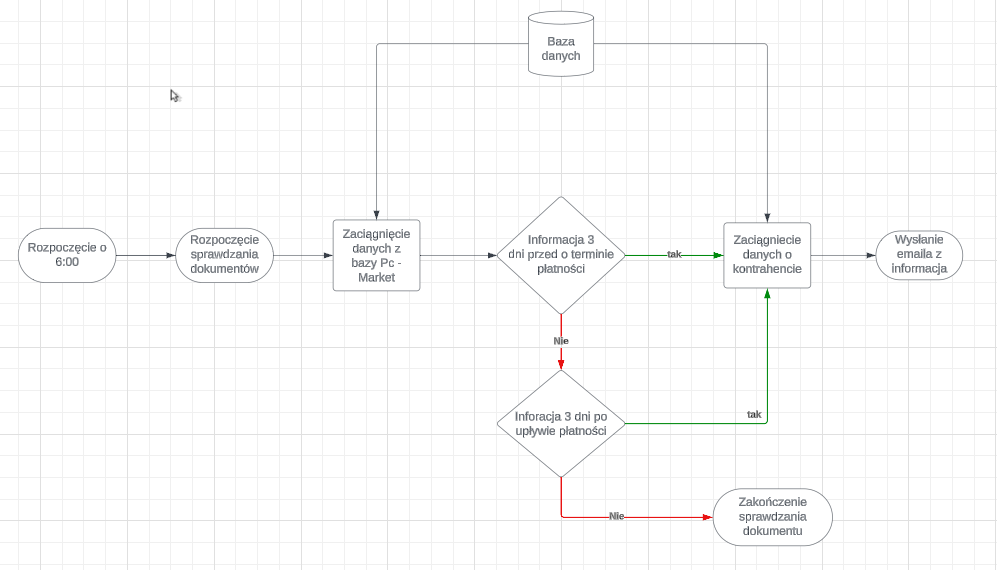
Przeanalizowane podejście do wyboru technologii, projektowanie interfejsu oraz systematyczna kontrola błędów składają się na kompleksową strategię mającą na celu optymalizację procesów wewnętrznych i zapewnienie niezawodności opracowywanej usługi.

## Schemat działania aplikacji – Diagram UML

Usługa jest zaprogramowana do uruchamiania się automatycznie każdego dnia o szóstej rano. Rozpoczęcie działania jest rejestrowane w systemie Logger, informując użytkownika o aktywacji usługi. Na początku działania, usługa pobiera zestaw danych z modułu dokumentów, gdzie analizuje okresy płatności. Jeśli jakiekolwiek zobowiązanie spełnia kryterium płatności (logiczne true), system identyfikuje kontrahenta związanego z danym dokumentem oraz pobiera jego adres email z modułu kontrahentów[[13]](#footnote-13).

Następnie, usługa przesyła te dane do metody odpowiedzialnej za wysyłanie powiadomień o zaległościach w uregulowaniu należności. Po przeprowadzeniu analizy wszystkich faktur z poprzedniego dnia, system informuje o zakończeniu procesu inicjalizacji i przechodzi w tryb czuwania, oczekując na kolejne uruchomienie następnego dnia o tej samej godzinie.

Interwał czasowy, zdefiniowany jako regulator czasowy uruchamiający się codziennie o szóstej rano, jest skutecznym rozwiązaniem zarówno pod kątem oszczędności danych, jak i minimalizacji zużycia procesora. Dzięki temu, użytkownik może regularnie monitorować dokumenty bez ryzyka powtórzenia wysyłki e-maili do osób z zaległościami płatniczymi. Ograniczenie aktywności usługi do jednorazowego działania w ciągu dnia przyczynia się do znacznej oszczędności zasobów.



Screen Shoot modułowego działania usługi – poprawna methodyka użycia źródło : https://lucid.app/

## Schemat analizy bazy danych - Diagramy UML

W ramach projektu, szczególnie ważna jest analiza struktury bazy danych, która umożliwia zrozumienie i optymalizację procesów biznesowych. Baza danych dostarczona przez Pc-Market zawiera złożone relacje i powiązania, które muszą być dokładnie zrozumiane, aby zapewnić efektywne zarządzanie danymi i interakcję z interfejsem użytkownika. Kluczowym wyzwaniem w tej fazie projektu jest identyfikacja kluczowych relacji tabel, które są niezbędne do prawidłowego funkcjonowania aplikacji.

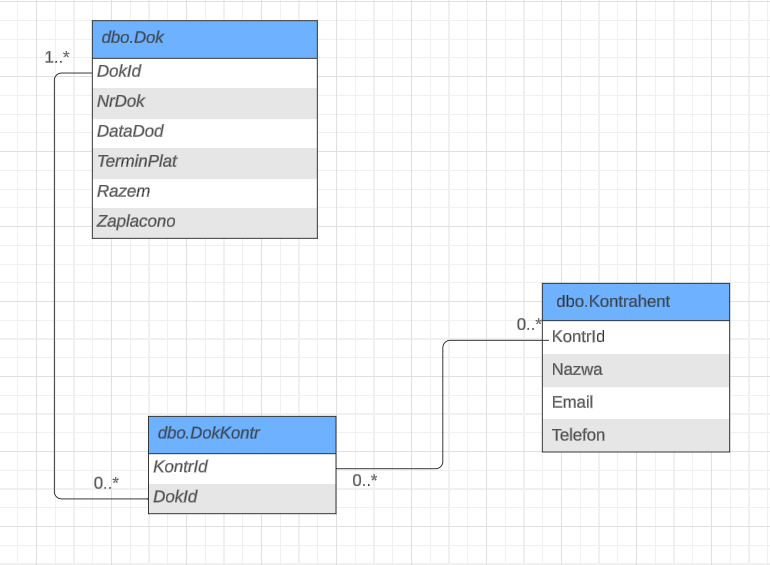
Analiza polega na odnalezieniu i opisaniu, które pola w tabeli odpowiadają za konkretne działania programu, a także na ustaleniu, jak te pola wpływają na użytkownika końcowego. Przykładem może być tabela dokumentów, z której dane są wykorzystywane do monitorowania statusu płatności i generowania alertów dla kontrahentów zalegających z płatnościami[[14]](#footnote-14).

Proces pobierania i przetwarzania danych:

Z bazy danych SQL Server pobierane są dane, które są niezbędne do wykonania określonych funkcji w usłudze. Na przykład, system musi pobrać i przetworzyć informacje o kontrahentach i powiązanych z nimi dokumentach, aby móc automatycznie wysyłać powiadomienia o zaległych płatnościach. Proces ten obejmuje selekcję, agregację oraz filtrację danych zgodnie z ustalonymi kryteriami biznesowymi, takimi jak okres płatności czy status rozliczenia.

Wykorzystanie Lucidchart do wizualizacji danych:

Do wizualizacji skomplikowanych powiązań w bazie danych użyto diagramów UML stworzonych w Lucidchart. Wybór tego narzędzia był motywowany jego intuicyjnym interfejsem użytkownika oraz bogatą funkcjonalnością, dostępną w ramach darmowej licencji. Lucidchart umożliwia projektowanie przejrzystych, łatwo zrozumiałych diagramów UML, które mogą być wykorzystane do prezentacji struktury bazy danych i procesów w niej zachodzących wszystkim zainteresowanym stronom, w tym zespołowi deweloperskiemu i menedżerom projektu[[15]](#footnote-15).



Screen Shoot schematu tabel Pc-Market – Analiza danych  
źródło: lucidchart

Dzięki Lucidchart możliwe jest nie tylko tworzenie diagramów, ale również ich współdzielenie i współpraca nad nimi online, co jest szczególnie wartościowe w projektach, gdzie zespoły pracują zdalnie. Ułatwia to komunikację i przyspiesza procesy decyzyjne, zapewniając wszystkim dostęp do aktualnych wersji dokumentacji projektowej.

## Koncepcja konfiguracji serwisu

Ten podrozdział skupia się na koncepcji instalacji oraz konfiguracji usługi, która opiera się na elastycznym modelu konfiguracji za pomocą zewnętrznego pliku JSON o nazwie appsettings.json. Ta metoda konfiguracji umożliwia użytkownikowi dostosowanie ustawień usługi bez konieczności interweniowania w kod źródłowy, co zapewnia większą elastyczność i bezpieczeństwo aplikacji.

* DatabaseConfig – Jest to główny moduł konfiguracyjny, który pozwala na nawiązanie połączenia z bazą danych. Konfiguracja ta umożliwia aplikacji bezpośredni dostęp do bazy danych, co jest kluczowe dla wydajności i bezpośredniego przetwarzania danych:
* SqlDatabaseServer – Umożliwia użytkownikowi podanie ścieżki do serwera bazy danych, co może być serwerem wewnętrznym lub zewnętrznym.
* SqlDatabaseName – Pozwala na określenie konkretnej bazy danych, z której usługa będzie korzystać do przetwarzania operacji.
* EmailConfig – Ten moduł konfiguracji jest odpowiedzialny za ustawienia związane z wysyłaniem e-maili do klientów:
* Email – Ustawienie to pozwala na konfigurację adresu e-mail, z którego będą wysyłane powiadomienia.
* EmailPassword – Zapewnia bezpieczne przechowywanie hasła do skonfigurowanego konta e-mail.
* SmptSender – Definiuje serwer SMTP, który będzie używany do wysyłania wiadomości e-mail, wspierając integrację z różnymi dostawcami usług pocztowych.
* ActiveWorkerConfig – Ta część konfiguracji odpowiada za zarządzanie cyklem życia usługi:
* Worker – Umożliwia uruchomienie lub zatrzymanie usługi poprzez ustawienie wartości logicznej true/false. Gdy wartość jest ustawiona na true, usługa aktywuje wewnętrzny timer, który uruchamia procesy sprawdzające o określonej godzinie, zazwyczaj o szóstej rano. Zmiana wartości w tym polu wymaga zatrzymania usługi, modyfikacji konfiguracji i ponownego uruchomienia, co może wymagać pewnej znajomości obsługi systemów komputerowych, co jest szczególnie istotne dla mniej zaawansowanych technologicznie użytkowników.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Screen Shoot pliku konfiguracyjnego z projektu Pc-Market -Service – Analiza Json  
źródło: projekt

Znaczenie i korzyści z zastosowania JSON w konfiguracji:

Wybór formatu JSON dla pliku konfiguracyjnego jest podyktowany jego prostotą i uniwersalnością. Format ten jest szeroko stosowany i łatwy do zrozumienia, co ułatwia zarówno deweloperom, jak i użytkownikom końcowym edycję i zarządzanie ustawieniami. JSON pozwala na szybką i efektywną serializację danych, co jest istotne w dynamicznie zmieniających się środowiskach aplikacji. Dodatkowo, łatwość użycia JSON sprzyja integracji z nowoczesnymi technologiami i narzędziami, które wspierają automatyczne procesy wdrożeniowe i utrzymaniowe[[16]](#footnote-16).

## Podział struktury projektu na części

Struktura projektu została zaprojektowana zgodnie z zasadami SOLID, co jest kluczowe dla budowania skalowalnego i łatwo utrzymywalnego oprogramowania w środowisku ASP.NET. Projekt podzielono na osiem modułów, które współpracują ze sobą przy użyciu Dependency Injection, zapewniając luźne powiązania między komponentami i ułatwiając zarządzanie zależnościami[[17]](#footnote-17).

Usługa została podzielona na :

* Configuration – To jeden z najważniejszych modułów projektu, odpowiadający za konfigurację połączenia z bazą danych oraz ustawienia komunikacji emailowej. Moduł umożliwia dynamiczne dostosowanie parametrów bez konieczności modyfikacji kodu źródłowego, co zwiększa elastyczność i bezpieczeństwo aplikacji.
* Email – Moduł odpowiedzialny za obsługę komunikacji emailowej, w tym wysyłanie powiadomień do kontrahentów. Jest kluczowy dla zapewnienia płynności finansowej i kontroli dokumentów, a jego efektywność wpływa bezpośrednio na satysfakcję klientów.
* Helper – Odpowiada za zarządzanie zasobami systemowymi, w tym optymalizację zużycia procesora. Moduł ten jest kluczowy dla efektywnego zarządzania czasem i zasobami, co przekłada się na stabilność i wydajność usługi.
* Logger – Pełni funkcję informacyjną i jest niezbędny do monitorowania działania systemu. Rejestruje różne typy zdarzeń (informacyjne, ostrzeżenia, błędy), co pomaga w diagnozowaniu problemów i utrzymaniu systemu[[18]](#footnote-18).
* Informacyjna – Odpowiada za poinformowanie użytkownika bądź osoby technicznej o etapach działania programu i kontrahentach do których wiadomość została odpowiednio wysłana.
* Ostrzegawcza – Wystąpienie tego czynnika jest, rzadkie ale może poinformować o niebezpieczeństwie podczas wykonywania czynności ze wskazaniem miejsca zagrożonego, lecz nie powoduje to błędu podczas całego procesu.
* Informacja o błędzie – jedna z najważniejszych informacji dla osoby technicznej. Dzięki tej wiadomości szybciej można zlokalizować wstąpiony błąd co znacząco skraca czas analizy błędu oraz próby odzwierciedlenia go na prywatnym środowisku. Skuteczność jego odczytu niestety zależy od doświadczenia osoby posługującej się programem, która będzie potrzebowała pomocy osoby technicznej
* Model – Definiuje strukturę danych używanych w projekcie, co jest fundamentem dla prawidłowego funkcjonowania aplikacji. Jest niezbędny do interakcji z bazą danych i przetwarzania danych.
* Repository – Moduł ten zarządza operacjami na bazie danych, wykonując zapytania SQL i przekształcając dane na modele używane w aplikacji. Jest to kluczowy komponent dla integritas danych i ich spójności.
* Service - To serce aplikacji, gdzie odbywa się przetwarzanie i analiza danych. Moduł ten odpowiada za logiczne operacje, takie jak weryfikacja terminów płatności czy sprawdzanie statusu faktur, a także za wysyłanie powiadomień, gdy płatność jest zaległa.
* Worker – Moduł ten zarządza cyklem życia usługi, decydując o jej uruchomieniu i zatrzymaniu. Odpowiada za inicjalizację usługi, co obejmuje zarządzanie interwałami czasowymi i kontrolę nad aktywnością usługi.

Wpływ na projekt:

Te moduły wspólnie tworzą spójny system, gdzie każda część ma jasno określoną rolę, co minimalizuje zależności i zwiększa modularność projektu. Zastosowanie Dependency Injection pozwala na łatwe zarządzanie zależnościami między modułami, co ułatwia testowanie, rozwój i utrzymanie systemu.

Projekt został zaprojektowany tak, aby był łatwy w konfiguracji dzięki użyciu plików JSON, co pozwala użytkownikom na szybką adaptację i personalizację aplikacji pod indywidualne potrzeby. Ta struktura zapewnia nie tylko lepsze zarządzanie projektami informatycznymi, ale także wpływa na zwiększenie zrozumienia i łatwość obsługi przez użytkownika końcowego.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Screen shoot – układu projektu

źródło: projekt

## Testowanie i Walidacja Systemu

Celem testowania i walidacji systemu jest upewnienie się, że zaimplementowana usługa automatyzacji kontroli płatności spełnia wszystkie wymagania funkcjonalne i niefunkcjonalne ustalone na początku projektu. Proces ten jest kluczowy do zapewnienia nie tylko poprawności działania usługi, ale również jej bezpieczeństwa oraz wyeliminowania potencjalnych błędów, które mogły zostać niezauważone podczas fazy projektowania i implementacji.

### 2.11.1 Strategia testowania

Testowanie systemu jest kluczowym elementem zapewnienia jakości i funkcjonalności usługi. Zawiera kilka krytycznych etapów[[19]](#footnote-19):

* Testy jednostkowe - Skupiają się na pojedynczych modułach lub klasach, testując każdy aspekt izolowany od pozostałej części systemu. Dla projektu ważne jest, aby sprawdzić, czy metody analizujące okresy płatności faktur, identyfikacja kontrahentów, oraz funkcjonalność wysyłania emaili działają poprawnie i efektywnie.
* Testy integracyjne – Przeprowadzane, aby upewnić się, że różne moduły i usługi współpracują ze sobą prawidłowo. Kluczowe jest tutaj przetestowanie integracji między bazą danych a modułami odpowiedzialnymi za przetwarzanie informacji o dokumentach oraz modułami odpowiedzialnymi za komunikację emailową.
* Testy akceptacyjne - (UAT) – Umożliwiają potwierdzenie, że system spełnia wymagania biznesowe i jest gotowy do wdrożenia. Są przeprowadzane w środowisku, które jak najwierniej odwzorowuje realne warunki użytkowania.

### 2.11.2 Walidacja systemu

Walidacja to proces potwierdzający, że finalna wersja usługi spełnia wszystkie założenia projektowe:

* Testy wydajnościowe – Skoncentrowane na ocenie, czy usługa efektywnie zarządza zasobami systemowymi i nie obciąża nadmiernie infrastruktury.
* Testy bezpieczeństwa - Oceniają, czy system jest zabezpieczony przed potencjalnymi zagrożeniami i atakami zewnętrznymi.

### 2.11.3 Proces walidacji

Walidacja powinna również obejmować zbieranie i analizę feedbacku od użytkowników testujących system, co pozwala na wyłapanie i skorygowanie nieoczekiwanych błędów przed ostatecznym wdrożeniem produktu. Jest to istotne dla dostarczenia użytkownikom końcowym produktu, który dokładnie spełnia ich potrzeby i jest wolny od błędów.

## Wdrożenie i Eksploatacja

### Inicjacja wdrożenia systemu wymaga dokładnego planowania i koordynacji:

### 2.12.1 Przygotowanie środowiska produkcyjnego:

Pierwszy krok to przygotowanie środowiska, na którym usługa będzie operować, co obejmuje instalację niezbędnych oprogramowań i konfigurację systemów zewnętrznych jak bazy danych SQL Server.

### 2.12.2 Migracja danych i testy integracyjne:

Przed pełnym uruchomieniem, kluczowe jest przeprowadzenie migracji danych i upewnienie się, że wszystkie systemy są ze sobą prawidłowo zintegrowane.

### 2.12.3 Wdrożenie usługi:

Wdrożenie powinno być starannie zaplanowane, aby zminimalizować przestoje i inne problemy operacyjne.

### 2.12.4 Szkolenie użytkowników i wsparcie techniczne:

Po wdrożeniu usługi, użytkownicy muszą być przeszkoleni, jak korzystać z nowego systemu, a wsparcie techniczne powinno być dostępne, aby pomóc w rozwiązywaniu problemów i odpowiedzieć na pytania.

### 2.12.5 Monitorowanie i utrzymanie:

Stałe monitorowanie działania systemu po wdrożeniu jest niezbędne, aby zapewnić jego ciągłą efektywność i rozwiązywanie napotkanych problemów.

### 2.12.6 Zapewnienie ciągłości działania:

Należy rozwijać procedury, które zapewnią działanie systemu nawet w przypadku awarii, włączając w to strategie tworzenia kopii zapasowych i plany awaryjne.

## Podsumowanie rozdziału drugiego

Rozdział drugi projektu skupiał się na szczegółowej analizie i konstrukcji usługi automatyzacji kontroli płatności. Począwszy od definiowania celów i założeń projektowych, przez analizę profilu użytkownika, po szczegółowe opisanie struktury i konfiguracji systemu, rozdział ten dostarcza kompleksowego spojrzenia na techniczne i funkcjonalne aspekty projektu.

Cele i założenia projektowe ujawniły kluczowe wymagania funkcjonalne i niefunkcjonalne, które kierowały projektowaniem systemu, z naciskiem na automatyzację procesów płatniczych w celu poprawy efektywności i redukcji błędów.

Analiza profilu użytkownika pozwoliła zrozumieć potrzeby i oczekiwania końcowych użytkowników, co umożliwiło bardziej celowe projektowanie funkcji systemu, takich jak integracja z systemami księgowymi i możliwości konfiguracji zgodnie z indywidualnymi potrzebami organizacji.

Strategia budowy usługi rozważała techniczne aspekty implementacji, w tym wybór technologii, zastosowanie wzorców projektowych oraz kierowanie się zasadami SOLID w celu zapewnienia czytelności i łatwości w utrzymaniu kodu.

Testowanie i walidacja systemu odgrywały kluczową rolę w zapewnieniu, że usługa działa zgodnie z oczekiwaniami i jest wolna od błędów przed jej wdrożeniem. Proces ten obejmował testy jednostkowe, integracyjne oraz akceptacyjne, a także walidację systemu poprzez testy wydajnościowe i bezpieczeństwa.

Wdrożenie i eksploatacja stanowiły końcowy etap projektu, gdzie szczegółowo omówiono proces przygotowania środowiska produkcyjnego, migracji danych, testów integracyjnych, a także szkolenia użytkowników i wsparcie techniczne. Dzięki temu zapewniono, że usługa będzie skutecznie funkcjonować w realnym środowisku operacyjnym, oferując stabilne i bezpieczne narzędzie do zarządzania płatnościami.

Podsumowując, rozdział drugi dostarcza kompleksowego przewodnika po każdym aspekcie rozwoju systemu, od początkowych koncepcji po finalne wdrożenie, zapewniając, że wszystkie elementy są starannie zaplanowane i wykonane z myślą o ostatecznym celu - stworzeniu niezawodnej i efektywnej usługi automatyzacji kontroli płatności.

# Proces implementacji i dokumentacja techniczna

Niniejszy rozdział poświęcony jest omówieniu aspektów technicznych związanych z realizacją projektu usługi automatyzacji kontroli płatności. Skupia się na przedstawieniu metod i procedur niezbędnych do zrozumienia, jak funkcjonuje system wyjątków, z implementacją metodyki logger, oraz jakie są mechanizmy obsługi komunikatów informacyjnych. Szczegółowo opisany zostanie proces instalacji usługi na środowisku produkcyjnym, wraz z niezbędnymi procedurami i komendami wymaganymi do prawidłowego wdrożenia u klienta.

Dodatkowo, rozdział zawiera dogłębną analizę metod zaciągania danych z różnych baz danych, które dostarczają te same wyniki, zilustrowane na konkretnych przypadkach użycia. Zostanie również przedstawiona metodologia realizacji zadań logicznych, w tym systemy sprawdzania zaległości poprzez stosowanie odpowiednich funkcji oraz technik mapowania danych na oczekiwane wartości.

Czytelnik zostanie zapoznany z procesem przekazywania danych i symulacji poprawnego ustawienia, na przykładzie wysyłania e-maila, co stanowi elementarny komponent komunikacji w usłudze. Ponadto, szczegółowo przeanalizowane zostaną kluczowe mechanizmy serca usługi, tj. system interwału odpowiedzialnego za jej regularne przebudzenie i utrzymanie ciągłości działania.

Ten rozdział ma na celu nie tylko zaprezentowanie praktycznych aspektów implementacji, ale także zwiększenie świadomości czytelnika na temat skomplikowanych procesów technologicznych leżących u podstaw działania nowoczesnych systemów informatycznych w kontekście usług automatyzacji.

## Konfiguracja środowiska pod usługę oraz instalacja

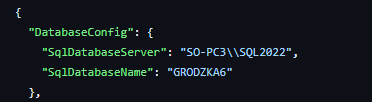
Instalacja usługi na systemie Windows wymaga szczegółowego przygotowania i konfiguracji środowiska operacyjnego, co gwarantuje stabilność i bezpieczeństwo działania aplikacji. Proces instalacji można podzielić na następujące kroki:

## 3.1.1 Krok 1 - Przygotowanie systemu :

* **Instalacja .NET 8 SDK** - Aby zainstalować najnowsze środowisko .NET 8, należy pobrać i zainstalować .NET 8 SDK ze strony Microsoft. Można to zrobić ręcznie lub za pomocą wiersza poleceń:
* Start-Process -FilePath "C:\Path\dotnet-sdk-8.0-win-x64.exe" -ArgumentList "/install /quiet /norestart" -NoNewWindow -Wait

## 3.1.2 Krok 2 - Ustawienie serwera baz danych :

* **Instalacja SQL Server** - Instalacja SQL Server może być przeprowadzona na kilka sposobów w zależności od potrzeb środowiska. Możliwości instalacji obejmują zarówno tradycyjną instalację z instalatora pobranego ze strony Microsoft, jak i wykorzystanie kontenerów Docker dla izolacji środowiska oraz łatwości w zarządzaniu wersjami i zależnościami.
* **Konfiguracja połączenia z bazą danych** - Po zainstalowaniu SQL Server, konfiguracja połączenia z bazą danych wymaga odpowiedniego ustawienia w pliku appsettings.json aplikacji .NET. Ten plik konfiguracyjny przechowuje parametry połączenia, które pozwalają aplikacji na komunikację z bazą danych. Przykład wpisu w pliku appsettings.json:



Screen shoot – Konfiguracja bazy danych

źródło: projekt

* W tym przypadku, Server zawiera nazwę serwera oraz instancję SQL Server, a Database określa nazwę bazy danych, do której aplikacja będzie się łączyć.
* **Server** - nazwa serwera lub adres IP, gdzie hostowana jest instancja SQL Server, w tym przypadku lokalny serwer SO-PC3\SQL2022, gdzie SQL2022 to nazwa instancji SQL Server[[20]](#footnote-20).
* **Database** - nazwa bazy danych, z którą aplikacja powinna się łączyć, tutaj GRODZKA6.

### Krok 3 - Instalacja i konfiguracja usługi :

* **Publikacja aplikacji .NET** - Publikacja aplikacji może być wykonana lokalnie z użyciem polecenia dotnet publish bądź interfejsu platformy Visual Studio 2022 :

Obraz zawierający zrzut ekranu, Czcionka, Grafika, tekst

Opis wygenerowany automatycznie

Screen shoot – konsola na środowisku

źródło: środowisko testow

* **Rejestracja usługi w Windows** - Użyj narzędzia sc.exe do zarejestrowania aplikacji jako usługi systemowej:

Obraz zawierający zrzut ekranu, tekst, Czcionka, linia

Opis wygenerowany automatycznie

Screen shoot – konsola na środowisku

źródło: środowisko testow

### Krok 4 - Testowanie i weryfikacja :

* **Testowanie funkcjonalne** - Wykonaj testy, aby upewnić się, że aplikacja działa prawidłowo i spełnia wszystkie wymogi funkcjonalne.

## Metodyka zaciągania danych

Proces zaciągania danych w aplikacjach .NET korzystających z SQL Server jako bazy danych odbywa się za pomocą kilku kluczowych komponentów ADO.NET, które zapewniają efektywne zarządzanie połączeniami, komendami i transakcjami danych. Poniżej przedstawiono dwie podstawowe metody zaciągania danych, które są wykorzystywane w omawianej aplikacji: pobieranie listy dokumentów oraz danych klienta[[21]](#footnote-21).

* **Pobieranie listy dokumentów** - Metoda GetDocumentList() ilustruje proces pobierania listy dokumentów z bazy danych. Używa ona obiektu SqlConnection do nawiązania połączenia z bazą danych, a następnie SqlCommand do wykonania zapytania SQL. Dane są zwracane za pomocą SqlDataAdapter, który wypełnia obiekt DataTable danymi z bazy[[22]](#footnote-22).

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Screen shoot – Kod odczytu dokumentów

źródło: projekt

* **Pobieranie danych klienta** - Metoda GetCustomer(int customerId) demonstruje, jak można zaciągnąć szczegółowe informacje o kliencie na podstawie jego identyfikatora. Proces jest podobny do pobierania listy dokumentów, lecz zawiera dodatkowy krok związany z parametryzacją zapytania SQL[[23]](#footnote-23).

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Screen shoot – Kod źródłowy odczytu danych o kontrahencie

źródło: projket

* **Obsługa błędów -** W obu metodach obecne jest obsługiwanie wyjątków typu SqlException, które mogą wystąpić podczas interakcji z bazą danych. Wyjątki są logowane jako błędy, co pozwala na szybką diagnozę i reakcję w przypadku problemów z połączeniem lub wykonaniem zapytania[[24]](#footnote-24).

Obraz zawierający zrzut ekranu

Opis wygenerowany automatycznie

Screen shoot – Kod źródłowy łapania wyjątków od strony SQL Server

źródło: środowisko projekt

* **Najlepsze Praktyki :** 
  + Używanie parametryzowanych zapytań dla zwiększenia bezpieczeństwa.
  + Otwieranie połączenia tuż przed wykonaniem zapytania i zamykanie go jak najszybciej, aby zasoby były efektywnie zarządzane.
  + Logowanie działania aplikacji w celu monitorowania i diagnozowania problemów.

## Metodyka obróbki danych oraz mapowania

W procesie obróbki i mapowania danych, aplikacja przekształca surowe dane z bazy danych na struktury danych, które są łatwiejsze do zarządzania i wykorzystywane w różnych operacjach biznesowych. Podstawowym elementem tego procesu jest mapowanie danych z obiektów DataRow na obiekty typu DocumentDto i CustomerDto. Poniżej przedstawiono szczegółowy opis tego procesu na przykładzie funkcji MapQueryDocumentResult().

* **Pobieranie Danych** - Metoda GetDocumentList() z repozytorium jest używana do pobierania danych z bazy danych. Zwraca ona DataTable, który zawiera wiersze danych surowych, które następnie są przetwarzane[[25]](#footnote-25).
* **Iteracja po danych -** Przez iterację po każdym wierszu DataRow w DataTable, dane są przekształcane do obiektów DTO. Każda kolumna w DataRow odpowiada właściwościom w DocumentDto[[26]](#footnote-26).

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Screen shoot – Kod źródłowy mappowaia danych

źródło: środowisko projekt

* **Mapowanie danych -** Dla każdego wiersza tworzony jest nowy obiekt DocumentDto i uzupełniany wartościami z odpowiednich kolumn. Oto szczegóły mapowania dla kluczowych pól:
  + **DokumentId -** Konwersja identyfikatora dokumentu.
  + **NazwaDokumentu -** Pobranie i przypisanie numeru dokumentu.
  + **DataWystawieniaDokumentu -** Konwersja daty dodania, z odpowiednim obsługiwaniem wartości DBNull.
  + **TerminPlatnosci, DoZaplaty, Uregulowano -** Konwersja wartości finansowych.
  + **KontrahentId -** Konwersja identyfikatora kontrahenta z odpowiednim sprawdzeniem DBNull.

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Screen shoot – Kod źródłowy metody filtrującej dane

oraz algorytm wybierający dłużników

źródło: środowisko projekt

## System przebudzający usługę

System przebudzający usługę w tej aplikacji jest kluczowym mechanizmem umożliwiającym cykliczne wykonywanie zadań w tle na serwerze. Poniżej szczegółowo opisano implementację poszczególnych metod, które składają się na ten system.

* **Metoda StartAsync –** Metoda StartAsync jest wywoływana przy uruchomieniu usługi. Odpowiada za inicjalizację i planowanie pierwszego uruchomienia zadania w tle. Wykorzystuje do tego metodę pomocniczą ScheduleNextRun, która jest odpowiedzialna za ustalenie czasu następnego uruchomienia.

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Screen shoot – Kod źródłowy metody uruchamiającej interwał

źródło: środowisko projekt

* **Metoda ScheduleNextRun –** Metoda ScheduleNextRun planuje następne wykonanie zadania. Oblicza ona czas do następnego uruchomienia, ustawiając timer, który po upłynięciu określonego czasu wywoła metodę DoWork. Dzięki temu usługa działa cyklicznie, wykonując zadanie co określony interwał czasowy.

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Screen shoot – Kod źródłowy metody sprawdzający czas globalny

źródło: środowisko projekt

* **Metoda DoWork –** DoWork jest metodą wywoływaną przez timer i zawiera logikę, która powinna być wykonana przy każdym uruchomieniu usługi. Po zakończeniu zadania, metoda ponownie wywołuje ScheduleNextRun w celu zaplanowania kolejnego uruchomienia.

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Screen shoot – Kod źródłowy metody uruchamiający całą usługę

źródło: środowisko projekt

* **Metoda StopAsync –** Metoda StopAsync jest wywoływana podczas zatrzymywania usługi. Odpowiada za anulowanie działania timera, co zapobiega kolejnym wykonaniom metody DoWork.

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Screen shoot – Kod źródłowy metody ustawiającej usługę na oczekiwanie

źródło: środowisko projekt

* **Znaczenie systemu –** System przebudzający usługę jest istotnym elementem aplikacji, umożliwiającym automatyczne i regularne wykonanie zadań w tle. Dzięki temu można minimalizować interwencję manualną i zapewniać, że kluczowe zadania są wykonywane na bieżąco. Ta automatyzacja jest szczególnie ważna w aplikacjach, które wymagają regularnego przetwarzania danych lub monitorowania stanu systemu.

# Wnioski końcowe

Realizacja projektu inżynierskiego dotyczącego tworzenia i wdrażania usługi windowsowej do analizy zagadnień teoretycznych oraz praktycznych implementacji technologii backendowych dostarczyła istotnych wniosków, które mają znaczący wpływ na zrozumienie i rozwój aplikacji systemowych:

* Automatyzacja poprawia efektywność: Projekt potwierdził, że automatyzacja procesów biznesowych znacząco zwiększa wydajność operacyjną, redukując błędy ludzkie i przyspieszając przetwarzanie danych.
* Wybór technologii ma znaczenie: Analiza technologii backendowych takich jak C#, Rust, i JavaScript wykazała, że każda z nich ma specyficzne zalety, które wpływają na wydajność, bezpieczeństwo i łatwość utrzymania aplikacji. C# i .NET Framework okazały się być szczególnie odpowiednie dla aplikacji systemowych na platformie Windows ze względu na ich integrację z systemem operacyjnym i wsparcie od Microsoft.
* Kluczowe jest efektywne zarządzanie danymi: Metody zaciągania i przetwarzania danych muszą być efektywne, aby zapewnić aktualność i dokładność informacji, co jest krytyczne dla decyzji biznesowych opartych na danych.
* Niezbędne jest skuteczne zarządzanie błędami: Systematyczne logowanie błędów i ich odpowiednie zarządzanie zwiększa niezawodność i ułatwia utrzymanie systemu.
* Testowanie jest fundamentem: Regularne testowanie komponentów systemu jest niezbędne do zapewnienia ich niezawodności i efektywności, co potwierdza znaczenie testów jednostkowych, integracyjnych i systemowych w procesie rozwoju oprogramowania.

# Podsumowanie

Praca nad projektem inżynierskim umożliwiła głębsze zrozumienie procesów projektowania, implementacji i utrzymania usługi windowsowej. Demonstruje to, jak istotne jest dogłębne rozumienie wybranych technologii i metodologii w kontekście rozwoju oprogramowania systemowego. Projekt nie tylko zwiększył moje kompetencje techniczne, ale także umiejętność adaptacji teoretycznej wiedzy do praktycznych zastosowań w rzeczywistych scenariuszach biznesowych. Dalsze badania mogłyby skupić się na rozszerzeniu funkcjonalności usługi o zaawansowane techniki analizy danych i uczenia maszynowego, co mogłoby jeszcze bardziej zautomatyzować i optymalizować przepływy pracy i procesy decyzyjne w organizacjach.

# Bibliografia

1. „C# 7.0 Kompletny przewodnik dla praktyków” Mark Michaelis, Eric Lippert,

Przedmowa Mads Torgersen

1. „C# 8.0 Kompletny przewodnik dla praktyków” Mark Michaelis, Eric Lippert,

Przedmowa Kevin Bost

1. „C# 9.0 w pigułce” Joseph Albahari

**Źródła internetowe**

1. <https://boringowl.io/blog/piec-zasad-solid-w-programowaniu-obiektowym>
2. <https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history>
3. <https://doc.rust-lang.org/book/>
4. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
5. <https://sekurak.pl/kilka-slow-o-podatnosci-xss-oraz-polyglot-xss/>
6. <https://learn.microsoft.com/pl-pl/visualstudio/releases/2022/compatibility>
7. <https://learn.microsoft.com/pl-pl/visualstudio/ide/whats-new-visual-studio-docs>
8. <https://code.visualstudio.com/docs>
9. <https://www.lucidchart.com/pages/pl/czym-jest-uml-unified-modeling-language>

https://www.json.org/json-pl.html

1. <https://boringowl.io/blog/piec-zasad-solid-w-programowaniu-obiektowym>
2. <https://learn.microsoft.com/pl-pl/dotnet/api/microsoft.entityframeworkcore.diagnostics.relationalloggerextensions.ambienttransactionwarning?view=efcore-8.0>
3. <https://learn.microsoft.com/pl-pl/dotnet/architecture/microservices/multi-container-microservice-net-applications/test-aspnet-core-services-web-apps>
4. <https://learn.microsoft.com/pl-pl/troubleshoot/developer/visualstudio/csharp/language-compilers/create-sql-server-database-programmatically>

<https://learn.microsoft.com/pl-pl/dotnet/framework/data/adonet/ado-net-overview>

<https://learn.microsoft.com/en-us/shows/on-net/how-do-i-use-sql-server-with-csharp-and-dotnet>

<https://learn.microsoft.com/en-us/shows/on-net/how-do-i-use-sql-server-with-csharp-and-dotnet>

<https://learn.microsoft.com/pl-pl/dotnet/api/system.data.sqlclient.sqlexception?view=dotnet-plat-ext->

<https://learn.microsoft.com/en-us/dotnet/api/system.data.datatable?view=net-8.0>

<https://learn.microsoft.com/pl-pl/dotnet/api/system.data.datarow?view=net-7.0>

**Spis rysunków**

Obraz 1. Zakładki branch do Manage github 7

Obraz 2. Logo programowania w Java Script 10

Obraz 3. Uruchomienia visual studio 2022 11

Obraz 4. Zakładki Tool do Manage NuGet 12

Obraz 5. Interfejsu manage nugets 13

Obraz 6. Interfejsu Git Changes 14

Obraz 7. Interfejsu Visual Studio Code 15

Obraz 8. Interfejsu Visual Studio Code dział pakietów 17

Obraz 9. Sposób wystawienia od strony interfacu 22

Obraz 10. Implementacja pól z bazy danych 22

Obraz 11. Poprawna methodyka użycia 24

Obraz 12. Schematu tabel Pc-Market – Analiza 25

Obraz 13. Pliku konfiguracyjnego z projektu Pc-Market -Service – Analiza Json 27

Obraz 14. Układu projektu 29

Obraz 15. Konfiguracja bazy danych … 34

Obraz 16. Konsola na środowisku … 35

Obraz 17. Konsola na środowisku … 35

Obraz 18. Kod odczytu dokumentów … 36

Obraz 19. Kod źródłowy odczytu danych o kontrahencie … 37

Obraz 20. Kod źródłowy łapania wyjątków od strony SQL Server… 37

Obraz 21. Kod źródłowy mappowaia danych … 38

Obraz 22.Kod źródłowy metody filtrującej dane oraz algorytm wybierający dłużników… 39

Obraz 23. Kod źródłowy metody uruchamiającej interwał … 40

Obraz 24. Kod źródłowy metody sprawdzający czas globalny … 40

Obraz 25. Kod źródłowy metody uruchamiający całą usługę … 41

Obraz 26. Kod źródłowy metody ustawiającej usługę na oczekiwanie… 41

1. https://boringowl.io/blog/piec-zasad-solid-w-programowaniu-obiektowym [↑](#footnote-ref-1)
2. https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history [↑](#footnote-ref-2)
3. https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history [↑](#footnote-ref-3)
4. https://doc.rust-lang.org/book/ [↑](#footnote-ref-4)
5. https://doc.rust-lang.org/book/ [↑](#footnote-ref-5)
6. https://developer.mozilla.org/en-US/docs/Web/JavaScript [↑](#footnote-ref-6)
7. https://sekurak.pl/kilka-slow-o-podatnosci-xss-oraz-polyglot-xss/ [↑](#footnote-ref-7)
8. https://learn.microsoft.com/pl-pl/visualstudio/releases/2022/compatibility [↑](#footnote-ref-8)
9. https://learn.microsoft.com/pl-pl/visualstudio/releases/2022/compatibility [↑](#footnote-ref-9)
10. https://learn.microsoft.com/pl-pl/visualstudio/ide/whats-new-visual-studio-docs [↑](#footnote-ref-10)
11. https://code.visualstudio.com/docs [↑](#footnote-ref-11)
12. https://code.visualstudio.com/docs [↑](#footnote-ref-12)
13. https://www.lucidchart.com/pages/pl/czym-jest-uml-unified-modeling-language [↑](#footnote-ref-13)
14. https://www.lucidchart.com/pages/pl/czym-jest-uml-unified-modeling-language [↑](#footnote-ref-14)
15. https://www.lucidchart.com/pages/pl/czym-jest-uml-unified-modeling-language [↑](#footnote-ref-15)
16. https://www.json.org/json-pl.html [↑](#footnote-ref-16)
17. https://boringowl.io/blog/piec-zasad-solid-w-programowaniu-obiektowym [↑](#footnote-ref-17)
18. https://learn.microsoft.com/pl-pl/dotnet/api/microsoft.entityframeworkcore.diagnostics.relationalloggerextensions.ambienttransactionwarning?view=efcore-8.0 [↑](#footnote-ref-18)
19. https://learn.microsoft.com/pl-pl/dotnet/architecture/microservices/multi-container-microservice-net-applications/test-aspnet-core-services-web-apps [↑](#footnote-ref-19)
20. https://learn.microsoft.com/pl-pl/troubleshoot/developer/visualstudio/csharp/language-compilers/create-sql-server-database-programmatically [↑](#footnote-ref-20)
21. https://learn.microsoft.com/pl-pl/dotnet/framework/data/adonet/ado-net-overview [↑](#footnote-ref-21)
22. https://learn.microsoft.com/en-us/shows/on-net/how-do-i-use-sql-server-with-csharp-and-dotnet [↑](#footnote-ref-22)
23. https://learn.microsoft.com/en-us/shows/on-net/how-do-i-use-sql-server-with-csharp-and-dotnet [↑](#footnote-ref-23)
24. https://learn.microsoft.com/pl-pl/dotnet/api/system.data.sqlclient.sqlexception?view=dotnet-plat-ext-8.0 [↑](#footnote-ref-24)
25. https://learn.microsoft.com/en-us/dotnet/api/system.data.datatable?view=net-8.0 [↑](#footnote-ref-25)
26. https://learn.microsoft.com/pl-pl/dotnet/api/system.data.datarow?view=net-7.0 [↑](#footnote-ref-26)