



POLITECHNIKA ŁÓDZKA



**Wydział Elektrotechniki, Elektroniki, Informatyki
i Automatyki**

WCEiA



Instytut Mechatroniki i Systemów Informatycznych

Praca dyplomowa inżynierska

na temat:

**Aplikacja do zarządzania sprzętem i programowaniem
firmowym zrealizowana na platformie .NET.**

**(Application for business hardware and software management
implemented using the .NET framework.)**

Imię i Nazwisko: **Jakub Kowalczyk**
Nr albumu: **215774**
Specjalność: **Technologie internetowe**
Kierunek: **Informatyka**

Opiekun pracy:
dr inż. **Henryk Welfle**

Łódź luty 2022

SPIS TREŚCI

1.	Wstęp.....	5
2.	Cel i zakres pracy.	7
3.	Technologie wykorzystane przy projektowaniu aplikacji.....	9
3.1.	.NET Framework.....	9
3.1.1.	ADO.NET	9
3.1.2.	.NET WPF	10
3.2.	Microsoft SQL Server	10
3.3.	Język programowania C#.....	11
4.	Narzędzia wykorzystane przy projektowaniu aplikacji.....	12
4.1.	Microsoft Visual Studio 2019	12
4.2.	SQL Server Management Studio	12
5.	Opis bazy danych	13
5.1.	Struktura tabel i ograniczenia.....	17
5.1.1.	Tabela Rentals.....	17
5.1.2.	Tabela Products.....	18
5.1.3.	Tabela Product_copies	20
5.1.4.	Tabela Requests	21
5.1.5.	Tabela Contacts.....	22
6.	Opis aplikacji.....	24
6.1.	Część wspólna.....	24
6.1.1.	Menu logowania	25
6.1.2.	Połączenie z bazą danych	26
6.1.3.	Sesja.....	26
6.1.4.	Plik App.xaml	27

6.2.	Część użytkownika	27
6.2.1.	Wypożyczone produkty.....	28
6.2.2.	Wysyłanie prośby o produkt	31
6.2.3.	Wysłane prośby o produkty.....	33
6.2.4.	System przypominania o terminie zwrotu.....	33
6.3.	Część administratora.....	35
6.3.1.	Dodawanie nowych użytkowników lub producentów do systemu	35
6.3.2.	Dodanie nowych produktów do systemu	37
6.3.3.	Edytowanie istniejących użytkowników lub producentów w systemie	38
6.3.4.	Edytowanie istniejących produktów w systemie	41
6.3.5.	Wypożyczanie produktu.....	41
6.3.6.	Prośby o wypożyczenie produktu	42
6.3.7.	Zwrot produktu.....	44
6.3.8.	Statystyki.....	45
6.3.9.	Proces powiadomień mailowych.....	48
7.	Podsumowanie	50
	Streszczenie	52
	Summary	52
	Literatura	53
	Spis rysunków	54
	Spis tabel	56
	Spis listingów	57

1. Wstęp

W XXI wieku, kiedy w życie wchodzi ogólnoswiatowa globalizacja, wiele firm nie ogranicza się do posiadania swojej siedziby w jednym, ustalonym miejscu na świecie, a posiada liczne oddziały nierzadko porozrzucane po kontynencie, a czasem i na całym globie. Dodając do tego pracowników pracujących to fizycznie z danego biura jak i zdalnie z niekiedy innego kraju należy przyłożyć szczególną uwagę na organizację i prawidłowe zarządzanie każdym elementem wchodzącym w część tej układanki zwanej firmą, aby wszystko mogło chodzić tak jak w przysłowiowym zegarku. Niestety, niektórzy w ramach cięć kosztów mogą przymykać oko jak i po prostu zaniedbywać pewne sprawy organizacyjne co może prowadzić do narastania kolejnych problemów w momentach, w których może nie być czasu by sobie z nimi poradzić.

Tak jak w informatyce istnieje zjawisko długu technologicznej spowodowanego zaniedbaniem przez programistów na tle spójności kodu, użytych rozwiązań jak i dokumentacji, w zarządzaniu firmą i pracownikami może dość do podobnego długu, wszakże innego rodzaju jakim jest zarządzanie sprzętem oraz oprogramowaniem używanym przez pracowników w ramach wykonywania zadanych obowiązków.

Aby wielooddziałowa firma mogła funkcjonować sprawnie, musi ona dokładnie znać jaki asortyment posiada, z którego ona jak i jej pracownicy mogą korzystać. Często do pracy potrzebne są wyspecjalizowane programy jak i sprzęt, dzięki którym dany oddział może wykonać powierzoną im rolę jednak w momencie, kiedy nastąpi brak któryś z wyżej wymienionych elementów, może dość do opóźnienia albo sparaliżowania całej firmy, a niekiedy i całych rynków. Kontrolowanie ilości komputerów na stanie, z których pracownicy mogą skorzystać, kończące się licencje oprogramowania potrzebne do codziennej pracy czy po prostu sprzęt, który przestał działać i wymaga naprawy, kontrolowanie tych rzeczy nie jest może jedynym zagadnieniem zarządzania firmą, ale na pewno stanowi ważny jej element, którego nie można przeoczyć.

Jednak pamiętanie o każdej dacie gwarancji każdej poszczególnej klawiatury w firmie jest nie tylko trudnym zadaniem, ale i równie nieoptymalnym, dlatego wykorzystanie oprogramowania wspomagającego cały ten proces powinno być naturalnym zabiegiem dla każdej większej firmy, by nie powstawały problemy już na najniższym szczeblu organizacyjnym jakim są sprawy wewnątrz firmy i właśnie

w ramach tej pracy inżynierskiej stworzona została aplikacja mająca pomóc zoptymalizować ten proces.

2. Cel i zakres pracy.

Celem pracy było wykonanie aplikacji desktopowej pomagającej w zarządzaniu wypożyczania sprzętu firmowego pracownikom jak i udzielaniu licencji do programów niezbędnych do wykonywania obowiązków. Aplikacja pozwala na kontrolowanie stanu jak i ilości sprzętu na stanie firmy, dodawania nowych produktów czy też aktualizowania tych już posiadanych. Posiada ona podział na zwykłego użytkownika jakim domyślnie jest pracownik firmy oraz na administratora zarządzającego wypożyczanym sprzętem.

Użytkownik ma możliwość sprawdzenia jakie produkty już wypożyczył, informacje o takich datach jak wygaśnięcie licencji, koniec gwarancji oraz data, do której dany sprzęt wstępnie jest mu wypożyczony z możliwością przedłużenia pod pewnymi warunkami. Może on również wysyłać prośby o wypożyczenie sprzętu, który firma posiada.

Administrator posiada pełną kontrolę nad wprowadzaniem czy to nowego sprzętu czy też samych użytkowników tego systemu. Może on ręcznie wprowadzić do systemu wypożyczenie określonego sprzętu wybranemu użytkownikowi na ustalony przez niego okres, przyjmować lub odrzucać prośby o produkty od użytkowników czy też odznaczać sprzęt zwrócony, aby inni pracownicy również mogli go wypożyczyć. Zaimplementowany również jest system zbierający statystyki m.in. wypożyczenia, prośby czy też czas wypożyczenia sprzętu.

Aplikacja ma również system przypominający, który wysyła e-mail pracownikowi, jeżeli temu zbliża się termin (tydzień przed) oddania produktu, na jego własną skrzynkę odbiorczą wprowadzoną w systemie w ramach kontaktu.

Oprogramowanie ma na celu jedynie pomoc w organizacji i zarządzaniu sprzętem oraz oprogramowaniem, samo jego wypożyczenie i wręczenie fizyczne pracownikowi leży już po stronie osoby będącej administratorem aniżeli samego programu.

Praca podzielona jest na część teoretyczną, w której omówione jest zaplecze aplikacji takie jak użyta technologia czy też oprogramowanie oraz na część praktyczną, w której przedstawiona jest sama aplikacja oraz dokładne jej działanie. Wszystko zostało zaprojektowane przy pomocy elementów platformy .NET takich jak ADO.NET, WPF jak i podstawowy język C# oraz bazę danych MS SQL Server.

Pobocznym celem pracy było również sprawdzenie jak bardzo przyjaznym środowiskiem jest platforma .NET oraz język C# dla osób doświadczonych w innych technologiach.

3. Technologie wykorzystane przy projektowaniu aplikacji.

W obecnych czasach nie ma problemu w mieszaniu technologii. Aplikacje napisane w Microsoftowym .NET są w stanie obsługiwać bazy danych postanowione na serwerach Oracle, aczkolwiek dla wygody i kompatybilności, w pracy wykorzystane są technologie tej samej firmy zapewniając najwyższy stopień kompatybilności między poszczególnymi elementami.

3.1. .NET Framework

Platforma programistyczna firmy Microsoft pozwalająca na tworzenie złożonych aplikacji na system operacyjny Windows. Jednym z najważniejszych elementów tej platformy jest to jak kompiluje programy. Jeżeli język programowania spełnia specyfikację Common Intermediate Language (a wszystkie języki obsługiwane przez .NET spełniają), wtedy aplikacja, niezależnie od języka w jakim jest napisana, kompilowana jest przez to samo wspólne środowisko CLR – Common Language Runtime, które pośredniczy pomiędzy samą aplikacją, a systemem Windows. [\[1\]](#)

3.1.1. ADO.NET

ADO.NET jest zbiorem klas pozwalających na interakcję z bazą danych. Pozwalają one nie tylko na połączenie się z serwerem baz danych typu SQL Server oraz interfejsami uzyskującymi dostęp do danych takich jak OLE DB czy ODBC, ale również na odczytywanie jak i modyfikowaniu tych danych [\[2\]](#).

Najważniejszą częścią ADO.NET są dostawcy danych (ang. data provider) [\[3\]](#). Są to obiekty stworzone z myślą o szybki dostępie do danych, są to:

- Connection – obiekt pozwalający na łączenie się z samą bazą danych i zarządzający całym połączeniem.
- Command – obiekt odpowiedzialny za odczytywanie i modyfikowanie danych przy pomocy zapytań, może również korzystać z procedur składowanych.
- DataReader – Pozwala na odczyt danych zawróconych w wyniku zapytania w obiekcie Command.
- DataAdapter – Pośredniczy w komunikacji między obiektem DataSet oraz samą bazą danych. Obiekt DataSet pozwala przechowywać w aplikacji dane z bazy

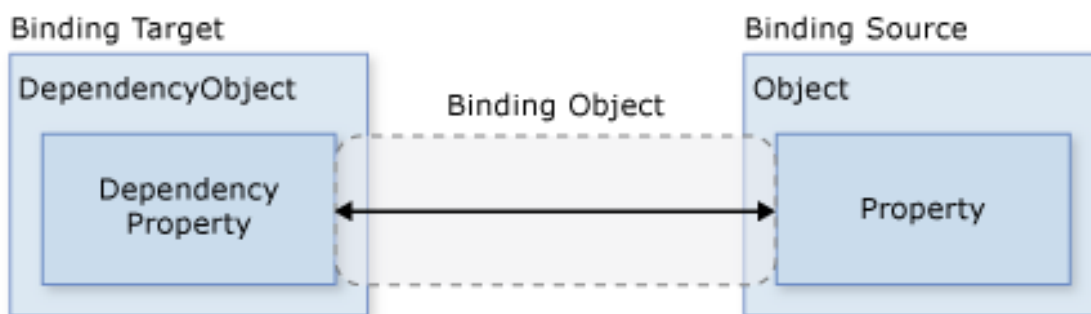
danych zachowując przy tym ich strukturę tabeli składającej się z kolumn i wierszy.

3.1.2. .NET WPF

WPF, czyli Windows Presentation Foundation jest to silnik graficzny pozwalający na tworzenie aplikacji windowsowych łączący język znaczników (ang. markup language) odpowiedzialnego za wygląd oraz technikę tzw. code-behind, która oddziela pliki odpowiedzialne za wygląd (w tym przypadku pliki XAML) oraz pliki odpowiedzialne za funkcjonowanie samej aplikacji.

WPF pozwala programiście na użycie kontroltek (ang. controls), są to odpowiednio przygotowane już klasy posiadające interfejs użytkownika oraz posiadających zdefiniowane zachowanie. To one stanowią główną interakcję między aplikacją, a użytkownikiem w postaci przycisków, list czy tabel.

Ważnym elementem aplikacji korzystających z WPF jest łączenie danych (ang. data binding). Służy ona połączeniu kontrolki w części wizualnej z danymi przygotowywanymi w kodzie. W momencie, kiedy nastąpi zmiana w danych połączonych, kontrolka jest wyświetlająca automatycznie zostanie odświeżona i pokaże aktualne dane. Relację połączonej kontrolki z kodem przedstawia rysunek 3.1.



Rysunek 3.1. Graf relacji przedstawiający łączenie danych [4]

3.2. Microsoft SQL Server

MS SQL Server jest to system zarządzający relacyjnymi bazami danych, którego głównym celem jest przechowywanie i udostępnianie danych innym aplikacjom. Edycja Express jest darmową wersją tego systemu posiadającą ograniczenia na polu technicznym

jak możliwości procesora, wielkość baz danych oraz bardziej zaawansowane opcje zarządzania, ale nie ogranicza niczego związanego z ich obsługą. W pracy użyta została okrojona wersja LocalDB, jest ona kierowana dla początkujących programistów lub studentów. Najważniejszą różnicą jest fakt, że nie zezwala ona na zdalny dostęp do serwera więc baza danych musi być przechowywana lokalnie [\[5\]](#).

3.3. Język programowania C#

C# jest obiektowym językiem programowania stworzonym przez Microsoft z myślą o platformie .NET. Oprócz spełniania podstawowych założeń paradygmatów języków obiektowych, ważnym jest wspomnienie jak system obsługuje aplikacje w tym języku.

Czerpie on swoje podstawy z języka C/C++ i dodaje do tego rozwiązania obecne w językach obiektowych takie jak automatyczne zwalnianie pamięci (garbage collection), obsługa wyjątków czy też wyrażenia lambda. Ważnym element języka C# jest również technologia LINQ [\[6\]](#). Domyślnie, zapytania do baz danych w języku C# są zwykłym typem string, jednak technologia LINQ, zamienia to zapytanie w obiekt. Dzięki temu nie tylko kompilator jest w stanie wykryć błędy, a system IntelliSense pomóc w tworzeniu zapytania, ale również zapewnia uniwersalność. Programista nie musi znać dokładnej składni zapytania SQL w konkretnej bazie danych, z której aplikacja czerpie dane, system LINQ przystosuje zapisane zapytanie do konkretnej bazy za niego [\[7\]](#).

Aplikacje napisane w języku C#, czy też innych językach obsługiwanych przez platformę .NET, nie są wykonywane na systemie użytkownika, a na wirtualnej maszynie zaimplementowanej w .NET Framework.

4. Narzędzia wykorzystane przy projektowaniu aplikacji

Tak jak w przypadku technologii, w pracy zostały wykorzystane narzędzia dedykowane do obsługi tych właśnie technologii, które zostały stworzone przez tę samą firmę. Jest to Microsoft Visual Studio 2019 do tworzenia części aplikacyjnej oraz SQL Server Management Studio do obsługi bazy danych.

4.1. Microsoft Visual Studio 2019

Visual studio jest zaawansowanym środowiskiem programistycznym firmy Microsoft. Pozwala ono na wygodnie tworzenie aplikacji w językach tej firmy takich jak C++, C# czy też Visual Basic [8]. Jedną z największych cech i zalet Visual Studio jest system IntelliSense, który ułatwia pisanie kodu podpowiadając zmiany, parametry metod i potencjalne rozwiązania problemów. Dzięki niemu, osoby nawet mało zapoznane z narzędziami otrzymują podpowiedzi w zależności od kontekstu kodu.

Pozwala on również na integrację z systemem Git oraz wspólne zdalne pisanie kodu dzięki czemu znacznie ułatwione jest zarządzanie kodem w projektach kilkusobowych.

Posiada on również wbudowaną wyszukiwarkę pakietów NuGet dzięki czemu programista może z łatwością wyszukiwać pakietów nie zawartych w domyślnych klasach .NET.

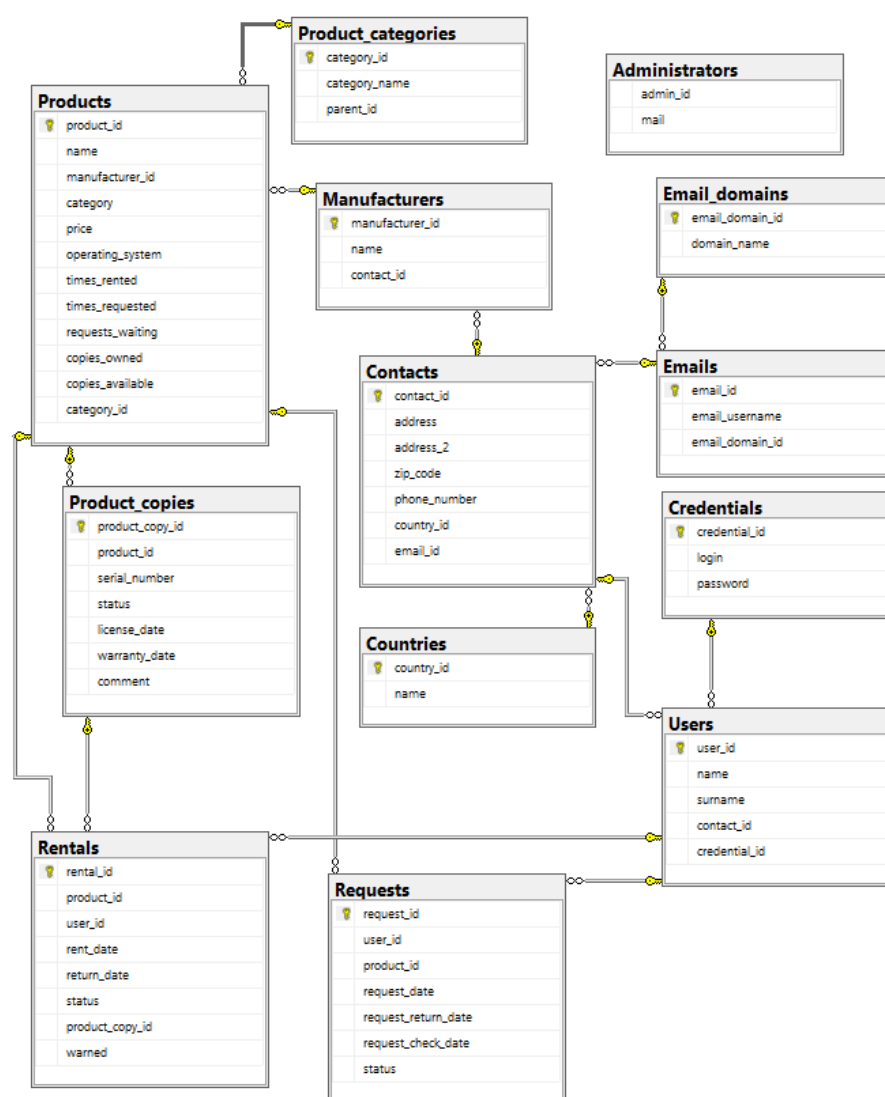
4.2. SQL Server Management Studio

Środowisko graficzne pozwalające na obsługę baz danych oraz innych komponentów wchodzących w skład MS SQL Server. Pozwala ono na połączenie się z serwerem i wykonywanie operacji na znajdujących się tam bazach danych, począwszy od tworzenia takowych po modyfikacje ich i wprowadzanie nowych wartości.

SSMS obsługuje również język T-SQL, który jest odmianą języka SQL używaną przez Microsoft. Pozwala on na wykorzystanie podstawowych operacji programistycznych takich jak pętle, zmienne czy instrukcje warunkowe w samych wyzwalaczach czy procedurach składowanych po stronie bazy.

5. Opis bazy danych

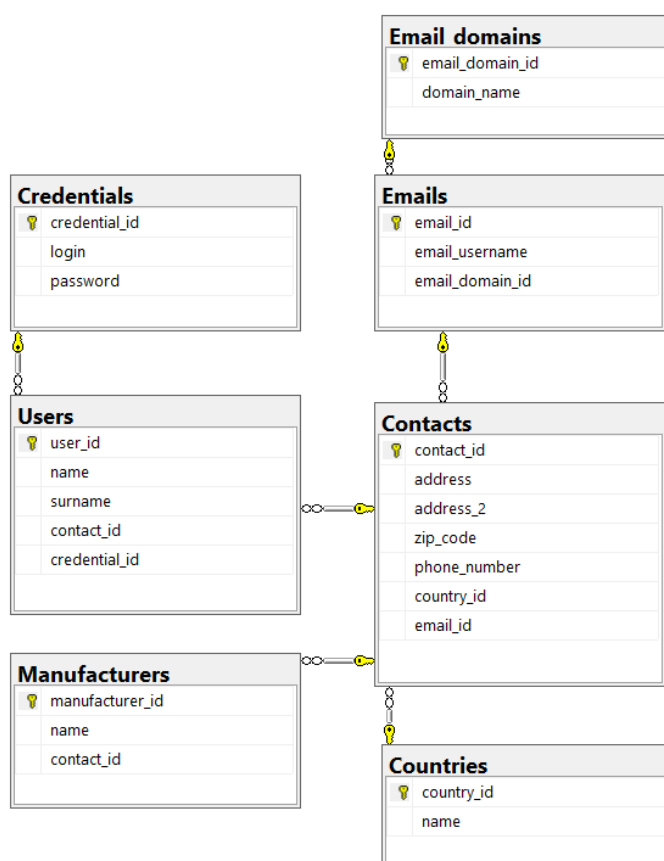
Serwer wykorzystany w pracy, LocalDB, jest to okrojona wersja darmowej edycji SQL Server Express. Z racji, że jest to pojedyncza, samodzielnie działająca aplikacja, która z bazy danych korzysta jedynie w celu przechowywania danych, aniżeli ich badaniu i analizie, taka wersja jest wystarczająca na potrzeby tej pracy. Istnieje możliwość polepszenia serwera na pełną wersję SQL Server Express, a nawet na jego płatnego odpowiednika w razie potrzeby i dalszego rozwijania programu.



Rys. 5.1. Pelen schemat bazy danych

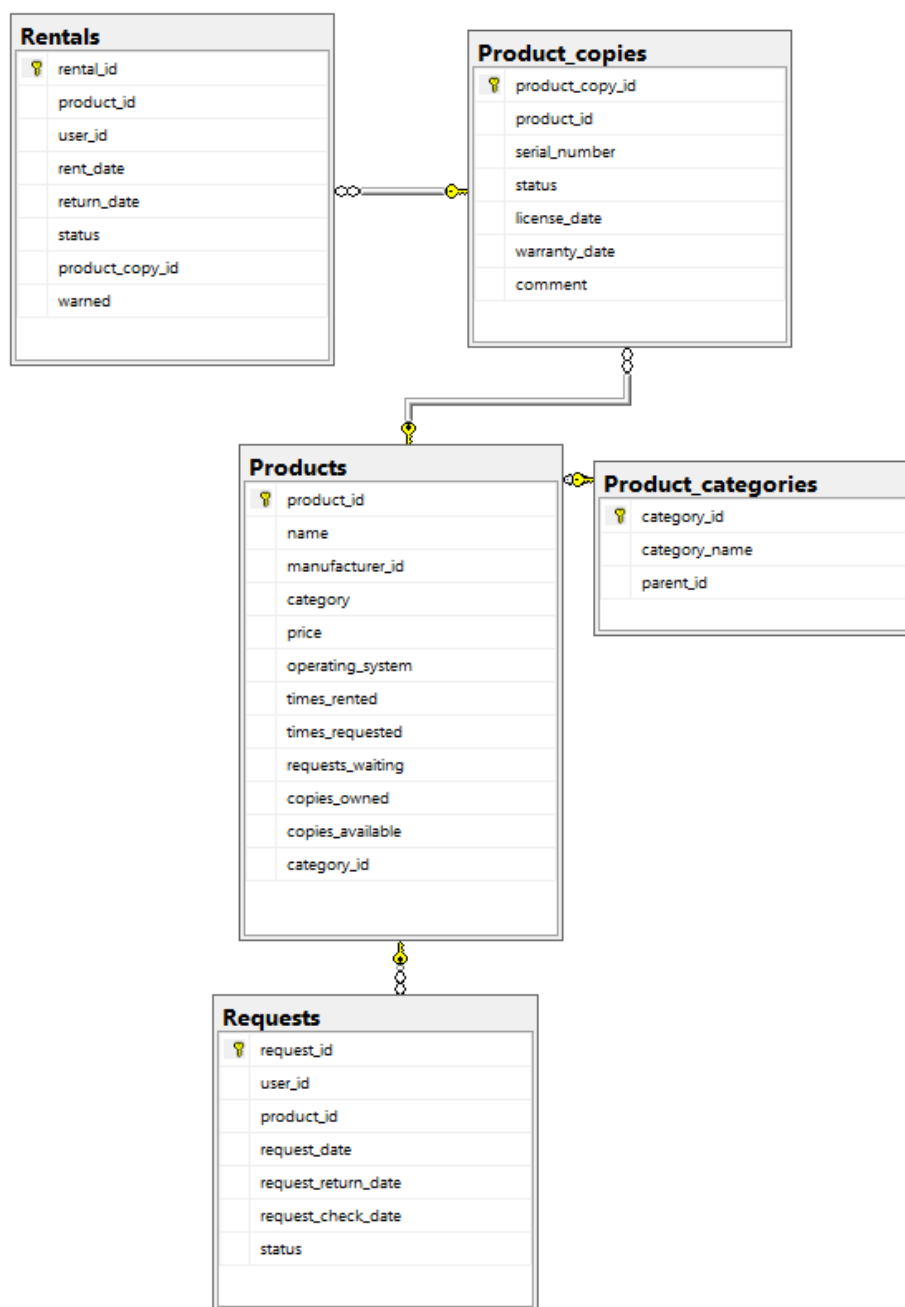
Na rys. 5.1. przedstawiono cały schemat relacyjnej bazy danych na, z której aplikacja korzysta i zapisuje w niej dane do przechowania. Można w niej wyróżnić dwie grupy tabel odpowiedzialnych za różną część aplikacji.

Część odpowiedzialną za zarządzanie użytkownikami, producentami oraz informacjami z nimi powiązanymi (Rys. 5.2.) to tabele przechowujące informacje każdego użytkownika potrzebne do kontaktu z nim w postaci adresów e-mail, numeru telefonu czy też adresu zamieszkania. Z tej samej tabeli (Contacts) korzysta również tabela producentów dzięki czemu niezależnie czy jest to kontakt do pracownika w firmie czy też producenta danego sprzętu, informacje znajdują się w jednej, wspólnej tabeli dzięki czemu nie ma potrzeby powielania atrybutów encji w dwóch osobnych tabelach. W kwestii zarządzania użytkownikami, występuje również tabela Credentials, która przechowuje dane potrzebne do zalogowania się do systemu w postaci loginu oraz zahasowanego hasła. Adres e-mail został podzielony na dwie tabele w celu zmniejszenia powielania danych w postaci tych samych domen mailowych.



Rys. 5.2. Tabele utrzymujące część zarządzająco-kontaktową

Drugą grupą tabel są tabele odpowiedzialne za utrzymywanie informacji na temat samego sprzętu oraz system wypożyczania takowego (Rys. 5.3.).



Rys. 5.3. Tabele utrzymujące część sprzętowo-wypożyczeniową

W tych tabelach można wyróżnić dwa podsystemy, system próśb o produkt oraz system właściwego wypożyczenia produktu pracownikowi. Ten pierwszy (tabela

Requests) przechowuje informacje na temat jaki użytkownik wysłał prośbę o jaki produkt oraz daty wysłania takowej prośby jak i wstępnej daty zwrócenia takowego.

Drugim systemem (tabela Rentals) jest przechowywanie informacji na temat udanych wypożyczeń oraz ich statusu. Widnieje tam informacja nie tylko na temat jaki użytkownik posiada jaki produkt, ale również jaki egzemplarz danego produktu został mu użyczony. Została również przeprowadzona normalizacja w celu usunięcia relacji tzw. wiele do wielu i wprowadzono tabelę Product_copies jako tabelę pośredniczącą pomiędzy Rentals oraz Products. Encja Product_copies zawiera informacje potrzebne do rozróżnienia w przypadku każdego egzemplarza danego produktu takie jak numer seryjny czy też data wygaśnięcia licencji lub gwarancji. Reszta informacji, wspólnych dla wszystkich kopii danego produktu znajduje się w tabeli Products. Rzeczy takie jak nazwa producenta, cena w przypadku potrzeby zakupu nowych egzemplarzy oraz informacje używane w statystyce danych produktów dostępnej w samej aplikacji. Przygotowana została również osobna tabela na kategorie danego produktu (Tabela 5.1), w której wprowadzono hierarchiczną strukturę danych mającą służyć bardziej zaawansowanemu podziałowi produktów, jednakże w obecnej wersji aplikacji nie jest ona wykorzystywana, a w jej miejsce jest wprowadzona odpowiadająca kolumna w tabeli Products.

Tabela 5.1 Kolumny oraz rekordy encji Product_categories

Lp.	category_id	category_name	parent_id
1	1	Assortment	0
2	2	Hardware	1
3	3	Software	1
4	4	Peripheral	2
5	5	Monitor	4
6	6	Mouse	4
7	7	Keyboard	4
8	8	Microphone	4
9	9	External_drive	4
10	10	Computer	2
11	11	Antivirus	3
12	12	OS	3
13	13	Graphic_editor	3
14	14	Audio_editor	3
15	15	Video_editor	3

Ostatnią, względnie odizolowaną encją jest Administrators. Przechowuje ona jedynie informacje potrzebne do odróżnienia w aplikacji użytkownika od administratora jak i hasło potrzebne do systemu powiadomień.

5.1. Struktura tabel i ograniczenia

Elementem wspólnym struktury wszystkich tabel jest klucz główny, którym jest pole z nazwą tabeli i zakończeniem id. Jedyną tabelą wyróżniającym się na tym tle jest tabela Administrators, która nie posiada klucza głównego, a jedynie te id użytkowników, którzy mają uprawnienia administratora.

5.1.1. Tabela Rentals

Sercem całego programu jest tabela Rentals, przechowuje ona kluczowe informacje w postaci jaki użytkownik(user_id) posiada jaki wypożyczony sprzęt(product_copy_id). Kolejnymi polami są informacje kiedy sprzęt został użytkownikowi wypożyczony oraz kiedy ma on go zwrócić. Dwoma ostatnimi polami są pola status, które rozdziela wypożyczenia na te w trakcie, a te już zakończone i zwrócone, oraz pole warned, które jest użyte w procesie wysyłania powiadomień mailowych w momencie zbliżania się terminu zwrotu produktu (Rys. 5.4.).

Rentals			
	Column Name	Data Type	Allow Nulls
🔑	rental_id	int	<input type="checkbox"/>
	user_id	int	<input type="checkbox"/>
	rent_date	date	<input checked="" type="checkbox"/>
	return_date	date	<input checked="" type="checkbox"/>
	status	varchar(10)	<input type="checkbox"/>
	product_copy_id	int	<input type="checkbox"/>
	warned	varchar(6)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Rys. 5.4. Tabela Rentals

```
CREATE TABLE [dbo].[Rentals](  
    [rental_id] [int] IDENTITY(1,1) NOT NULL,  
    [user_id] [int] NOT NULL,  
    [rent_date] [date] NULL,  
    [return_date] [date] NULL,  
    [status] [varchar](10) NOT NULL,
```

```

        [product_copy_id] [int] NOT NULL,
        [warned] [varchar](6) NULL,
PRIMARY KEY CLUSTERED
(
    [rental_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Rentals] WITH CHECK ADD FOREIGN KEY([product_copy_id])
REFERENCES [dbo].[Product_copies] ([product_copy_id])
GO


ALTER TABLE [dbo].[Rentals] WITH CHECK ADD FOREIGN KEY([user_id])
REFERENCES [dbo].[Users] ([user_id])
GO

```

Listing 5.1. Skrypt tworzący tabelę Rentals

5.1.2. Tabela Products

Drugą najważniejszą tabelą jest tabela samych produktów, które firma posiada na stanie. Pola tej tabeli można podzielić na kilka grup, pierwszą są informacje ogólne na temat produktu, które są wspólne dla każdego dostępnego egzemplarzu takie jak nazwa samego produktu, klucz obcy wskazujący na producenta, kategoria hardware lub software, sama cena w przypadku potencjalnej potrzeby zakupu kolejnych egzemplarzy oraz nazwa systemu operacyjnego w przypadku oprogramowania. Kolejną grupą są informacje statystyczne, to ile razy dany produkt został wypożyczony, to ile razy została na niego wysłana prośba oraz ile osób czeka w kolejne na jego wypożyczenie. Ostatnią grupą są 2 pola informujące o dostępności, ilość egzemplarzy, które firma posiada oraz ile z tych egzemplarzy jest dostępnych do wypożyczenia. Ostatnie pole category_id służy wstępnemu rozwinięciu systemu kategoryzowania przedmiotów (Rys. 5.5.). Tabela posiada również ustawione wartości domyślne w postaci 0 dla pól związanych ze statystyką.

Products			
	Column Name	Data Type	Allow Nulls
	product_id	int	<input type="checkbox"/>
	name	varchar(20)	<input type="checkbox"/>
	manufacturer_id	int	<input type="checkbox"/>
	category	varchar(10)	<input type="checkbox"/>
	price	decimal(10, 2)	<input type="checkbox"/>
	operating_system	varchar(15)	<input checked="" type="checkbox"/>
	times_rented	int	<input type="checkbox"/>
	times_requested	int	<input type="checkbox"/>
	requests_waiting	int	<input type="checkbox"/>
	copies_owned	int	<input type="checkbox"/>
	copies_available	int	<input type="checkbox"/>
	category_id	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Rys. 5.5. Tabela Products

```

CREATE TABLE [dbo].[Products](
    [product_id] [int] IDENTITY(1,1) NOT NULL,
    [name] [varchar](20) NOT NULL,
    [manufacturer_id] [int] NOT NULL,
    [category] [varchar](10) NOT NULL,
    [price] [decimal](10, 2) NOT NULL,
    [operating_system] [varchar](15) NULL,
    [times_rented] [int] NOT NULL,
    [times_requested] [int] NOT NULL,
    [requests_waiting] [int] NOT NULL,
    [copies_owned] [int] NOT NULL,
    [copies_available] [int] NOT NULL,
    [category_id] [int] NULL,
    PRIMARY KEY CLUSTERED
    (
        [product_id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
    [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Products] WITH CHECK ADD FOREIGN KEY([category_id])
REFERENCES [dbo].[Product_categories] ([category_id])
GO

ALTER TABLE [dbo].[Products] WITH CHECK ADD FOREIGN KEY([manufacturer_id])
REFERENCES [dbo].[Manufacturers] ([manufacturer_id])
GO

```

Listing 5.2. Skrypt tworzący tabelę Products

5.1.3. Tabela Product_copies

Równie ważną jest tabela Product_copies, które funkcjonuje jako tabela pośrednia pomiędzy tabelami Rentals oraz Products. Dzięki niej usunięta jest relacja wielu do wielu, a aplikacja pozwala na wypożyczenia poszczególnych egzemplarzy aniżeli samego typu produktu. Oprócz klucza obcego do produktu, posiada ona informacje, które mogą się różnić w zależności od egzemplarza tego samego produktu, są to status – opisujący czy dany egzemplarz jest wolny, w użyciu lub może być uszkodzony, numer seryjny lub inny kod unikatowy dla każdego z wyprodukowanych egzemplarzy, data końca gwarancji w przypadku produktu hardware oraz data wygaśnięcia licencji w przypadku software. Ostatnim polem jest pole pozwalające na zostawienie komentarza np. precyzującego co jest uszkodzone w danym egzemplarzu (Rys. 5.6.).

Product_copies			
	Column Name	Data Type	Allow Nulls
🔑	product_copy_id	int	<input type="checkbox"/>
	product_id	int	<input type="checkbox"/>
	serial_number	varchar(50)	<input type="checkbox"/>
	status	varchar(15)	<input type="checkbox"/>
	license_date	date	<input checked="" type="checkbox"/>
	warranty_date	date	<input checked="" type="checkbox"/>
	comment	varchar(150)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Rys. 5.6. Tabela Product_copies

```
CREATE TABLE [dbo].[Product_copies](
    [product_copy_id] [int] IDENTITY(1,1) NOT NULL,
    [product_id] [int] NOT NULL,
    [serial_number] [varchar](50) NOT NULL,
    [status] [varchar](15) NOT NULL,
    [license_date] [date] NULL,
    [warranty_date] [date] NULL,
    [comment] [varchar](150) NULL,
    PRIMARY KEY CLUSTERED
    (
        [product_copy_id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
    [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Product_copies] WITH CHECK ADD FOREIGN KEY([product_id])
REFERENCES [dbo].[Products] ([product_id])
GO
```

Listing 5.3. Skrypt tworzący tabelę Product_copies

5.1.4. Tabela Requests

Tabela zapewniająca komunikację pomiędzy użytkownikiem, a administratorem. To ona przechowuje informacje o tym jaki użytkownik wysłał prośbę o wypożyczenie jakiego sprzętu. Składa się ona z dwóch kluczy obcych w postaci id użytkownika oraz id produktu, 3 pól z datą – data złożenia prośby, wstępna data zwrotu zaproponowana przez użytkownika oraz data kiedy administrator zapoznał się i zatwierdził lub odrzucił prośbę. Ostatnie pole ze statusem przechowuje werdykt tej decyzji lub informację, że użytkownik sam wycofał prośbę (Rys. 5.7.). Data złożenia prośby jest przechowywana w formacie, który przetrzymuje również część godzinową daty, zabieg jest wykonany w celu ustalenia kolejki do produktu jeżeli kilka osób wyśle prośbę na ten sam produkt.

Requests			
	Column Name	Data Type	Allow Nulls
🔑	request_id	int	<input type="checkbox"/>
	user_id	int	<input type="checkbox"/>
	product_id	int	<input type="checkbox"/>
	request_date	datetime	<input type="checkbox"/>
	request_return_date	date	<input type="checkbox"/>
	request_check_date	date	<input checked="" type="checkbox"/>
	status	varchar(10)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Rys. 5.7. Tabela Requests

```
CREATE TABLE [dbo].[Requests](
    [request_id] [int] IDENTITY(1,1) NOT NULL,
    [user_id] [int] NOT NULL,
    [product_id] [int] NOT NULL,
    [request_date] [datetime] NOT NULL,
    [request_return_date] [date] NOT NULL,
    [request_check_date] [date] NULL,
    [status] [varchar](10) NULL,
    PRIMARY KEY CLUSTERED
    (
        [request_id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
    [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Requests] WITH CHECK ADD CONSTRAINT [FK_Requests_Products]
FOREIGN KEY([product_id])
REFERENCES [dbo].[Products] ([product_id])
GO
```

```

ALTER TABLE [dbo].[Requests] CHECK CONSTRAINT [FK_Requests_Products]
GO

ALTER TABLE [dbo].[Requests] WITH CHECK ADD CONSTRAINT [FK_Requests_Users] FOREIGN
KEY([user_id])
REFERENCES [dbo].[Users] ([user_id])
GO


ALTER TABLE [dbo].[Requests] CHECK CONSTRAINT [FK_Requests_Users]
GO

```

Listing 5.4. Skrypt tworzący tabelę Requests

5.1.5. Tabela Contacts

Z części tabel odpowiedzialnych za komunikację, najważniejszą jest tabela Contacts, która przechowuje dane adresowe użytkowników jak i producentów. Pierwsze pola przechowują podstawowe dane w postaci adresu, kodu pocztowego czy też numeru telefonu (Rys. 5.8.). Pole country_id to klucz obcy do tabeli przechowującej państwa już zapisane w systemie, to z tej tabeli administrator wybiera kraj pochodzenia, a w przypadku braku takowego, dodaje nowy kraj. Ostatnie pole to klucz do tabeli tworzącej adresy email, znajduje się tam nazwa adresu oraz kolejny klucz obcy prowadzący do tabeli przechowującej domeny, na których poczta jest utrzymywana. Rozwiązanie to pozwala zmniejszyć ilość powtórzeń w przypadku kiedy np. 20 osób posiada pocztę na domenie @gmail.com.

Contacts			
	Column Name	Data Type	Allow Nulls
	contact_id	int	<input type="checkbox"/>
	address	varchar(30)	<input checked="" type="checkbox"/>
	address_2	varchar(30)	<input checked="" type="checkbox"/>
	zip_code	varchar(10)	<input checked="" type="checkbox"/>
	phone_number	varchar(15)	<input type="checkbox"/>
	country_id	int	<input type="checkbox"/>
	email_id	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Rys. 5.8. Tabela Contacts

```

CREATE TABLE [dbo].[Contacts](
    [contact_id] [int] IDENTITY(1,1) NOT NULL,
    [address] [varchar](30) NULL,
    [address_2] [varchar](30) NULL,
    [zip_code] [varchar](10) NULL,
    [phone_number] [varchar](15) NOT NULL,
    [country_id] [int] NOT NULL,
    [email_id] [int] NULL,
PRIMARY KEY CLUSTERED
(
    [contact_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Contacts] WITH CHECK ADD FOREIGN KEY([country_id])
REFERENCES [dbo].[Countries] ([country_id])
GO

ALTER TABLE [dbo].[Contacts] WITH CHECK ADD FOREIGN KEY([email_id])
REFERENCES [dbo].[Emails] ([email_id])
GO

```

Listing 5.5. Skrypt tworzący tabelę Contacts

6. Opis aplikacji

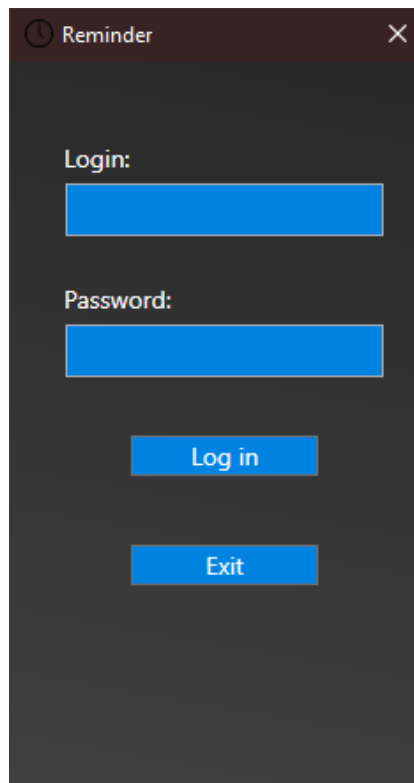
Wygląd aplikacji jest oparty na dwóch elementach dostępnych w WPF, oknie oraz stronach. Występują 3 osobne okna: okno logowania, okno użytkownika oraz okno administratora. Kiedy aplikacja przejdzie już do jednego z tych okien, wewnątrz znajduje się panel menu z przyciskami funkcyjnymi stworzonego z połączenia dock panel i stack panel, w przypadku administratora dochodzi również scroll viewer z racji większej ilości przycisków. Pod tym znajduje się wiadomość witająca użytkownika z imienia i nazwiska jak i przycisk do wylogowania.

Prawa część okna aplikacji składa się z elementu frame, w którym ładowane są wybrane strony podpięte pod przyciski z menu.

6.1. Część wspólna

W celu zmniejszenia powtarzalności kodu jak i umożliwienia działania aplikacji w postaci wielu stron, w aplikacji zostało użytych kilka rozwiązań w postaci klas pomocniczych niezależnych od tego czy użytkownik logujący się ma uprawnienia administratora czy też nie. Oprócz nich częścią wspólną jest również okno logowania, które jest pierwszą rzeczą jaką zobaczy użytkownik po włączeniu aplikacji.

6.1.1. Menu logowania



Rys. 6.1. Okno logowania

Menu logowania (Rys. 6.1.) jest wspólne dla każdego użytkownika i to w tym miejscu następuje weryfikacja użytkownika w postaci sprawdzenia czy takowy istnieje w systemie (Listing 6.1.), przypisanie id użytkownika do sesji, jeżeli poprzednie sprawdzenie się powiodło i udało się zalogować oraz sprawdzenie czy takowe id należy do administratora. Ostatecznie aplikacja przejdzie do odpowiedniego okna, użytkownika lub administratora.

```
SqlCommand command = new SqlCommand("SELECT credential_id FROM Credentials WHERE login  
= @UserLogin AND password = @UserPassword", sqlConnection);  
command.Parameters.AddWithValue("@UserLogin", LoginTextBox.Text);  
command.Parameters.AddWithValue("@UserPassword", HashedPass);  
  
SqlDataReader reader = command.ExecuteReader();  
  
if (reader.HasRows)  
{  
    reader.Read();  
  
    CredID = (int)reader["credential_id"];  
    User = true;  
}  
  
else  
{  
    MessageBox.Show("Incorrect login or password");  
    PasswordTextBox.Clear();  
}
```

```

}
reader.Close();
command.Parameters.Clear();

```

Listing 6.1. Weryfikacja czy użytkownik istnieje

6.1.2. Połączenie z bazą danych

Aplikacja posiada osobną klasę odpowiedzialną za obsługę połączenia się z bazą danych. Posiada ona konstruktor (Listing 6.2.) pobierający informacje na temat lokalnego pliku bazy danych z pliku App (Listing 6.3.).

```

ConnectionStringSettings settings = ConfigurationManager.ConnectionStrings["DB"];
_Connection = new SqlConnection(settings.ConnectionString);

```

Listing 6.2. Konstruktor połączenia z bazą danych

```

<connectionStrings>
  <add name="DB" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;
Initial Catalog=Praca;
Integrated Security=True;
AttachDbFileName=|DataDirectory|Praca.mdf"
  providerName="System.Data.SqlClient" />
</connectionStrings>

```

Listing 6.3. Informacje połączeniowe z pliku App.config

6.1.3. Sesja

Klasa sesji skupia się na dwóch rzeczach, przechowywaniu informacji o użytkowniku oraz przechowywaniu informacji o jego przechodzeniu przez aplikację.

Jeżeli chodzi o informacje o użytkowniku, klasa posiada 2 pola prywatne, `_UserID` oraz `_EditID`. To pierwsze zajmuje się przechowywaniem id użytkownika pobranym z bazy danych podczas logowania i jest wykorzystywane za każdym razem, kiedy użytkownik chce np. wysłać prośbę o produkt albo sprawdzić produkty jemu udostępnione. `_EditID` natomiast zajmuje się przechowywaniem id przedmiotu, który administrator właśnie wybrał do edycji i przechodzi do innego okna, w którym mają się wyświetlić informacje dotyczące tego szczególnego przedmiotu.

Drugą funkcjonalnością sesji jest zapamiętywanie ruchu użytkownika po aplikacji, osiągnięte jest to za pomocą stosu LIFO. Przy każdym przejściu do kolejnego okna, poprzednie jest wrzucane na stos i w momencie, kiedy użytkownik chciałby się cofnąć przy pomocy przycisku return na aplikacji, ten zabierze go do poprzedniego okna.

6.1.4. Plik App.xaml

Najważniejszym elementem w pliku App.xaml jest automatyczne ustawienie stylu dla wielu kontrolki na przestrzeni całej aplikacji w celu niepowielania kodu na każdej kolejnej stronie.

```
<Style x:Key="Labels" TargetType="{x:Type Control}">
    <Setter Property="Foreground" Value="White" />
</Style>
<Style x:Key="Boxes" TargetType="{x:Type Control}">
    <Setter Property="Background" Value="#FF0082E1"/>
    <Setter Property="Foreground" Value="White"/>
    <Setter Property="VerticalContentAlignment" Value="Center"/>
</Style>
<Style x:Key="Date" TargetType="{x:Type DatePicker}">
    <Setter Property="Background" Value="#FF0082E1"/>
</Style>

<Style TargetType="{x:Type Label}" BasedOn="{StaticResource Labels}"/>
<Style TargetType="{x:Type CheckBox}" BasedOn="{StaticResource Labels}"/>
<Style TargetType="{x:Type TextBox}" BasedOn="{StaticResource Boxes}"/>
<Style TargetType="{x:Type PasswordBox}" BasedOn="{StaticResource Boxes}"/>
<Style TargetType="{x:Type Button}" BasedOn="{StaticResource Boxes}"/>
<Style TargetType="{x:Type ListBox}" BasedOn="{StaticResource Boxes}"/>
<Style TargetType="{x:Type DatePicker}" BasedOn="{StaticResource Date}"/>
```

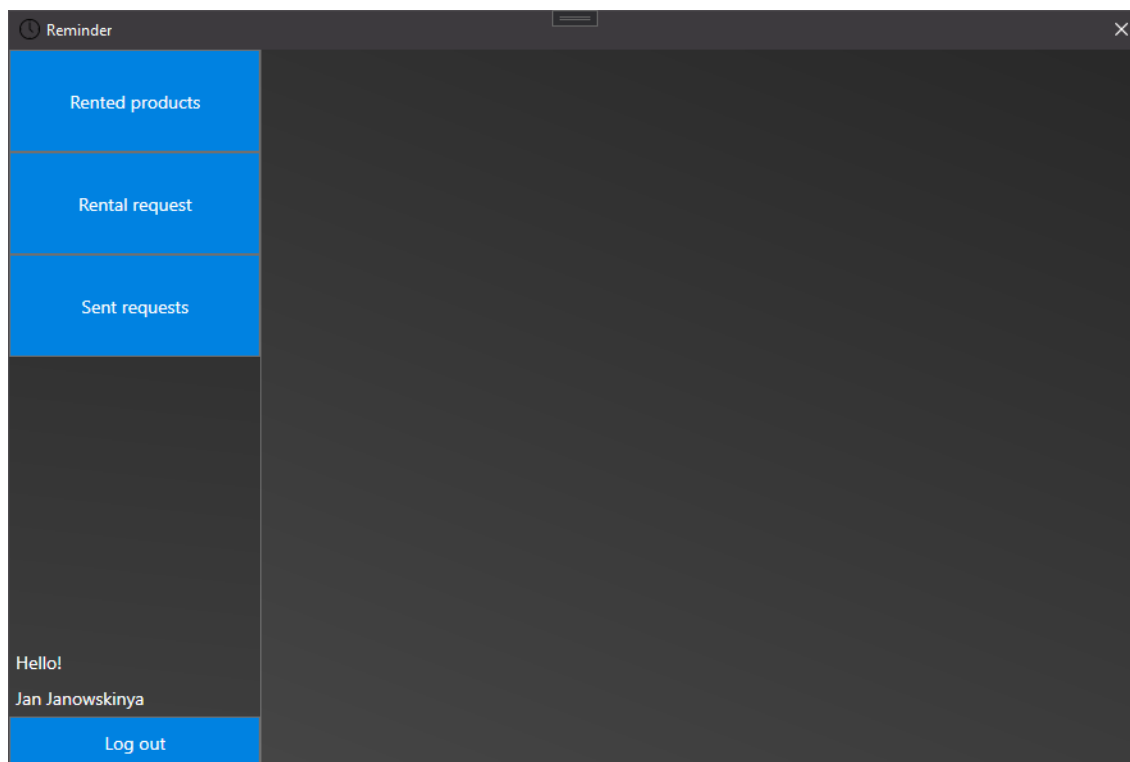
Listing 6.4. Ustawienia stylu aplikacji

Na listingu 6.4. widnieje automatyczna zmiana koloru wielu rodzaju elementów użytych w całej aplikacji. Dzięki temu zabiegowi, każda strona automatycznie ustawi kolory wymienionych elementów na te z pliku App.xaml.

Drugą rzeczą znajdującą się w tym pliku jest zmodyfikowany arkusz stylu dla kontrolki combobox. Jest to na tyle zaawansowany element, że nie da się go zmodyfikować jednym seterem w stylu i należy zmienić wartości z całym jego szablonie.

6.2. Część użytkownika

W momencie udanego zalogowania oraz stwierdzenie przez bazę danych, że id logującego się użytkownika, nie należy do id administratora, wyświetlone zostanie okno użytkownika (Rys. 6.2.).

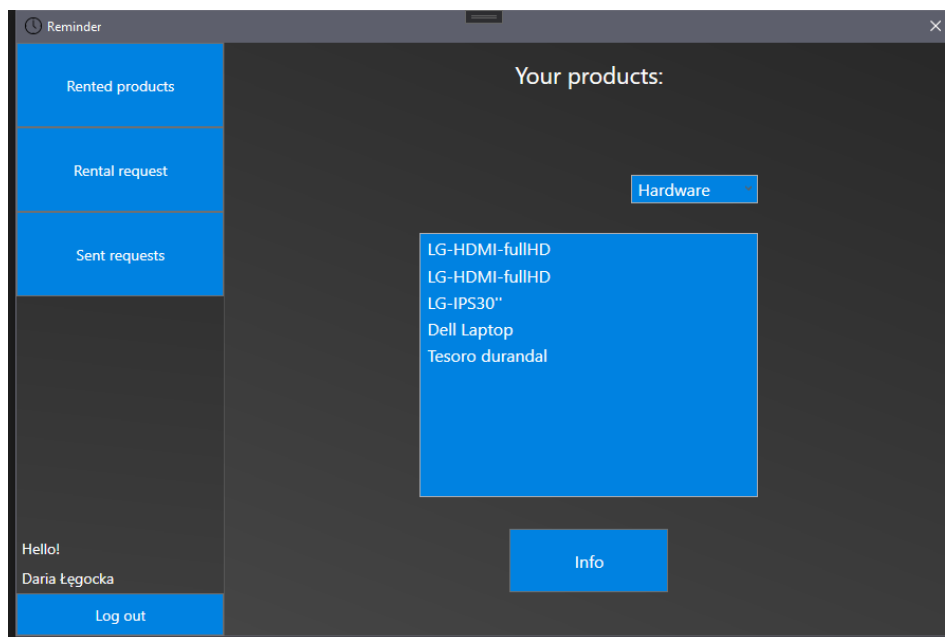


Rys. 6.2. Okno użytkownika

Użytkownik ze swojego panelu ma dostęp do 3 stron: wypożyczone przez niego produkty, wysłanie prośby o wypożyczenie produktu oraz lista próśb już wysłanych i oczekujących na przetworzenie.

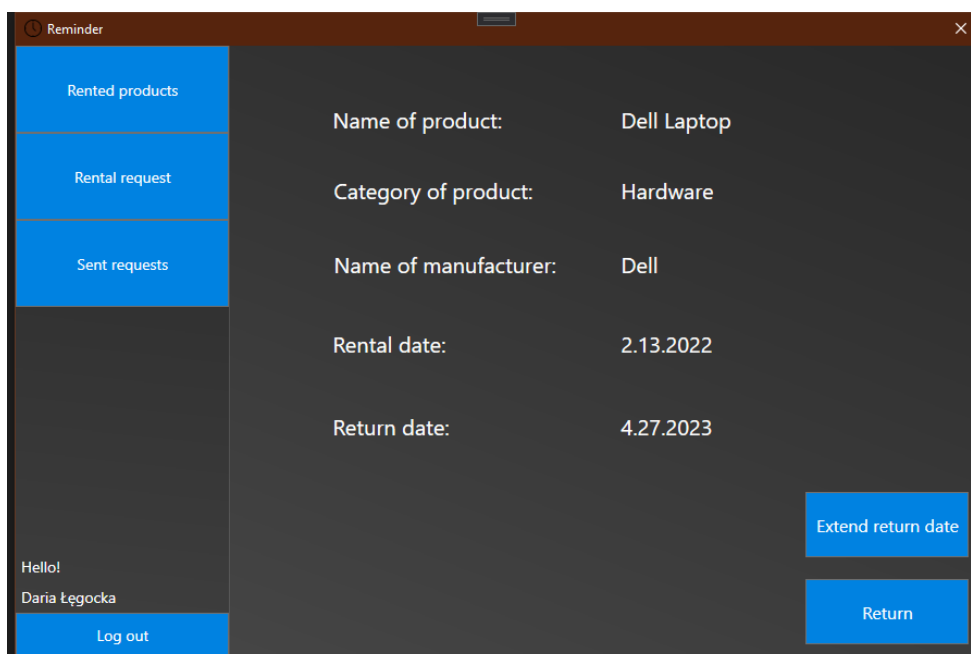
6.2.1. Wypożyczone produkty

W pierwszym panelu użytkownik może zobaczyć wszystkie swoje wypożyczone przedmioty z podziałem listy na przedmioty typu hardware oraz software (Rys. 6.3.). Z tego okna użytkownik może wybrać przedmiot i przejść dalej w szczegółowe informacje na temat danego wypożyczenia.



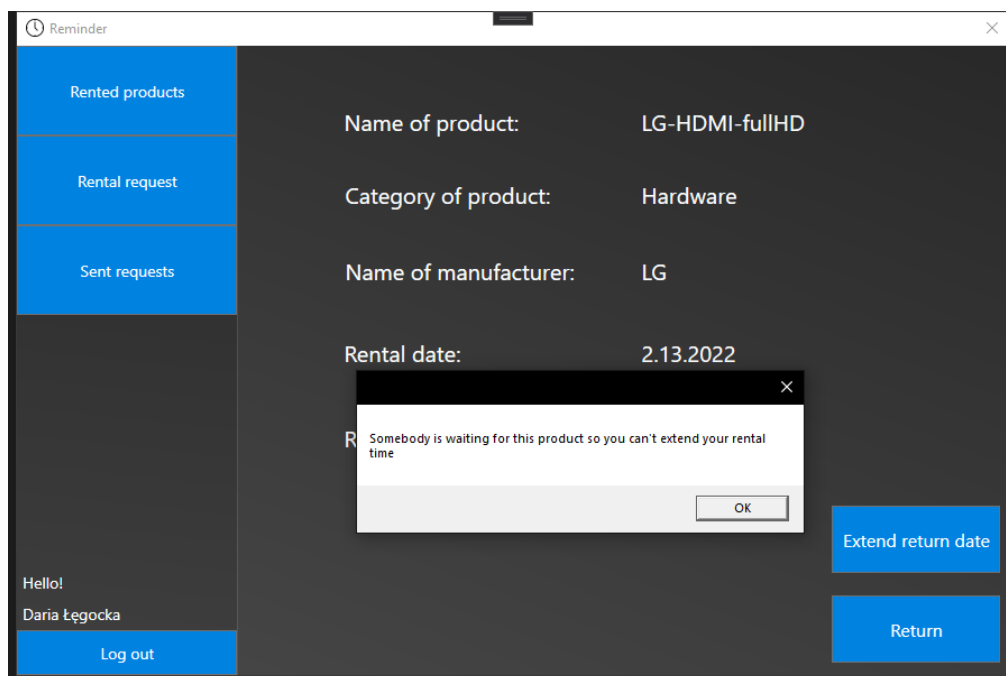
Rys. 6.3. Lista wypożyczonych produktów

W panelu informacyjnym (Rys. 6.4.) umieszczone są podstawowe informacje o produkcie oraz dwie daty: pierwsza mówiąca, kiedy dany produkt został wypożyczony oraz druga mówiąca do kiedy produkt należy zwrócić albo przedłużyć. Z tego samego okna, jest możliwość przejścia do procesu przedłużenie wypożyczenia.



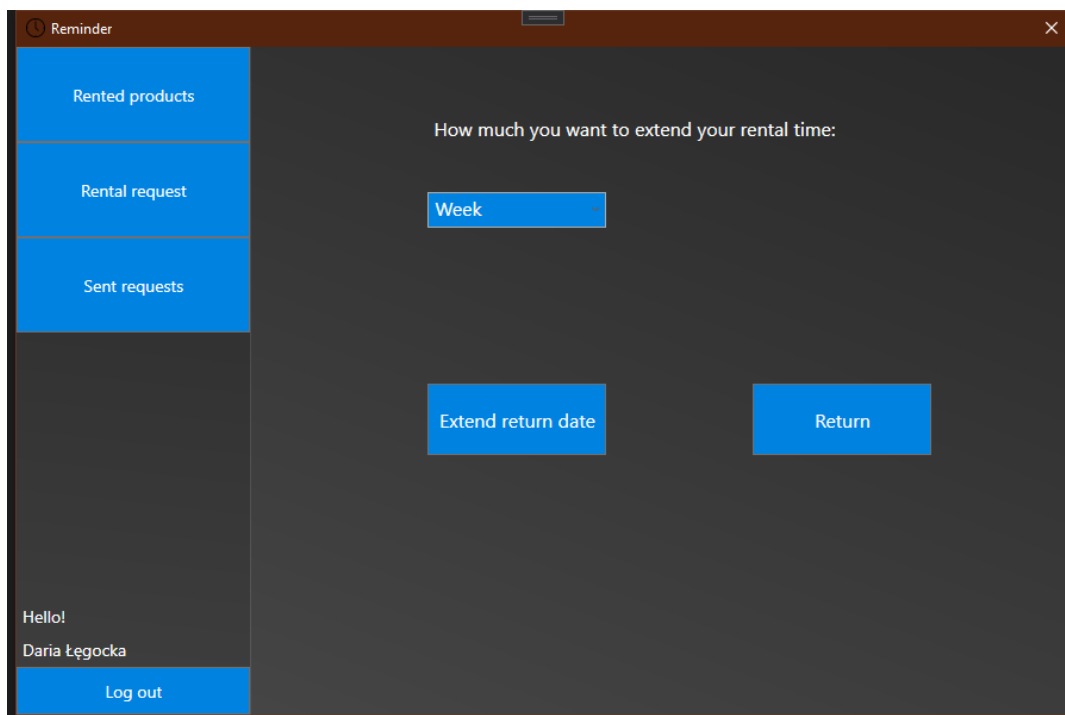
Rys. 6.4. Strona informacyjna na temat wypożyczenia

Przejsie jak i przedluzenie moze zostac zablokwane, jezeli na dany produkt inny uzytkownik juz wyslal prosbe o wypozyczenie, wtedy uzytkownik produkt moze jedynie zwrócic (Rys. 6.5.).



Rys. 6.5. Odmowa przedluzenia produktu

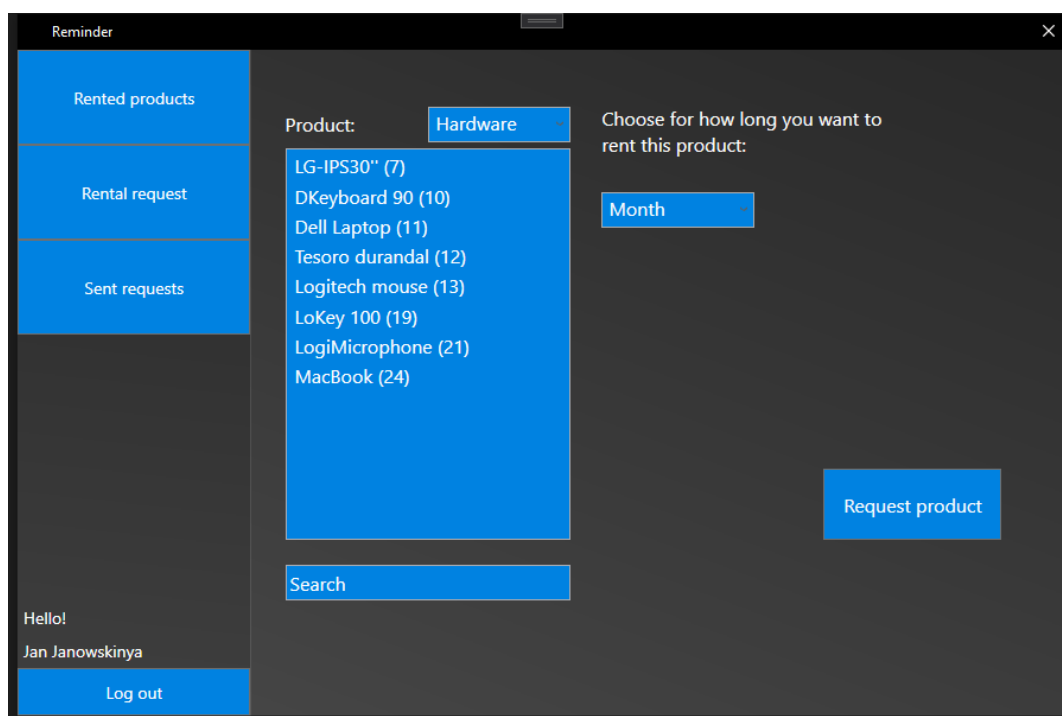
Jeżeli nikt nie oczekuje na dany produkt, aplikacja pokaże okno, w którym użytkownik ma opcję przedłużenia danego wypożyczenia (Rys. 6.6.). Okres przedłużenia można wybrać z listy gotowych, a jeżeli istnieje potrzeba wprowadzenia innej daty, po wyborze odpowiadającej opcji z listy, zostanie wyświetlony kalendarz z możliwością wyboru daty.



Rys. 6.6. Strona przedłużenia wypożyczenia

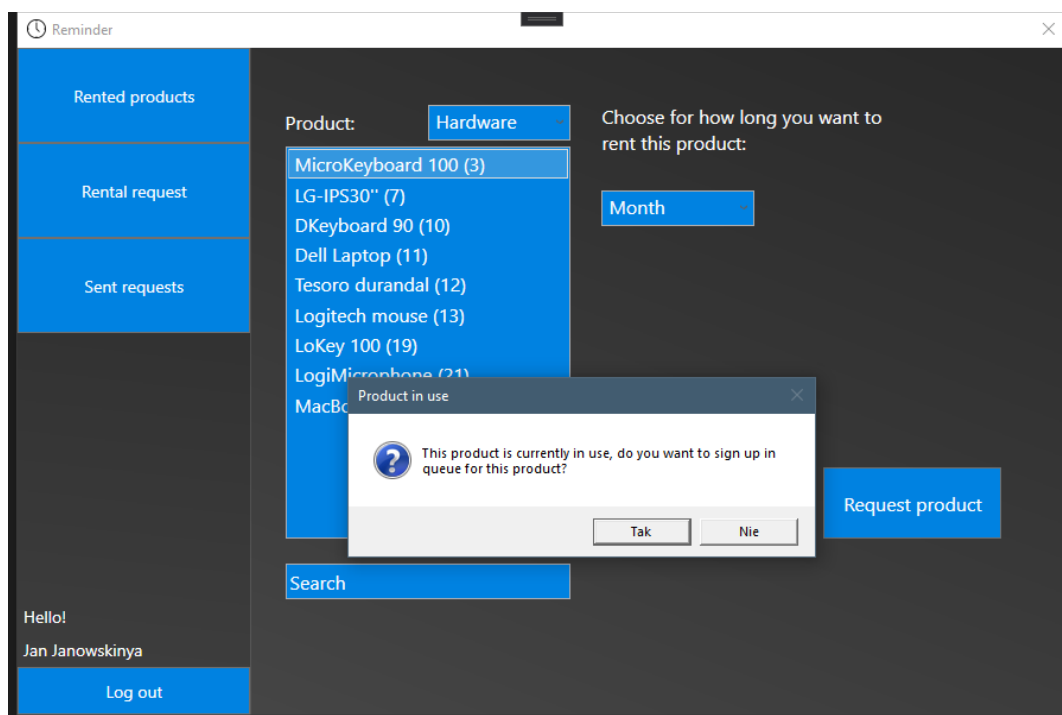
6.2.2. Wysyłanie prośby o produkt

W oknie wysyłania prośby o produkt (Rys. 6.7.), użytkownik może wybrać z listy produktów dostępnych w firmie, z wyłączeniem tych, na które prośba została już wysłana, ale nie została jeszcze rozpatrzona. Użytkownik podaje również wstępną datę do kiedy sprzęt planuje zwrócić, ponownie tak jak w przypadku przedłużania, może on wybrać z listy gotowych terminów albo z kalendarza po wybraniu opcji „Other” z listy.



Rys. 6.7. Strona wysyłania prośby o produkt

Zaimplementowany został również podział na produkty typu hardware oraz software jak i możliwość wyszukiwania produktu po nazwie przy pomocy pola search poniżej listy.

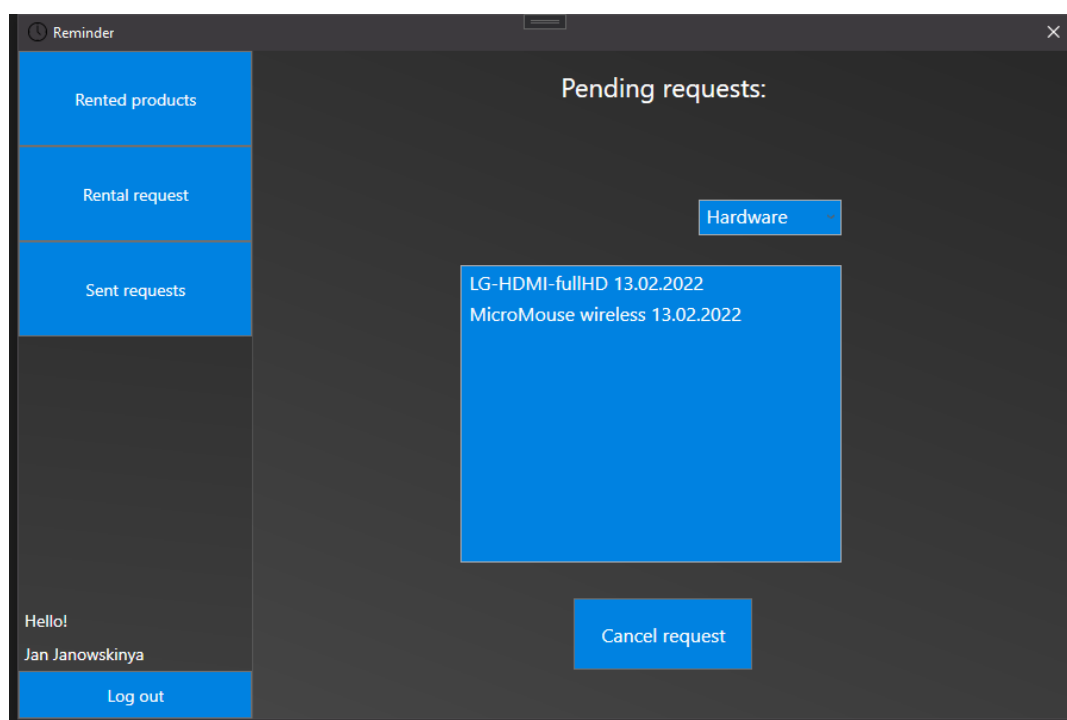


Rys. 6.8. Ostrzeżenie informujące o kolejce do produktu

W momencie, kiedy użytkownik będzie chciał wysłać prośbę o produkt, który nie jest obecnie dostępny i/lub oczekuje na niego już kolejka próśb od innych użytkowników, zostanie wyświetlone odpowiednie okno informujące o tym oraz dające wybór użytkownikowi czy ten dalej chce wysłać prośbę i wpisać się na kolejkę (Rys. 6.8.).

6.2.3. Wysłane prośby o produkty

Ostatnią możliwością użytkownika jest sprawdzenie już wysłanych przez niego próśb o produkt i ewentualne jego anulowanie (Rys. 6.9.). Lista jest podzielona na produkty typu hardware i software oraz mają zamieszczoną datę wysłania prośby obok nazwy produktu.



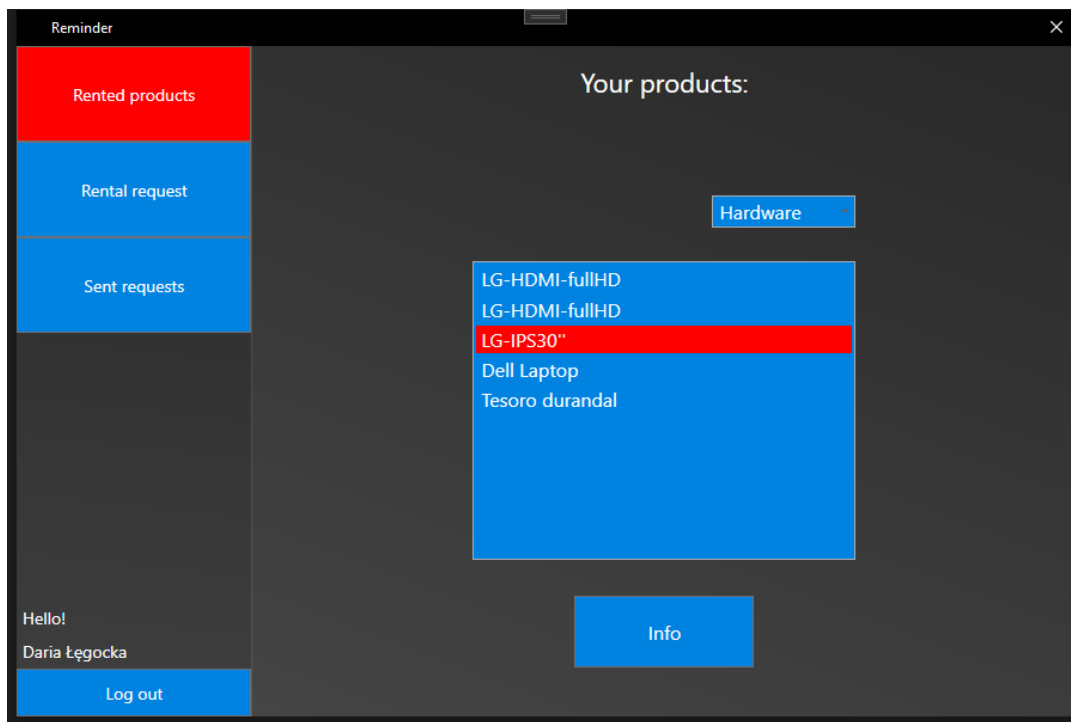
Rys. 6.9. Strona wysłanych próśb o produkt

6.2.4. System przypominania o terminie zwrotu

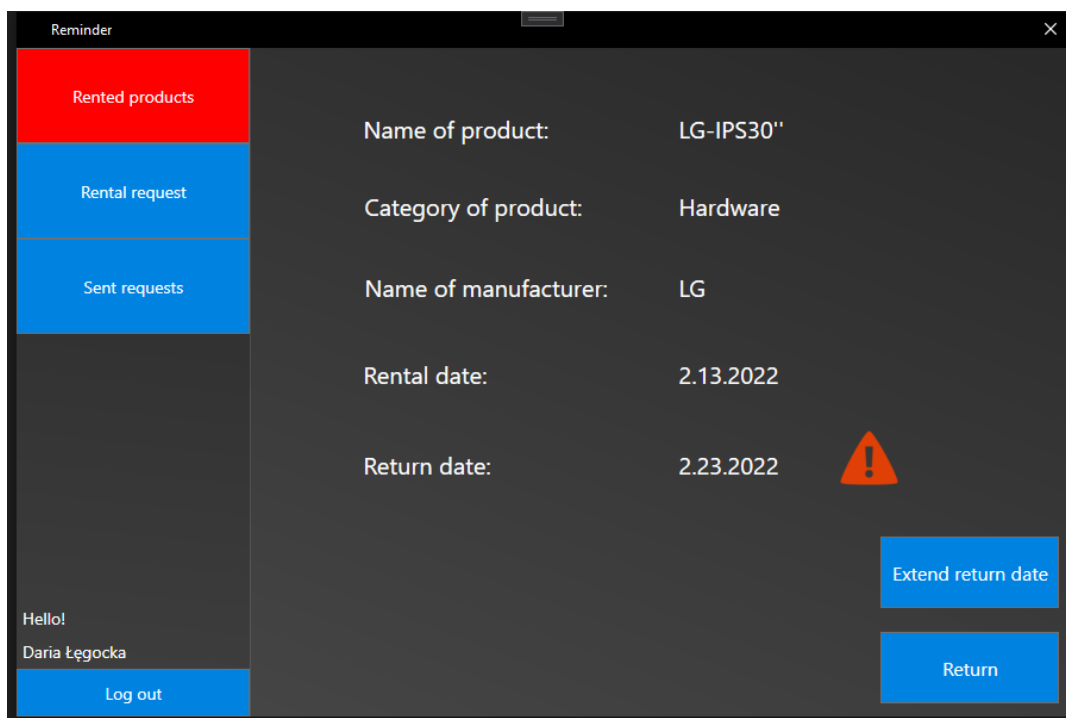
Okno użytkownika ma również zaimplementowaną funkcję wizualnego przypominania, jeżeli posiada on jakieś produkty, których termin zwrotu wypada w przeciągu tygodnia. Tymi wizualnymi wskaźnikami jest zaznaczony na czerwono przycisk okna wypożyczonych produktów jak i produkty, których termin oddania się

zbliża (Rys. 6.10.) oraz ikonę ostrzegającą w oknie informacyjnym na temat danego wypożyczenia (Rys. 6.11.).

Istnieje również system przypominający drogą mailową, aczkolwiek jest on obsługiwany po stronie administratora.



Rys. 6.10. Lista wypożyczonych produktów z ostrzeżeniem

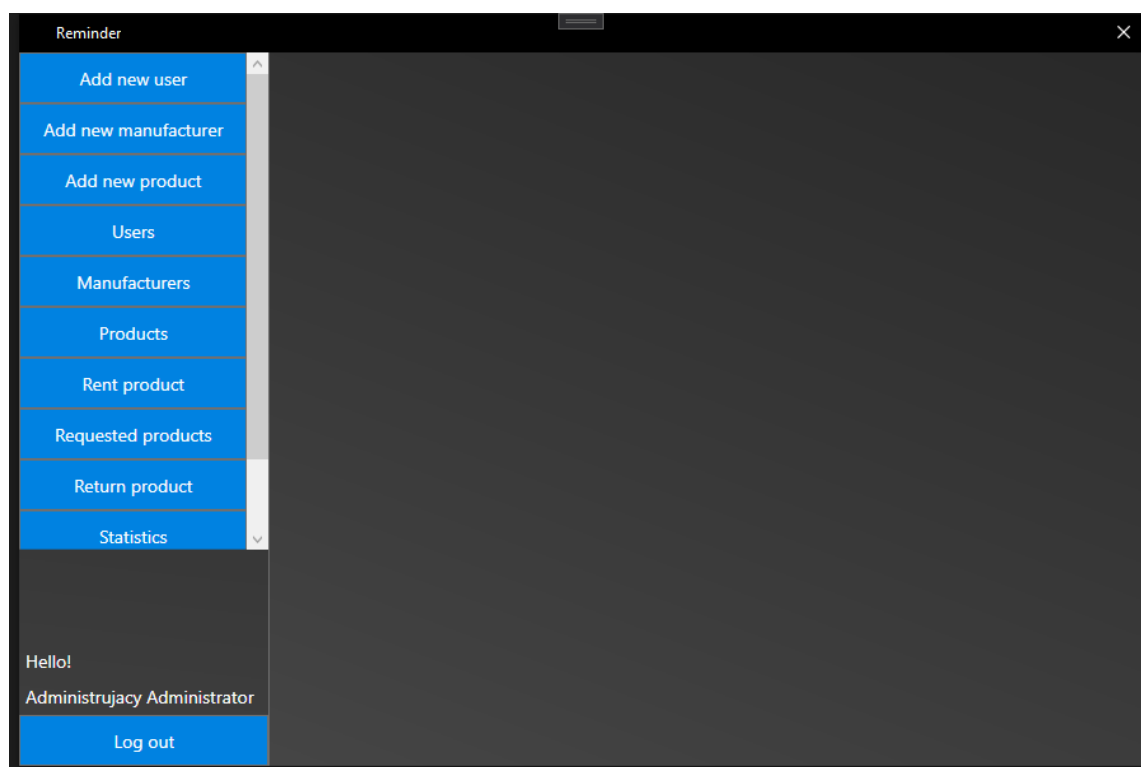


Rys. 6.11. Strona informacyjna z ostrzeżeniem

6.3. Część administratora

Po udanym zalogowaniu, kiedy system zobaczy, że id logującego się właśnie użytkownika należy do administratora, otworzy ono okno administratora (Rys. 6.12.). Okno działa tak samo jak w przypadku okna użytkownika z jedyną różnicą w postaci ilości przycisków menu.

Menu administratora można podzielić na 3 główne grupy: przyciski odpowiedzialne za wprowadzanie nowych rzeczy do systemu, przyciski odpowiedzialne za edytowanie znajdujących się już w systemie przedmiotów oraz te odpowiedzialne za system zarządzania sprzętem. Dwoma przyciskami nie znajdującymi się w powyższych grupach są te odpowiedzialne za statystyki oraz powiadomienia użytkowników poprzez mail.



Rys. 6.12. Okno administratora

6.3.1. Dodawanie nowych użytkowników lub producentów do systemu

Administrator ma możliwość wprowadzenia kolejnych nowych elementów: nowy użytkownik, nowy producent oraz nowy produkt. Dwie pierwsze opcje są do siebie

podobne z natury z różnicą leżącą jedynie w nazwie pul do wypełnienia jak i tablicy, w której zostanie umieszczony nowy rekord.

The image shows a web application window titled "Reminder". On the left is a sidebar menu with the following items: "Add new user", "Add new manufacturer", "Add new product", "Users", "Manufacturers", "Products", "Rent product", "Requested products", "Return product", and "Statistics". Below the menu, it says "Hello! Administrujący Administrator" and has a "Log out" button. The main area contains a form for adding a new user with the following fields: "Name", "Surname", "Login", "Password", "Country" (a dropdown menu), "Zip code", "Address", "2nd address", "Phone number", and "E-mail". There is a checkbox labeled "Administrator" and an "Accept" button at the bottom right.

Rys. 6.13. Strona wprowadzania nowego użytkownika

Na rysunku 6.13. widać zainicjowany proces dodawania nowego użytkownika. Poza podstawowymi informacjami w postaci imienia, nazwiska czy też kontaktu, administrator ustawia również login i hasło, którymi użytkownik będzie się logował. Istnieje również możliwość zaznaczenia opcji, że tworzony użytkownik posiada prawa administratora.

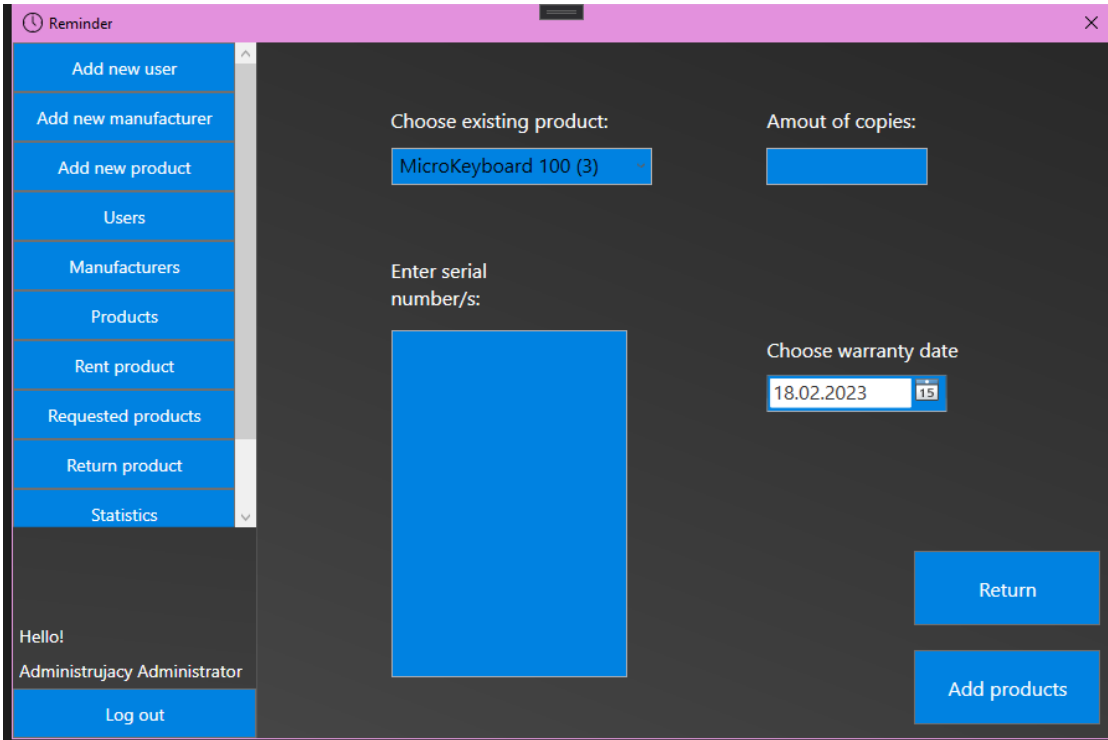
Wyróżniającym się elementem jest combobox na informację dotyczącą kraju z jakiego pochodzi dany użytkownik. System pozwala administratorowi na wybranie obecnych już w bazie danych krajów z listy, a w przypadku, kiedy takowy jeszcze nie został w niej zapisany, może on ręcznie wprowadzić nazwę nowego kraju, który zostanie dodany do list na przyszłość.

Okno dodania nowego producenta różni się jedynie brakiem pola na login oraz hasło, zamianą imienia i nazwiska na pole nazwa i brak oznaczenia administratora.

6.3.2. Dodanie nowych produktów do systemu

Na wstępie po kliknięciu przycisku wyświetlona zostanie strona, z której administrator wybiera czy chce dodać nowe egzemplarze do istniejących już produktów czy może chce dodać zupełnie nowy produkt, którego firma jeszcze nie posiada. Po wybraniu, załadowana zostanie odpowiednia strona.

W oknie dodawania egzemplarzu do istniejącego produktu (Rys. 6.14.) administrator wybiera do jakiego produktu chce dodać nowe kopie, musi podać faktyczną ilość nowych egzemplarzy jak i numer seryjny każdego z nich oraz datę wygaśnięcia gwarancji lub licencji w przypadku, kiedy produktem jest oprogramowanie. Po zatwierdzeniu nowe egzemplarze zostają dodane do bazy i użytkownicy mogą je wypożyczyć.



Rys. 6.14. Strona dodania nowych egzemplarzy

W przypadku wprowadzania zupełnie nowego produktu, oprócz informacji wymienionych wyżej, dochodzą informacje wspólne dla każdego egzemplarzu, które definiują produkt sam w sobie (Rys. 6.15.).

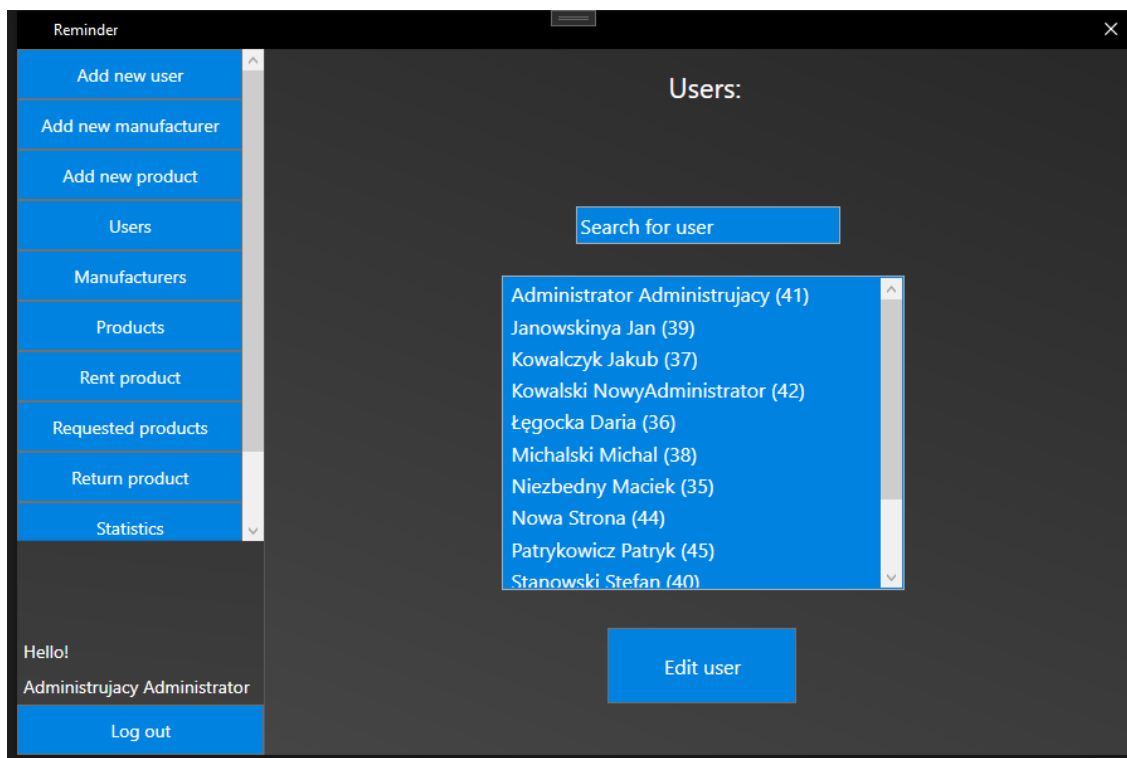
Rys. 6.15. Dodawanie nowego produktu

Nowymi polami do wypełnienia są do wyboru rodzaj produktu, ich wstępny stan, cena produktu oraz jego producent. Z racji potrzeby informacji kontaktowych do producenta, administrator może jedynie wybrać z listy już stworzonych, jeżeli produkt jest od kogoś, kogo nie ma jeszcze zapisanego w bazie danych, należy w pierwszej kolejności dodać go w oknie nowego producenta.

Jeżeli kategoria zostanie zmieniona na software, pojawi się nowe okno z informacją na jaki system operacyjny jest ono przeznaczone oraz data gwarancji zamieni się na datę wygaśnięcia licencji.

6.3.3. Edytowanie istniejących użytkowników lub producentów w systemie

Kolejna grupa przycisków jest odpowiedzialna za modyfikowania istniejących już w bazie danych produktów, ale również użytkowników jak i producentów. Pierwszym krokiem jest wybranie z listy elementu, który administrator chce zmodyfikować (Rys. 6.16.). Wygląd list jest podobny bez względu na to jaka rzecz jest modyfikowana.



Rys. 6.16. Strona wyboru użytkownika do edycji

Okno zawiera również wyszukiwarkę pozwalającą łatwiej znaleźć obiekt do edycji. Przy nazwach podane są numery id w celu rozróżnienia w przypadku podobieństw, np. dwóch pracowników o tym samym imieniu i nazwisku, jak i w celu ułatwienia komunikacji z bazą danych.

Po wybraniu użytkownika lub producenta do edycji, otwarte zostanie to samo okno jak w rysunku 6.13. (w przypadku modyfikowania użytkownika) z różnicą, że pola będą wypełnione informacjami od danego użytkownika. Jest to możliwe dzięki systemowi sesji oraz ustawionym polem `_EditID`. W momencie załadowania tej strony, aplikacja sprawdza czy ów pole jest równe 0 – jest to nowy użytkownik, czy też ma numer, który oznacza w tym przypadku id użytkownika do modyfikacji (Listing 6.5.). Jeżeli jest to drugi przypadek, uruchamiana jest metoda ładująca informacje do pul oraz wyświetlany jest przycisk umożliwiający powrót do listy, z której użytkownik został wybrany.

```

private void PageLoaded(object sender, RoutedEventArgs e)
{
    ReturnButton.Visibility = Visibility.Hidden;
    SqlConnection sqlConnection = connection.Connection;
    connection.OpenConnection(sqlConnection);

    CountryComboBox.ItemsSource = assistant.LoadCountries(sqlConnection);

    if (session.EditID != 0)
    {
        ReturnButton.Visibility = Visibility.Visible;

        LoadUser(sqlConnection);
    }
    connection.CloseConnection(sqlConnection);
}

```

Listing 6.5. Załadowanie strony

Rozróżnienie przy pomocy EditID występuje jeszcze w przypadku kliknięciu przycisku dodającego nowego użytkownika. Jeżeli pole jest równe 0, zostanie wywołana metoda wprowadzająca nowego użytkownika do bazy danych od podstaw, jeżeli inna niż 0, metoda aktualizująca istniejące już rekordy (Listing 6.6.).

```

private void AddNewUserButton_Click(object sender, RoutedEventArgs e)
{
    SqlConnection sqlConnection = connection.Connection;
    connection.OpenConnection(sqlConnection);

    int errorCode;

    errorCode = FillInChecker();

    if(errorCode == 0)
    {
        if (session.EditID != 0)
        {
            UpdateUser(sqlConnection);
        }
        else
        {
            CreateUser(sqlConnection);
        }
    }
    connection.CloseConnection(sqlConnection);
}

```

Listing 6.6. Rozróżnienie między nowym użytkownikiem, a aktualizacją istniejącego

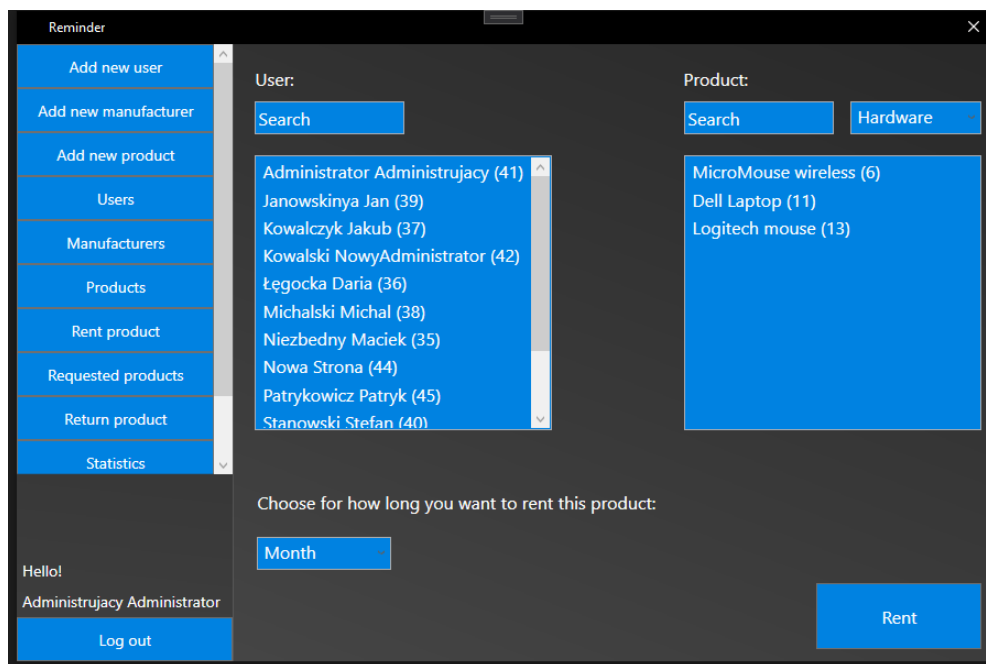
6.3.4. Edytowanie istniejących produktów w systemie

Proces wydłuża się o jedną dodatkową stronę w przypadku modyfikowania produktów. Wpierw ładowana jest strona pozwalająca na modyfikację produktu (Rys. 6.17.). Można w nim edytować nazwę, producenta jak i cenę. Obecna jest również lista wszystkich egzemplarzy danego produktu, wylistowanych według numerów seryjnych z możliwością wyszukiwania oraz doprecyzowaniem numery jakich egzemplarzy mają zostać wyświetlone: obecnie używanych, wolnych czy też uszkodzonych. Po wybraniu i kliknięciu przycisku Edit, otworzy się strona z załadowanymi wartościami tak jak w przypadku użytkowników czy producentów.

Rys. 6.17. Strona modyfikacji produktu

6.3.5. Wypożyczanie produktu

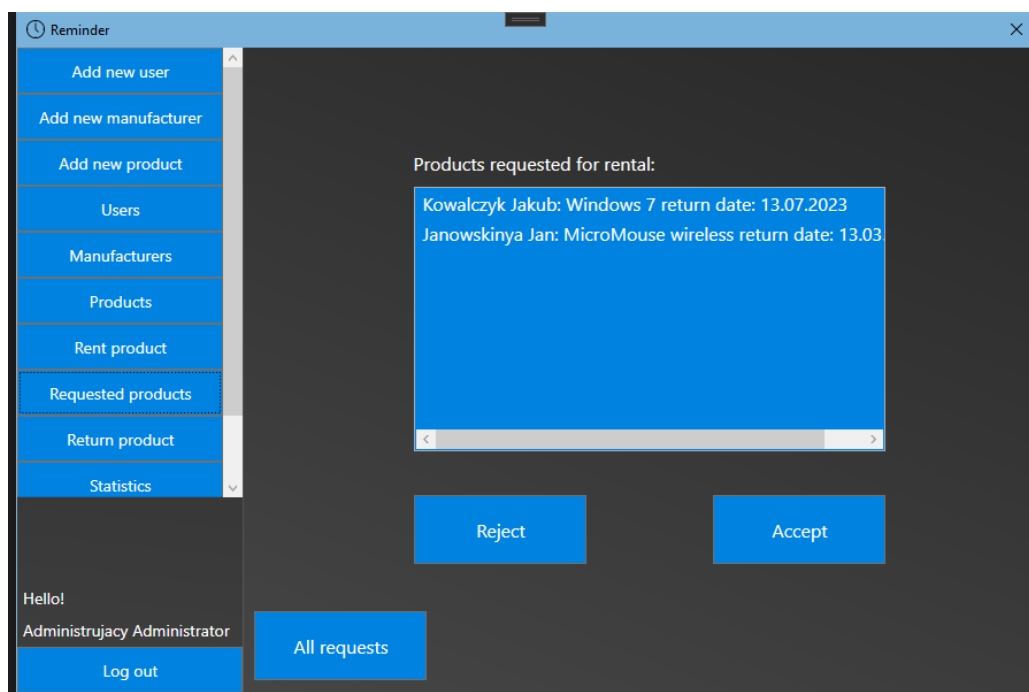
Strona wypożyczenia produktu składa się z dwóch kolumn, z których administrator może wybrać użytkownika jakiemu chce wydać produkt jak i listę samych produktów (Rys. 6.18.). Na liście znajdują się jedynie te produkty, które mają jeszcze dostępne egzemplarze do wypożyczenia. Obydwie listy mają swoją własną wyszukiwarkę. Przed wypożyczeniem, administrator musi również wyznaczyć wstępną datę zwrotu towaru



Rys. 6.18. Strona wypożyczenia produktu

6.3.6. Prośby o wypożyczenie produktu

W oknie prośb o produkty, administrator ma przedstawioną listę z imionami, nazwiskami oraz nazwami produktów, na które użytkownik wysłał prośbę (Rys. 6.19.).



Rys. 6.19. Strona prośb o produkty

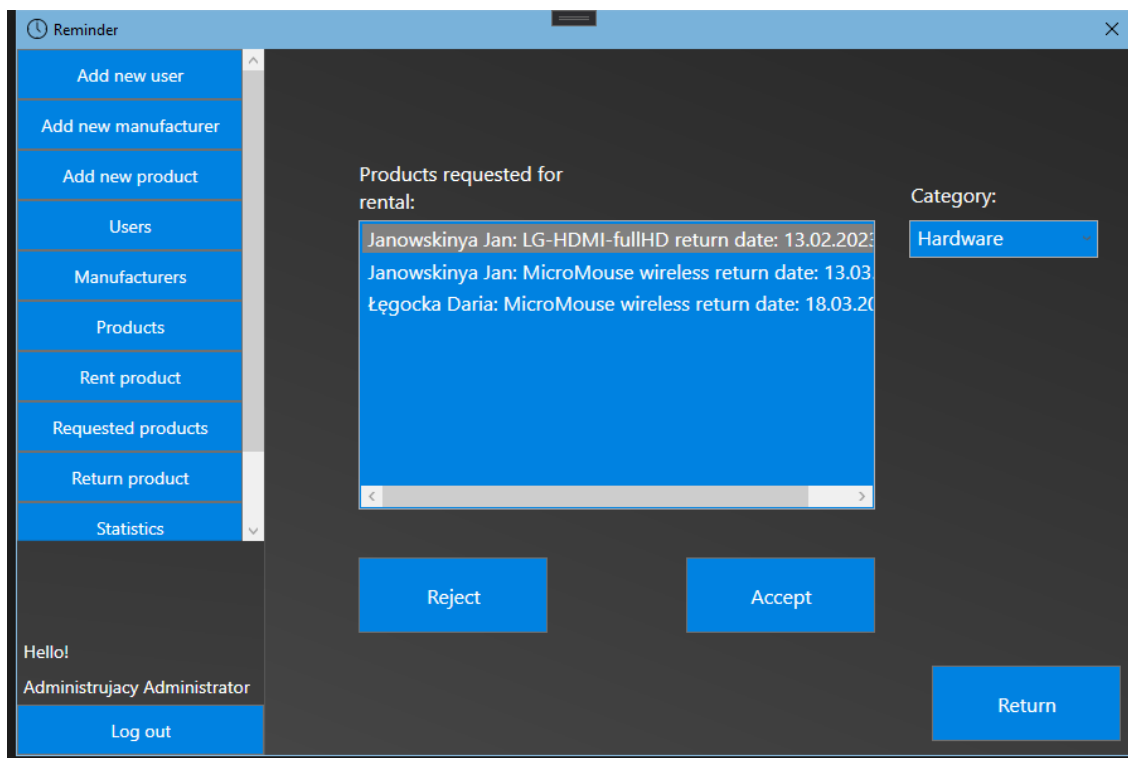
Na liście znajdują się jedynie te prośby, które są na produkty, które posiadają wolne egzemplarze oraz są one najwcześniej złożone. W przypadku, kiedy dwóch użytkowników wysła prośbę o ten sam produkt, na tej liście wyświetlony zostanie jedynie prośba od użytkownika, który zgłosił się pierwszy (Listing 6.7.).

Administrator w tym momencie ma możliwość albo odrzucenia zgłoszenia i usunięcia go z kolejki oczekujących prośb albo zatwierdzenie wypożyczenia i wydanie pracownikowi sprzętu.

```
SELECT U.name, U.surname, P.name as product, R.request_return_date FROM Requests as R
JOIN Users as U ON U.user_id = R.user_id
JOIN Products as P ON P.product_id = R.product_id
JOIN (
SELECT product_id, min(request_date) as MinDate FROM Requests WHERE status = 'Sent'
group by product_id
) tm ON tm.product_id = R.product_id AND R.request_date = tm.MinDate
WHERE P.copies_available > 0 AND R.status = 'Sent' ORDER BY R.request_id"
```

Listing 6.7. Zapytanie zwracające najstarszą prośbę

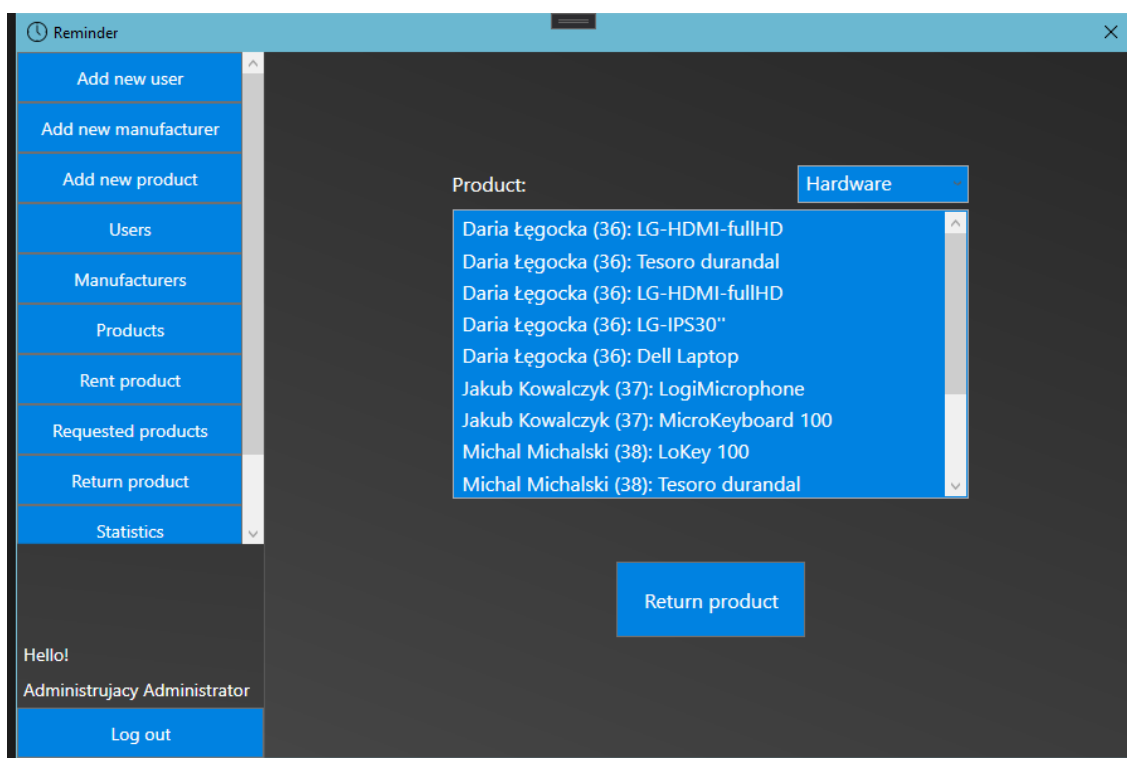
Istnieje również możliwość zobaczenia wszystkich oczekujących prośb od użytkowników. Po kliknięciu w odpowiedni przycisk, załadowana zostanie strona z listą, w której pokazane są wszystkie prośby na ten sam produkt, jak i te na produkt, który nie ma wolnych egzemplarzy (Rys. 6.20.). Prośby te są odpowiednio oznaczone i administrator nie jest w stanie zaakceptować takowej prośby, jedynie może ją odrzucić. Istnieje również podział na produkty typu hardware oraz software.



Rys. 6.20. Strona wszystkich próśb użytkowników

6.3.7. Zwrot produktu

Ostatnią stroną związaną z zarządzaniem sprzętem jest zwrot sprzętu. Na tej stronie administrator widzi listę wszystkich wypożyczonych produktów wraz tym informacją, kto posiada obecnie ten produkt (Rys. 6.21.). Po wybraniu wypożyczenia, administrator może oznaczyć takowe jako zwrócone co zwolni egzemplarz jak i oznaczy wypożyczenie za zwrócone.



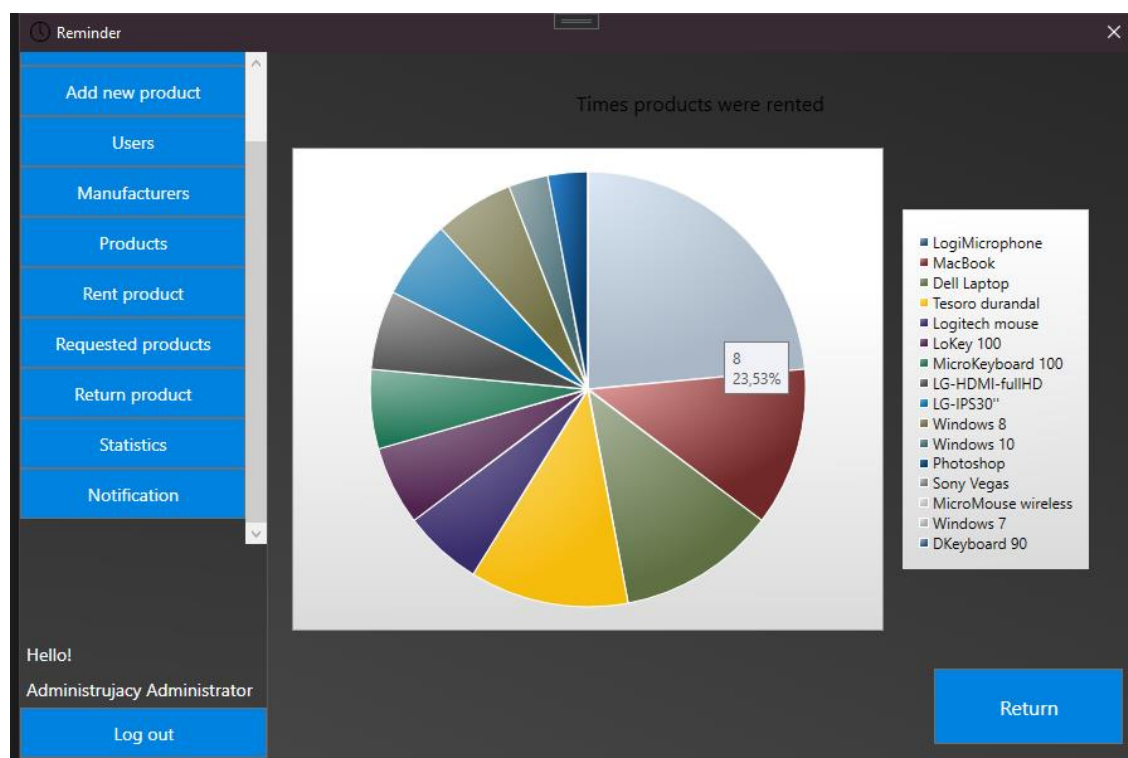
Rys. 6.21. Strona zwrotu produktu

6.3.8. Statystyki

Administrator ma możliwość wejścia w stronę ze statystykami tworzonymi na podstawie informacji zbieranych w bazie danych i zobaczeniu ich w formie wykresów. Obecnie w aplikacji są 4 rodzaje przygotowanych wykresów:

- Produkty, na które użytkownicy najczęściej wysyłają prośbę o wypożyczenie
- Produkty, które najczęściej są wypożyczane
- Średni czas oczekiwania pomiędzy wysłaniem prośby o wypożyczenie, a podjęciem decyzji administratora o zatwierdzenie lub odrzucenie prośby, grupowane względem produktów
- Średni czas trwania pojedynczego wypożyczenia danego produktu

Na rysunku 6.22. widnieje wykres najczęściej wypożyczanych przedmiotów. Statystyka jest przedstawiona w postaci diagramu kołowego przedstawiającego każdy produkt oraz to, ile raz był wypożyczany i jak odnosi się to procentowo do wszystkich wypożyczeń.



Rys. 6.22. Najczęściej wypożyczone przedmioty

Wykres tworzony jest za pomocą dodatkowego pakietu o nazwie WPF Toolkit Data Visualization Controls [9]. Po dodaniu odpowiedniej przestrzeni nazw (Listing 6.8.) oraz dodaniu samego wykresu z odpowiednimi parametrami w samym kodzie XML (Listing 6.9.), aplikacja następnie w klasie ładuje listę par klucz-wartość wypełnionymi danymi z bazy danych (Listing 6.10.).

```
xmlns:chartingToolkit="clr-
namespace:System.Windows.Controls.DataVisualization.Charting;assembly=DotNetProjects.
DataVisualization.Toolkit"
```

Listing 6.8. Przestrzeń nazw w kodzie XML

```
<chartingToolkit:Chart Name="pieChart" Title="Times products were rented"
VerticalAlignment="Top" Margin="10,10,10,0" Height="475" BorderThickness="0"
Foreground="Black">
<chartingToolkit:PieSeries DependentValuePath="Value" IndependentValuePath="Key"
ItemsSource="{Binding}" IsSelectionEnabled="True"/>
</chartingToolkit:Chart>
```

Listing 6.9. Kod XML wykresu

```

private void showPieChart()
{
    SqlConnection sqlConnection = connection.Connection;
    connection.OpenConnection(sqlConnection);
    List<KeyValuePair<string, int>> valueList = new List<KeyValuePair<string,
int>>();

    SqlCommand command = new SqlCommand("SELECT name, times_rented FROM Products
ORDER BY times_rented DESC", sqlConnection);
    SqlDataReader reader = command.ExecuteReader();

    if (reader.HasRows)
    {
        while(reader.Read())
        {
            valueList.Add(new KeyValuePair<string,
int>(reader["name"].ToString(), (int)reader["times_rented"]));
        }
    }

    reader.Close();

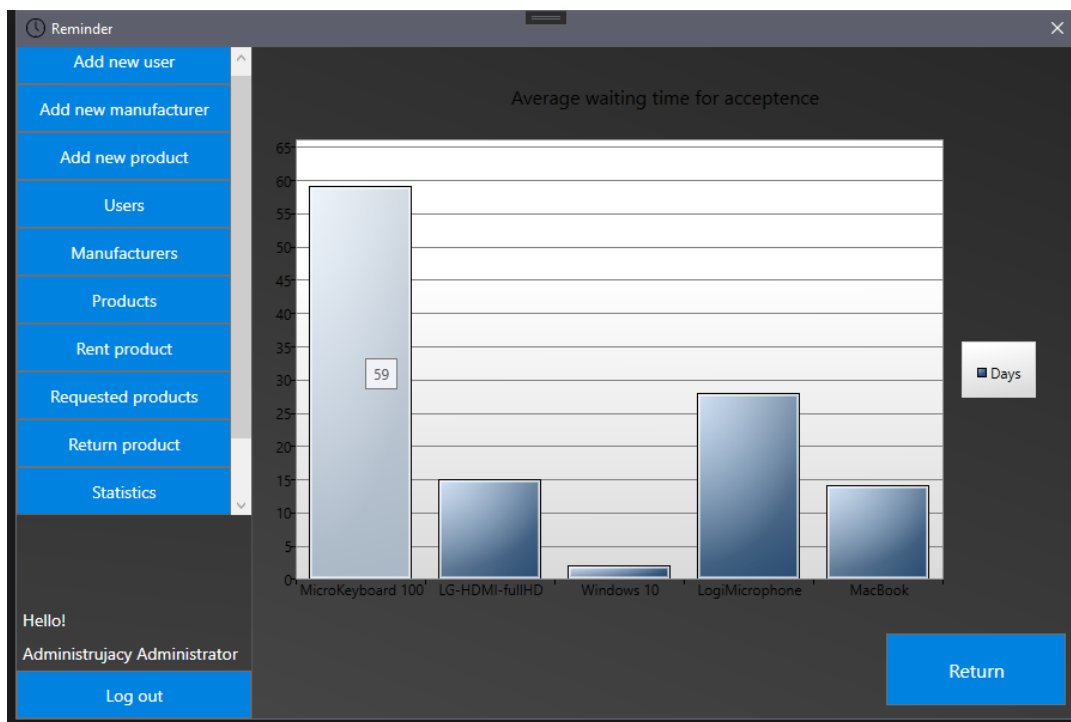
    pieChart.DataContext = valueList;

    connection.CloseConnection(sqlConnection);
}

```

Listing 6.10. Wypełnienie listy klucz-wartość danymi z bazy danych

Drugim przykładem jest wykres słupkowy średniego czasu oczekiwania na podjęcie decyzji administratora względem prośby o wypożyczenie (Rys. 6.23.). Kod XML oraz tworzenie listy par klucz-wartość przebiega w sposób identyczny z jedyną różnicą będącą samo zapytanie SQL (Listing 6.11.).



Rys. 6.23. Średni czas oczekiwania na podjęcie decyzji

```
SELECT P.name, AVG(DATEDIFF(d, request_date, request_check_date)) AS DIFF
FROM Requests as R
JOIN Products as P
ON P.product_id = R.product_id
WHERE R.status = 'Accepted' GROUP BY R.product_id, P.name
```

Listing 6.11. Zapytanie SQL do średniego czasu oczekiwania na decyzję

6.3.9. Proces powiadomień mailowych

W aplikacji został jeszcze zaimplementowany system powiadomień użytkowników o zbliżającym się terminie oddania produktu. Za każdym razem, kiedy administrator zaloguje się na konto, aplikacja sprawdza czy nie ma użytkowników wpisujących się w kryterium wysłania ostrzeżenia. Jeżeli system takowych znajdzie, automatycznie wysyła wiadomość mailową ze skrzynki obecnie zalogowanego administratora na skrzynkę użytkownika. Proces ten jest jednorazowy i ponowne logowania nie wywołają kolejnych maili. Do pełnego funkcjonowania systemu, użyta została przestrzeń nazw System.Net.Mail [10].

Aplikacja wpierv pobiera listę id użytkowników, których termin zwrotu opływa dokładnie w przeciagu 7 dni i zapisuje je do listy. Następnie pobierany jest adres mail administratora i hasło do poczty z bazy danych. Następnie tworzony jest obiekt będący wiadomością mailową oraz obiekt zdalnego połączenia z pocztą (Listing 6.12.). Następnie pobierany jest adres mailowy każdego użytkownika, którego id znalazło się na liście i aplikacja wysyła na takowy adres wiadomość.

```
var message = new MailMessage();
message.From = new MailAddress(adminMail, "Administrator");
message.Subject = "Return date of rented product";
message.Body = "Your rented product(s) have their return date in next week. If nobody
is waiting for it, you can extend rental time.";

var smtp = new SmtpClient("smtp.gmail.com");
smtp.UseDefaultCredentials = false;
smtp.Credentials = new NetworkCredential(adminMail, adminPass);
smtp.EnableSsl = true;
smtp.Port = 587;
```

Listing 6.12. Tworzenie obiektu wiadomości mailowej oraz obsługi zdalnej poczty

7. Podsumowanie

Celem pracy było zaprojektowanie i wykonanie aplikacji, która pomoże potencjalnym firmom w ułatwieniu zarządzania posiadanym sprzętem i lepszą kontrolę nad śledzeniem do kogo on trafia. Użyta technologia w postaci .NET WPF, C# oraz MS SQL Server okazały się być przyjemnym połączeniem, a w połączeniu z dedykowanym środowiskiem w postaci Visual Studio, tworzenie aplikacji nawet dla osób niedoświadczonych jest u podstaw łatwa do przyswojenia, ustawienia nowych kontroltek w kolejnych stronach to kwestia przeciągnięcia elementu z przybornika i umiejscowienie w dowolnym miejscu. Wszystkie wydarzenia związane z elementami aplikacji można wybrać z listy dzięki czemu nawet jak nie zna się danej kontrolki, od razu widać jakie wydarzenia może ona obsłużyć.

Niestety, czasem można napotkać problemy związane z tym jak rozwinięty jest ten system. Podczas gdy ustawienie automatycznego settera na niektóre elementy to kwestia 3 linijek kodu, są kontrolki o wiele bardziej złożone, w których użytkownik, aby zmienić styl, musi zmodyfikować odpowiedni parametr w arkusz stylu na kilka set linijek kodu, co może dla mniej doświadczonych programistów okazać się problematyczne z racji na ilość zmiennych w takowym arkuszu.

Nieodzownym programem również jest SQL Server Management Studio, dzięki któremu jakiekolwiek operacje na bazie danych, począwszy od stworzenia takowej, dodawaniu i edytowaniu tabel na możliwości ręcznego edytowania poszczególnych rekordów kończąc, wykonywało się bardzo prosto przy nawet minimalnej znajomości języka SQL. Obsługuje on również język T-SQL dzięki czemu idealnie współgra się z aplikacjami pisanymi na platformie .NET.

Jednakże darmowa odmiana MS SQL Server również nie posiada wszystkich opcji w porównaniu z płatną wersją. W aplikacji odczuć to można przy systemie powiadomień mailowych, który można by przeprowadzić w pełni po stronie bazy danych korzystając z SQL Agent'a i jego schedule jobs jednak nie jest on dostępny w darmowej wersji, stąd rozwiązanie po stronie aplikacji.

Aplikację udało się ukończyć, a cel pracy osiągnąć pomimo bardzo ograniczonego doświadczenia w użytych technologiach na początku projektu, co tylko utwierdza w przekonaniu jak przyjaznym i łatwym w obsłudze środowiskiem jest platforma .NET, system WPF jak i sam język C#.

Jeżeli chodzi o potencjalny rozwój aplikacji, najważniejszą zmianą byłoby przeniesienie bazy danych z pliku lokalnego na postawiony już w firmie serwer, dzięki temu każdy będzie miał zdalny dostęp do aplikacji w biurze o ile będzie podłączony do sieci, a wprowadzane zmiany będą dostępne zdalnie na serwerze aniżeli w pliku na jednej stacji. Drugą rzeczą byłoby przeniesienie pewnych funkcjonalności aplikacji na poziom samej bazy danych przy użyciu wyzwalaczy czy też procedur składowanych dzięki czemu zmniejszy się ilość kodu w samej aplikacji oraz ilość operacji na połączeniu z bazą danych. Ostatnią potencjalną zmianą byłoby również wdrożenie kopii zapasowych. Dane nie są newralgiczne do pracy firmy aczkolwiek ich utrata spowoduje spore zamieszanie więc zapewnienie np. pełnej kopii zapasowej co 2 tygodnie z kopią różnicową wykonywaną na początku każdego dnia zapewniłoby pokrycie potencjalnych błędów.

Od strony aplikacji rozwinąć dokładniej można część statystyczną, aby administrator miał jeszcze większą możliwość dostosowywania produktów do potrzeb w firmie np. które przedmioty są najbardziej popularne ale najrzadziej dostępne, z którymi najczęściej są problemy techniczne, a z których można kompletnie zrezygnować z racji braku potrzeb i zainteresowania. Poszerzenie kategorii produktów, obecny podział na software oraz hardware może być niewystarczający w momencie kiedy firma będzie posiadać większą ilość produktów więc zwiększenie dokładności o podkategorie np. hardware -> laptopy potencjalnie będzie potrzebne, aby nie przeszukiwać listy kilkudziesięciu produktów jednocześnie.

Streszczenie

Celem pracy było stworzenie aplikacji wspomagającej zarządzanie sprzętem wypożyczanym pracownikom w firmie.

W części teoretycznej opisane zostały po krótku technologie użyte w projekcie w postaci platformy .NET, systemu WPF jak i MS SQL Server, jak i narzędzia wykorzystane do przygotowania aplikacji oraz bazy danych takich jak Visual Studio oraz SQL Server Management Studio.

W części praktycznej skupiono się na przedstawieniu funkcjonalności samej aplikacji jak i jej zaplecza w postaci kodu C#. Przedstawiono podział aplikacji na tryb użytkownika oraz administratora oraz możliwości poszczególnych trybów.

Słowa kluczowe: C#, .NET, WPF, SQL Server, bazy danych

Summary

Main goal was developing application that would help companies with organising rental system for their employees.

In the first part of this thesis, focus was put on used technologies such as .NET, WPF system and MS SQL Server as well as on programming tools used while developing this application like Visual Studio and SQL Server Management Studio for managing database.

The second part was used to showcase the general workflow and appearance of application and its backend written using C#. It also showed the difference between user part and administrator part with their own functionalities.

Keywords: C#, .NET, WPF, SQL Server, databases

Literatura

- [1] <https://docs.microsoft.com/en-us/dotnet/standard/clr>
- [2] <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview>
- [3] <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-architecture>
- [4] <https://docs.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-6.0>
- [5] <https://expressdb.io/sql-server-express-vs-localdb.html>
- [6] <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>
- [7] <https://docs.microsoft.com/en-us/dotnet/csharp/linq/>
- [8] <https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022>
- [9] <https://www.nuget.org/packages/DotNetProjects.WpfToolkit.DataVisualization/>
- [10] <https://docs.microsoft.com/en-us/dotnet/api/system.net.mail?view=net-6.0>

Spis rysunków

Rys. 5.1. Pełen schemat bazy danych.....	13
Rys. 5.2. Tabele utrzymujące część zarządzająco-kontaktową.....	14
Rys. 5.3. Tabele utrzymujące część sprzętowo-wypożyczeniową.....	15
Rys. 5.4. Tabela Rentals.....	17
Rys. 5.5. Tabela Products.....	19
Rys. 5.6. Tabela Product_copies	20
Rys. 5.7. Tabela Requests	21
Rys. 5.8. Tabela Contacts.....	22
Rys. 6.1. Okno logowania	25
Rys. 6.2. Okno użytkownika	28
Rys. 6.3. Lista wypożyczonych produktów	29
Rys. 6.4. Strona informacyjne na temat wypożyczenia	29
Rys. 6.5. Odmowa przedłużenia produktu	30
Rys. 6.6. Strona przedłużenia wypożyczenia	31
Rys. 6.7. Strona wysyłania prośby o produkt.....	32
Rys. 6.8. Ostrzeżenie informujące o kolejce do produktu	32
Rys. 6.9. Strona wysłanych próśb o produkt.....	33
Rys. 6.10. Lista wypożyczonych produktów z ostrzeżeniem	34
Rys. 6.11. Strona informacyjne z ostrzeżeniem	34
Rys. 6.12. Okno administratora.....	35
Rys. 6.13. Strona wprowadzania nowego użytkownika.....	36
Rys. 6.14. Strona dodania nowych egzemplarzy	37
Rys. 6.15. Dodawanie nowego produktu	38
Rys. 6.16. Strona wyboru użytkownika do edycji.....	39
Rys. 6.17. Strona modyfikacji produktu	41
Rys. 6.18. Strona wypożyczania produktu	42
Rys. 6.19. Strona próśb o produkty	42
Rys. 6.20. Strona wszystkich próśb użytkowników	44
Rys. 6.21. Strona zwrotu produktu.....	45
Rys. 6.22. Najczęściej wypożyczane przedmioty	46

Rys. 6.23. Średni czas oczekiwania na podjęcie decyzji	48
--	----

Spis tabel

Tabela 5.1 Kolumny oraz rekordy encji Product_categories	16
--	----

Spis listingów

Listing 5.1. Skrypt tworzący tabelę Rentals	18
Listing 5.2. Skrypt tworzący tabelę Products	19
Listing 5.3. Skrypt tworzący tabelę Product_copies.....	20
Listing 5.4. Skrypt tworzący tabelę Requests.....	22
Listing 5.5. Skrypt tworzący tabelę Contacts	23
Listing 6.1. Weryfikacja czy użytkownik istnieje	26
Listing 6.2. Konstruktor połączenia z bazą danych	26
Listing 6.3. Informacje połączeniowe z pliku App.config	26
Listing 6.4. Ustawienia stylu aplikacji.....	27
Listing 6.5. Załadowanie strony	40
Listing 6.6. Rozróżnienie między nowym użytkownikiem, a aktualizacją istniejącego	40
Listing 6.7. Zapytanie zwracające najstarszą prośbę	43
Listing 6.8. Przestrzeń nazw w kodzie XML.....	46
Listing 6.9. Kod XML wykresu	46
Listing 6.10. Wypełnienie listy klucz-wartość danymi z bazy danych	47
Listing 6.11. Zapytanie SQL do średniego czasu oczekiwania na decyzję	48
Listing 6.12. Tworzenie obiektu wiadomości mailowej oraz obsługi zdalnej poczty	49