



UNIWERSYTET ŁÓDZKI

WYDZIAŁ MATEMATYKI I INFORMATYKI

System ERP

Praca licencjacka napisana pod kierunkiem dra Marka Majewskiego

Autor:

Dominik Sobierański

Łódź, 2016

Spis treści

1.	Wstęp.....	3
2.	Założenia i cele projektu.....	4
3.	System ERP i jego historia.....	5
3.1.	Definicja systemu ERP.....	5
3.2.	Historia systemu ERP.....	6
3.2.1.	EOQ.....	6
3.2.2.	TPS.....	6
3.2.3.	MRP.....	6
3.2.4.	MRP II.....	7
3.2.5.	ERP.....	7
4.	Środowisko.....	8
5.	Dokumentacja techniczna.....	9
5.1.	Opis klas występujących w projekcie.....	9
5.2.	Klasa Function.....	12
5.3.	Zastosowane algorytmy.....	14
6.	Dokumentacja użytkownika.....	18
7.	Wnioski.....	29
8.	Bibliografia.....	30

1. Wstęp

Celem pracy jest stworzenie systemu, który pozwoli na integrację wszystkich funkcji oraz działów w firmie. Proces integracji polega na wykorzystywaniu wspólnej bazy danych dla jednego systemu, dzięki czemu firma ma możliwość posługiwania się tylko jednolitym zbiorem informacji.

Na początku pracy zdefiniowane są założenia jakie powinien spełniać program aby być w pełni użytecznym narzędziem. W rozdziale drugim przedstawiona jest charakterystyka systemu ERP oraz jego ewolucja na przestrzeni ostatnich 100 lat. Częścią poprzedzającą właściwą dokumentację jest krótki opis użytego środowiska m. in. baza danych, język programowania i biblioteki. Dwa następne rozdziały opisują program w sposób bardziej szczegółowy. Zaprezentowane są stworzone klasy, ze szczególnym uwzględnieniem biblioteki 'Function' oraz ważniejsze algorytmy. Druga część skupia się na stronie użytkowej aplikacji i pokazuje sposób pracy ze środowiskiem. Podsumowaniem całości dokumentacji są wnioski na temat założeń i osiągniętych celów oraz bibliografia.

2. Założenia i cele projektu

Jedno z wyzwań, przed jakim stoją przedsiębiorstwa to zagwarantowanie dostępu do poprawnych i aktualnych informacji. Jest to prawdopodobnie kluczowy czynnik w procesie podejmowania decyzji w przedsiębiorstwie. Często informacje w firmie są rozproszone w kilku bazach danych. Nie ma więc możliwości współdzielenia danych w zakresie jednego systemu, przez co korzystanie z nich jest znacznie utrudnione. W systemach zarządzania ERP nie napotyka się na tego rodzaju przeszkody, ponieważ głównym ich założeniem jest zgromadzenie danych we wspólnej bazie. Oznacza to, że jeśli jeden pracownik wprowadzi dane do systemu, to będą one widoczne dla wszystkich innych pracowników również w innych działach.

Następnym problemem często napotykanym wśród systemów starszej generacji jest fakt, że wprowadzanie informacji nie są dostępne w czasie rzeczywistym, lecz są aktualizowane raz na jakiś czas. ERP zakłada natychmiastową dostępność. Dodatkowo umożliwia przedsiębiorstwom prowadzenie księgowości, wspiera dystrybucję, serwis i produkcję.

System ERP modyfikuje się często w taki sposób, aby dopasować go do zapotrzebowania w danej firmie. Omawiany w tej pracy system jest dedykowany dla serwisu komputerowego zajmującego się naprawą, jak i dystrybucją sprzętu.

Kolejnym niezwykle istotnym założeniem jest autoryzacja dostępu. Funkcjonalność oprogramowania jest uzależniona od stanowiska i roli danej osoby w firmie. Na przykład, magazynier ma wgląd do stanu magazynowego, ale nie powinien mieć możliwości dodawania nowych pracowników.

Celem przedstawionej w tej pracy aplikacji jest wspomaganie efektywnego zarządzania całością zasobów przedsiębiorstwa poprzez wsparcie dystrybucji, serwisowania oraz zintegrowanie wszystkich działów i funkcji w firmie. Realizowane jest to poprzez wspólną bazę danych, do której dostęp mają wszyscy użytkownicy aplikacji. Wielodostępowość jest realizowana przez odpowiedni interfejs, który dodatkowo uaktualnia dane w czasie rzeczywistym.

Program jest dedykowany dla serwisu komputerowego, gdzie pracownicy dzielą się na cztery grupy: administracja, sprzedawcy, technicy i magazynierzy, więc zaimplementowano w nim cztery rodzaje autoryzacji. Każda z grup ma charakterystyczny zbiór uprawnień, które wykorzystuje na swoim stanowisku. Wpływa to w znacznym stopniu na bezpieczeństwo i pozwala efektywniej wykorzystywać zasoby. (zob. [1])

3. System ERP i jego historia

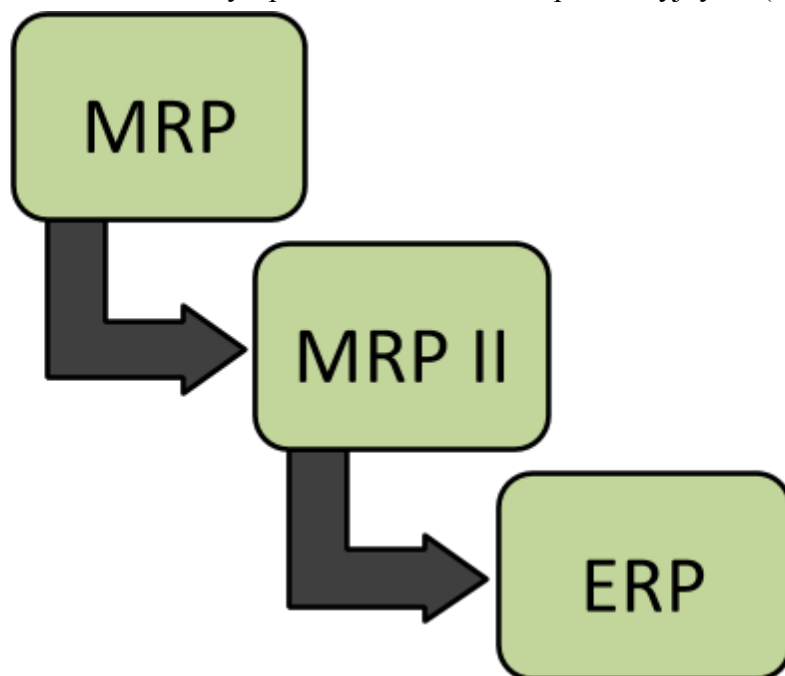
3.1. Definicja systemu ERP

System ERP służy wspomaganie zarządzania przedsiębiorstwem lub współdziałania grupy współpracujących ze sobą przedsiębiorstw poprzez gromadzenie danych oraz umożliwienie wykonywania operacji na zebranych danych. Wspomaganie to może obejmować wszystkie lub część szczebli zarządzania i ułatwia optymalizację wykorzystania zasobów przedsiębiorstwa oraz zachodzących w nim procesów.

Istnieje kilka rodzajów systemów klasy ERP:

- a) modułowe, które składają się ze współpracujących ze sobą, choć niezależnych od siebie aplikacji.
- b) zintegrowane, które korzystają z tej samej bazy danych i platformy biznesowej. Nie występuje w nich żadna wymiana informacji pomiędzy modułami, ponieważ wszystkie funkcjonalności korzystają w czasie rzeczywistym z tych samych zasobów.

System ERP jest rozwinięciem systemu MRP II, który z kolei jest rozwinięciem MRP (ang. Material Requirements Planning, planowanie zapotrzebowania materiałowego), wzbogaconym o moduł CRP czyli planowanie zdolności produkcyjnych. (zob. [2])



3.2. Historia systemu ERP

3.2.1. EOQ

W czasach gdy przemysł nie był zdominowany przez komputery (1913) używano metod takich jak EOQ (Economic Order Quantity). Służyły one do zarządzania zapasami w łańcuchu dostaw. Definiowały one optymalną wartość do zamówienia tak, aby zminimalizować koszty zmienne. (zob. [3])

3.2.2. TPS

Pierwszym poważnym systemem tego typu był wprowadzony pod koniec lat 40' Toyota Production System (TPS). Był on zbiorem niepowtarzalnych japońskich metod zarządzania. Obejmował zasady kultury przyjęte w korporacji ale również sposób prowadzenia działalności i postrzegania świata. Technologia ta była skoncentrowana na logistyce i organizacji produkcji oraz na pozytywnych relacjach z dostawcami i klientami. TPS był syntezą wielu zasad, koncepcji i technik mających na celu wyeliminowanie 3M (muri – nadwyreżenie i trudności, mura – nieregularność, muda – marnotrawstwo). (zob. [4])

3.2.3. MRP

Stworzony w 1964 przez Josepha Orlicky'ego MRP był odpowiedzią dla systemu Toyota Production System. Pierwszą firmą, w której został wprowadzony była Black & Decker. Z czasem liczba firm wzrosła do 700 w 1975 roku, a w 1981 roku do 8 000. Służył on do określania zapotrzebowania na zasoby materiałowe takie jak surowce, materiały, komponenty. Za jego pomocą można było precyzyjnie wykalkulować ilość materiałów i kalendarz dostaw w odpowiedni sposób. Pozwalało to sprostać ciągle zmieniającemu się na rynku popytowi na poszczególne produkty. (zob. [5])

Jego głównymi założeniami były:

- Redukcja zapasów.
- dokładne określenie czasów dostaw surowców i półproduktów.
- dokładne wyznaczenie kosztów produkcji.
- lepsze wykorzystanie posiadanej infrastruktury (magazynów, możliwości wytwórczych).
- szybsze reagowanie na zmiany zachodzące w otoczeniu.
- kontrola realizacji poszczególnych etapów produkcji.

3.2.4. MRP II

W 1983 na podstawie MRP powstał MRPII, który został rozbudowany względem poprzednika o możliwość planowania zdolności produkcyjnych (CRP) oraz o funkcjonalności związane z procesami sprzedaży i wspierającymi podejmowanie decyzji na szczeblach zarządzania produkcją. Oprócz materiałów związanych bezpośrednio z produkcją, MRP II uwzględniał także materiały pomocnicze, zasoby ludzkie, pieniądze, czas, środki trwałe i inne. (zob. [6])

3.2.5. ERP

W latach dziewięćdziesiątych systemy MRP i MRPII ewoluowały w system ERP, który jest zbiorem wielu modułów, a jego podstawowym elementem jest wspólna baza danych.

Indywidualne moduły systemu ERP obejmują następujące dziedziny:

- Zarządzanie zapasami.
- Magazynowanie.
- Planowanie produkcji.
- Śledzenie realizowanych dostaw.
- Sprzedaż.
- Zaopatrzenie.
- Księgowość.
- Zarządzanie relacjami z klientami.
- Zarządzanie zasobami ludzkimi (płace, kadry).
- Finanse.

W strukturę systemów ERP nierzadko wchodzi też inne moduły takie jak np. moduł zarządzania transportem, projektami czy controlling. Komponenty te mogą być wzajemnie niezależne od siebie co umożliwia dopasowanie oprogramowania do specyfiki poszczególnego przedsiębiorstwa. Systemy te często pozwalają też na ustalenie praw dostępu dla poszczególnych użytkowników. (zob. [7])

4. Środowisko

4.1. MySQL Workbench - rozwijane przez firmę Oracle wolnodostępne narzędzie administracyjne do zarządzania relacyjnymi bazami danych.

4.2. Netbeans - zintegrowane środowisko programistyczne przeznaczone dla języka Java. Jego głównym celem jest przyspieszenie projektowania i budowy aplikacji Java, w tym również aplikacji mobilnych oraz usług sieciowych.

4.3. iText - biblioteka dla programistów, która umożliwia tworzenie i manipulowanie dokumentów w formacie PDF, w poziomie języka Java.

4.4. Java DataBase Connectivity - interfejs umożliwiający aplikacjom napisanym w języku Java łączyć się z bazami danych za pośrednictwem języka SQL.

4.5. Java Development Kit (JDK) – oprogramowanie udostępniające środowisko niezbędne do programowania w języku Java.

4.6. JavaMail API – biblioteka umożliwiająca walidację adresu email.

5. Dokumentacja techniczna.

5.1. Opis klas występujących w projekcie:

5.1.1. Category - odpowiedzialna za przechowywanie w polu 'name' kategorii produktu. Pole id odnosi się do pola o tej samej nazwie w klasie Product. Zabieg ten pozwala na uniknięcie redundancji w bazie danych. Kategoryzowanie produktów pozwala na łatwiejsze ich wyszukiwanie, zamawianie i sprzedaż.

```
public class Category {  
    private int id;  
    private String name;  
}
```

5.1.2. Email - pozwala na tworzenie, wysyłanie i odbieranie wiadomości między użytkownikami systemu. Pole 'id_sender' zawiera id nadawcy, 'id_receiver' - id odbiorcy. 'Text' przechowuje treść wiadomości, 'date' - dokładną datę wysłania wiadomości. Wartość 'checked' informuje program czy dana wiadomość została już kiedyś odczytana.

```
public class Email {  
    private int id;  
    private int id_sender;  
    private int id_receiver;  
    private String text;  
    private Timestamp date;  
    private boolean checked;  
}
```

5.1.3. Employee - każdy obiekt tej klasy odwzorowuje jednego pracownika/użytkownika systemu przechowując jego dane takie jak imię, nazwisko, email, zaszyfrowane hasło oraz pozycję w firmie, która to z kolei jest pobierana z bazy za pośrednictwem kolumny 'id_position'. Użytkownik, logując się, informuje system o swojej obecności, a każdy jego ruch jest odnotowany za pośrednictwem pola 'id'.

```
public class Employee {  
    private int id;  
    private String name;  
    private String full_name;  
    private String email;  
    private String password;  
    private String position;
```

```
}
```

5.1.4. Log - dostarcza dokładniejszych informacji na temat realizowania zamówień. Dzięki niej użytkownik może się dowiedzieć, kiedy złożono zamówienie lub kiedy je przyjęto.

```
public class Log {  
    private int id;  
    private int id_object;  
    private Timestamp date;  
    private String action;  
}
```

5.1.5. Order - obiekty tej klasy służą głównie do przechowywania listy zamawianych produktów ('list'). Oprócz tego zawierają informację o tym, kto je zamówił ('id_employee'), kiedy ('date'), czy przyjęto dostawę ('executed') i ile wynosiły koszty ('price'). Informacja o produktach z listy jest pobierana z bazy jako obiekt typu String, który jest przetwarzany na listę za pomocą specjalnego algorytmu, który zostanie omówiony później.

```
public class Order {  
    private int id;  
    private int id_employee;  
    private Timestamp date;  
    private List<Product> list;  
    private boolean executed;  
    private Double price;  
}
```

5.1.6. PdfFiles – klasa, której zadaniem jest generowanie pliku. Pdf zawierającego fakturę w formie tabeli wraz z łączną ceną i datą. Zawiera ona następujące metody:

- public static void createPdf(java.util.List<Product> selling_list) - tworzy pełny dokument.
- public static void addTitlePage(Document document) - dodaje nagłówek do dokumentu, który zawiera m.in. bieżącą datę.
- private static void addEmptyLine(Paragraph paragraph, int number) - dodaje do paragrafu puste linie w ilości podanej w drugim argumencie.
- private static void createTable(Document document, java.util.List<Product> selling_list) - najważniejsza metoda, która generuje główną tabelę z produktami i cenę na podstawie podanej listy produktów.

5.1.7. Position - klasa podobna do Category z tą różnicą, że przechowuje pełną nazwę stanowiska pracownika, do którego pracownik odwołuje się za pomocą odpowiedniego pola.

```
public class Position {  
    private int id;  
    private String name;  
}
```

5.1.8. Product - klasa przechowująca dane na temat produktu takie jak m.in. nazwa ('name'), cena detaliczna ('retail_price'), VAT ('vat'), kategoria ('category') oraz ilość na magazynie ('quantity').

```
public class Product {  
    private int id;  
    private String name;  
    private Double retail_price;  
    private Double vat;  
    private String category;  
    private int quantity;  
}
```

5.1.9. Repair - obiekt tej klasy odzwierciedla złożone przez klienta zamówienie na naprawę sprzętu. Zawiera ono informację o pracowniku przyjmującym ('id'), imię i nazwisko klienta ('client_name', 'client_full_name'), opis problemu ('description'), datę złożenia zamówienia ('date') oraz informację o tym czy zamówienie zostało wykonane i jest gotowe do odbioru ('executed').

```
public class Repair {  
    private int id;  
    private int id_employee;  
    private String client_name;  
    private String client_full_name;  
    private String description;  
    private Timestamp date;  
    private boolean executed;  
}
```

5.1.10. Function - główna klasa, która jest swoistą biblioteką funkcyjną. Obiekt tej klasy gwarantuje dostęp do wszelkich niezbędnych metod używanych w aplikacji. Zawiera pole typu Connection, dzięki któremu konstruktor automatycznie tworzy połączenie z bazą danych wykorzystując do tego informacje zawarte w zewnętrznym pliku DBInfo.properties. Pola 'ALGORITHM' i 'keyValue' używane są w funkcjach szyfrujących.

```

public class Function {
    Connection myConn = null;
    private static final String ALGORITHM = "AES";
    private static final byte[] keyValue
        = new byte[]{'T', 'h', 'i', 's', 'I', 's', 'A',
'S', 'e', 'c', 'r', 'e', 't', 'K', 'e', 'y'};
}

```

5.2 Klasa Function

Do klasy Function należą następujące funkcje:

- **public String encrypt(String valueToEnc)** - szyfruje podany jako argument ciąg znaków.
- **public String decrypt(String encryptedValue)** - deszyfruje ciąg znaków.
- **private static Key generateKey()** - pomocnicza metoda tworząca klucz niezbędny do szyfrowania i odszyfrowywania.
- **private Employee convertRowToEmployee(ResultSet rs)** - przetwarza wynik zapytania do bazy na obiekt typu Employee.
- **public String getPosition(int index)** - zwraca nazwę stanowiska na podstawie podanego indeksu.
- **public List<Employee> getAllEmployees()** - pobiera z bazy wszystkich pracowników i zwraca ich w formie listy.
- **private Email convertRowToEmail(ResultSet rs)** - przetwarza wynik zapytania do bazy na obiekt typu Email.
- **public List<Email> getAllEmails()** - pobiera z bazy wszystkie wiadomości email i zwraca je w formie listy.
- **private Repair convertRowToRepair(ResultSet rs)** - przetwarza wynik zapytania do bazy na obiekt typu Repair.
- **public List<Repair> getAllRepairs()** - pobiera z bazy wszystkie formularze naprawy i zwraca je w formie listy.
- **private Product convertRowToProduct(ResultSet rs)** - przetwarza wynik zapytania do bazy na obiekt typu Product.
- **public String getCategory(int index)** - zwraca nazwę kategorii na podstawie podanego indeksu.
- **public List<Product> getAllProducts()** - pobiera z bazy wszystkie produkty i zwraca je w formie listy.
- **public List<Product> convertStringCodeToProductsList(String string)** - metoda, która przetwarza ciąg znaków na listę produktów. Dokładniejsze wyjaśnienie działania

algorytmu zostanie wyjaśnione w dalszej części.

- **private Product getProductById(int id)** - pobiera z bazy danych produkt o podanym id.
- **private Order convertRowToOrder(ResultSet rs)** - przetwarza wynik zapytania do bazy na obiekt typu Order.
- **public List<Order> getAllOrders()** - pobiera z bazy wszystkie zamówienia produktów i zwraca je w formie listy.
- **public Employee getEmployeeById(int id)** - pobiera z bazy danych użytkownika o podanym id.
- **public void fillComboboxWithEmployees(JComboBox<Employee> combobox)** - czyści comboboxa i wypełnia go pobranymi z bazy danych pracownikami.
- **public void addEmail(Employee temp)** - dodaje do bazy nową wiadomość mailową.
- **void fillComboboxWithEmployeesWithoutUser(JComboBox combobox, Employee user)** - czyści comboboxa i wypełnia go pobranymi z bazy danych pracownikami za wyjątkiem jednego podanego w argumencie.
- **public List<Email> getAllEmailsForUser(Employee user)** - pobiera z bazy wszystkie wiadomości email adresowane dla podanego użytkownika i zwraca je w formie listy.
- **public void addRepair(Repair temp)** - dodaje do bazy nowy formularz naprawy.
- **public void setEmailChecked(Employee email)** - oznacza podany email jako przeczytany.
- **public void fillTableWithEmailsForUser(JTable jTableMailbox, Employee user)** - wypełnia tabelę wiadomościami email adresowanymi do podanego użytkownika.
- **public boolean thereIsNewMail(Employee user)** - sprawdza czy dla podanego użytkownika istnieje nieprzeczytany jeszcze mail.
- **public void fillTableWithRepairs(JTable jTableRepairs)** - wypełnia podaną tabelę formularzami napraw.
- **public void setRepairExecuted(Repair temp)** - oznacza podany formularz naprawy jako wykonany.
- **public void fillTableWithAllProducts(JTable jTableProducts)** - wypełnia podaną tabelę wszystkimi produktami bez względu na kategorie.
- **public List<Product> getFoodProducts()** - pobiera z bazy produkty z kategorii 'Jedzenie' i zwraca je w formie listy.
- **public void fillTableWithFoodProducts(JTable jTableProducts)** - wypełnia podaną tabelę tylko produktami z kategorii 'Jedzenie'.
- **public List<Product> getPartsProducts()** - pobiera z bazy wszystkie produkty z wyjątkiem tych z kategorii 'Jedzenie' i zwraca je w formie listy.
- **public void fillTableWithPartsProducts(JTable jTableProducts)** - wypełnia podaną tabelę wszystkimi produktami z wyjątkiem tych z kategorii 'Jedzenie'.
- **public List<Product> searchPartsProducts(String name)** - zwraca listę Produktów z wyjątkiem tych z kategorii 'Jedzenie' i tych, które w nazwie zawierają podany wzorzec.

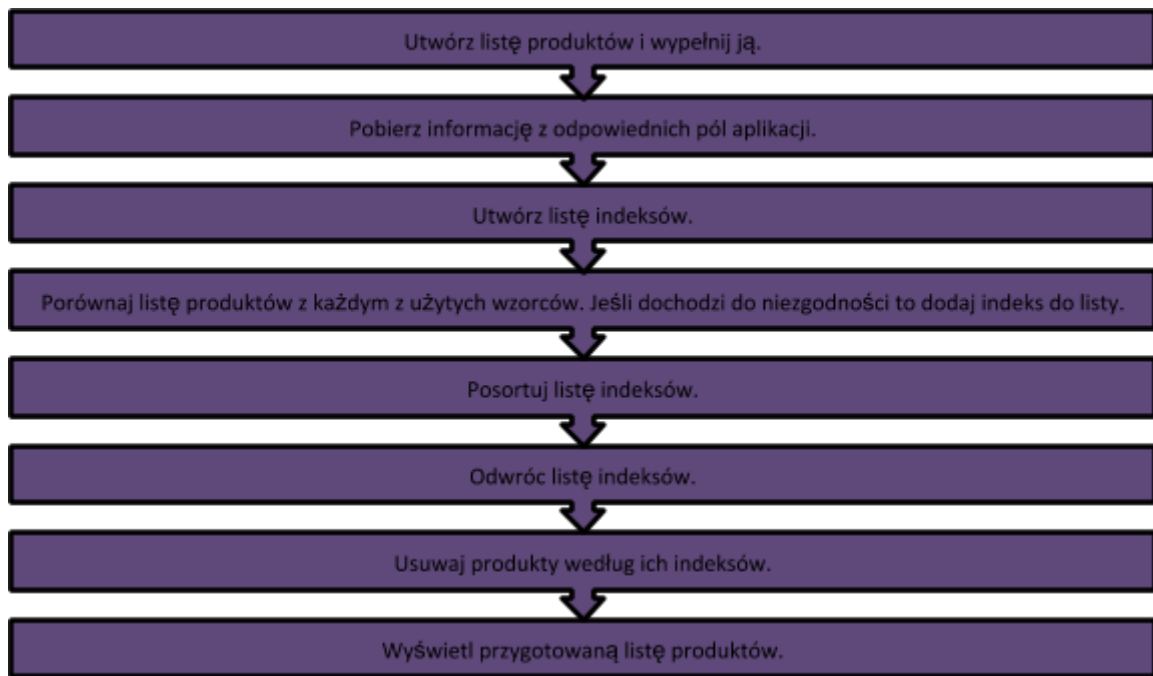
- **public void fillTableWithPartsProducts(JTable jTableProducts,String name)** - wypełnia podaną tabelę wszystkimi produktami z wyjątkiem tych z kategorii 'Jedzenie' które pasują do podanego wzorca.
- **public void removeColumn(JTable table, int x)** - usuwa z podanej tabeli kolumnę o numerze x.
- **public void selectColumns(JTable jTableFoodOrdered, JTable jTableFoodToOrder)** - pomocnicza funkcja, która dostosowuje podane tabele usuwając niepotrzebne kolumny.
- **void refreshOrderedFood(JTable jTableFoodOrdered, List<Product> ordered_list)** - odświeża podaną tabelę, wypełniając ją na nowo produktami z listy.
- **public boolean containsProductID(List<Product> ordered_list, Product temp)** – sprawdza, czy w liście istnieje już produkt o podanym id.

5.3. Zastosowane algorytmy

Algorytm wykorzystany w wyszukiwarce zaawansowanej pozwala na efektywne wyszukiwanie produktów według następujących kategorii:

- Nazwa
- Cena minimalna
- Cena maksymalna
- Ilość minimalna
- Ilość maksymalna
- Kategoria

Pierwszym wykonywanym krokiem jest utworzenie pustej liczby produktów. Lista ta jest następnie wypełniana wszystkimi produktami z bazy z wyjątkiem produktów żywnościowych. Jeśli użytkownik nie zaznaczył żadnej kategorii skrypt nie będzie kontynuowany. Kolejny punkt to pobranie nazwy, wartości cen oraz ilości. W przypadku, gdy te pola są puste, program nie bierze ich pod uwagę. Później tworzona jest lista indeksów, która będzie informowała, który obiekt nie jest zgodny ze wzorcem i należy go usunąć. Teraz algorytm sprawdza każdy wzorzec wypełniony przez użytkownika i porównuje z produktami w liście. Jeśli dochodzi do konfliktu na listę indeksów zostaje wpisany indeks danego produktu (uwzględniając sytuację, że dany obiekt może się tam już znajdować). Kolejnym istotnym krokiem jest posortowanie listy indeksów w kolejności od najmniejszego do największego. Następnie lista jest odwracana. W tym momencie, iterując po indeksach, usuwamy każdy niepasujący element z listy produktów. Zabieg odwrócenia indeksów i usuwania obiektów od końca zapobiega powstawaniu luk i przesuwaniu się indeksów w liście produktów. Na tym etapie lista produktów zawiera tylko te elementy, które pasują do wzorca podanego przez użytkownika. Przebieg algorytmu przedstawia następujący wykres.



Kolejnym istotnym algorytmem jest sposób na konwersję ciągu znaków przechowywanego w zamówieniu na listę produktów. Jest on zawarty w funkcji `convertStringCodeToProductsList` w klasie `Function`. Kod wygląda następująco:

```

public List<Product> convertStringCodeToProductsList(String string) {
    List<Product> list = new ArrayList<Product>();
    String index = "";
    String quantity = "";
    boolean existsIndex = false;

    try {
        for (char x : string.toCharArray()) {
            if (x != ',' && x != ';' && existsIndex == false) {
                index += x;
                continue;
            }
            if (x != ',' && x != ';' && existsIndex == true) {
                quantity += x;
                continue;
            }
            if (x == ',') {
                existsIndex = true;
                continue;
            }
            if (x == ';') {
                int nr = Integer.parseInt(index);
                int count = Integer.parseInt(quantity);
                Product temp = getProductById(nr);
                temp.setQuantity(count);
                list.add(temp);
                index = "";
                quantity = "";
                existsIndex = false;
                continue;
            }
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error while getting products
list.");
    }
    return list;
}

```

Lista produktów jest kodowana w następującym formacie:

[ID produktu],[Ilość produktu]; [ID produktu],[Ilość produktu];...

W ten sposób zamówienie na 5 produktów o ID = 4, 30 produktów o ID = 14 i 2 produkty o ID = 110 wygląda tak:

4,5;14,30;110,2;

Poniżej przedstawione są kolejne iteracje algorytmu (iterator odczytuje każdy znak po kolei).
Za przykład posłuży ciąg

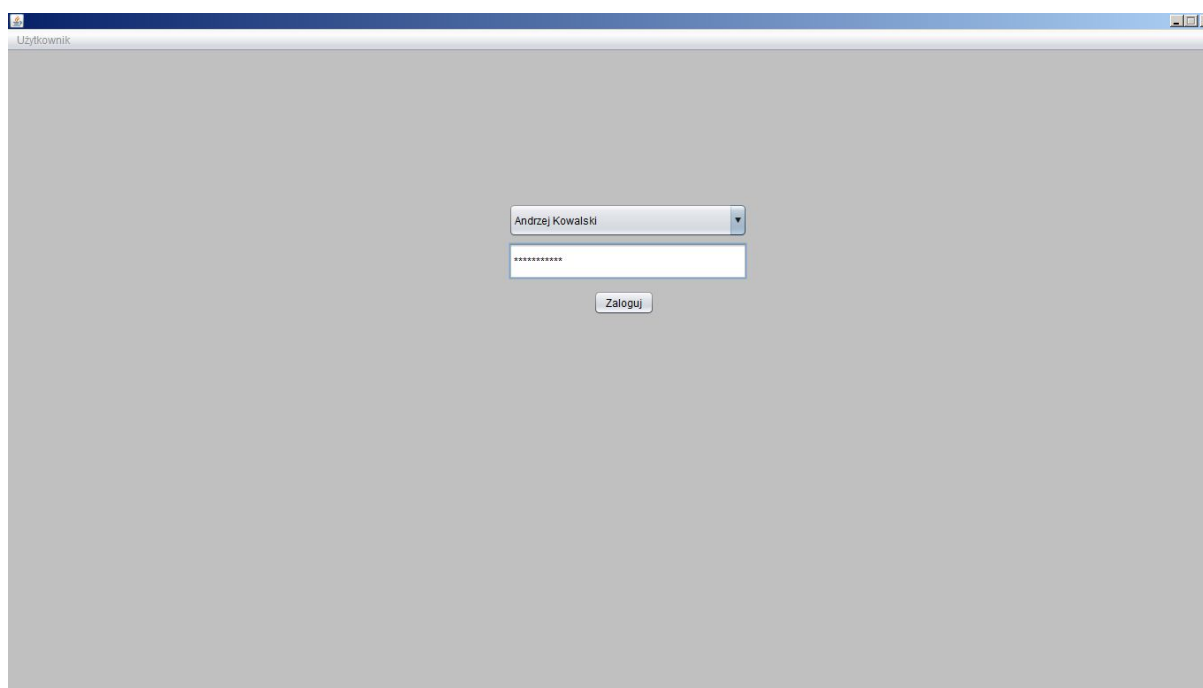
12,3;4,56;

Nr iteracji	Odczytany znak	index	quantity	existsIndex
1	1	„1”	„”	false
2	2	„12”	„”	false
3	,	„12”	„”	true
4	3	„12”	„3”	true
5	;	„12”	„3”	true
W piątej iteracji mamy wszystkie dane na temat produktu. Dodajemy go do listy i zerujemy zmienne index, quantity i existsIndex.				
6	4	„4”	„”	false
7	,	„4”	„”	true

8	5	„4”	„5”	true
9	6	„4”	„56”	true
10	;	„4”	„56”	true
Tutaj znowu dodajemy produkt do listy. Na tym kończy się pętla i mamy gotową listę z produktami.				

6. Dokumentacja użytkownika.

Pierwszym oknem, jakie widzi użytkownik po włączeniu aplikacji, jest okno logowania.



Należy tutaj wybrać z listy rozwijanej odpowiedniego pracownika, wpisać hasło i kliknąć 'Zaloguj'. Hasła w bazie są przechowywane w formie zaszyfrowanej, więc nawet jeśli ktoś zdobędzie dostęp do skryptu z informacjami na temat pracowników, dalej nie będzie mógł zalogować się na nieswoje konto. W lewym górnym rogu znajduje się przycisk z ustawieniami i pocztą meilową. Dostęp do niego uzyska się dopiero po zalogowaniu.

Aplikacja opiera się na prawach dostępu, tj. każdy użytkownik ma dostęp tylko do tej części oprogramowania, która jest mu niezbędna do pracy. Obrazuje to poniższy wykres.

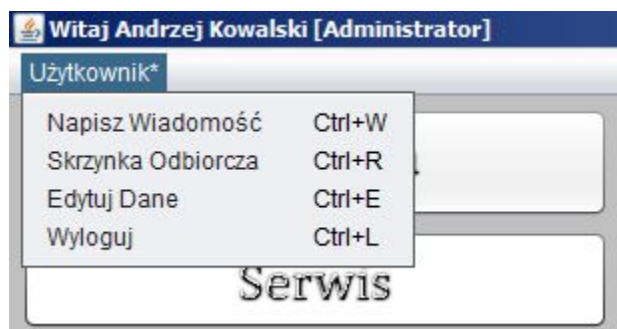
<p>Tworzenie zamówień na naprawę.</p> <p>Sprzedaż części.</p> <p>Zamawianie jedzenia.</p>	<p>Przeglądanie stanu magazynowego.</p> <p>Zamawianie dostaw części.</p> <p>Przyjmowanie zamówień.</p> <p>Dodawanie nowego produktu.</p> <p>Zamawianie jedzenia.</p>	<p>Wykonywanie i zatwierdzanie zamówień na naprawę.</p> <p>Zamawianie jedzenia.</p>	<p>Wszystkie poprzednie możliwości.</p> <p>Dodawanie nowego pracownika.</p>
---	--	---	---

Administrator ma dostęp do wszystkich funkcji i możliwości. Z tego względu dalsza część opisu będzie oparta o konto administratora.

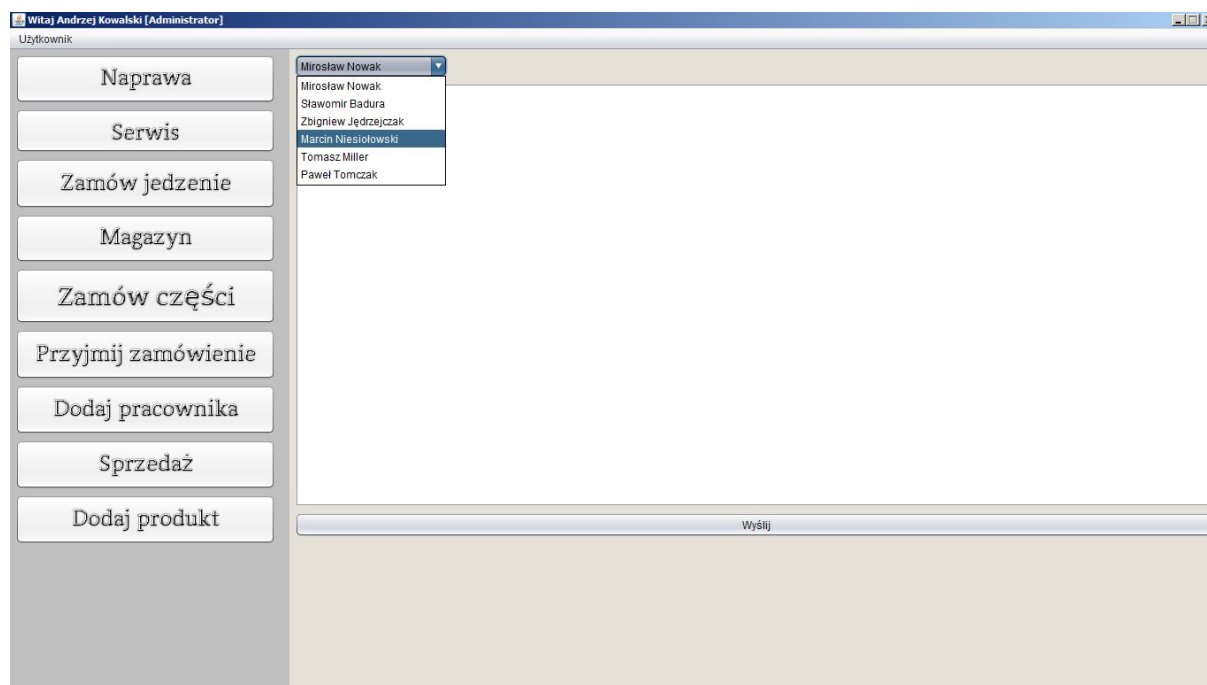
Po zalogowaniu ukazuje się następujące okno.



Podzielone jest ono na dwie części. Pionowy pasek po lewej zawiera główne przyciski przełączające między panelami. Te z kolei wyświetlane są na prawej części okna. W tej chwili obszar ten jest pusty. Na górnym pasie widnieje informacja o zalogowanym użytkowniku. Pod nią mieści się przycisk z rozwijanym menu, którym można zmienić prywatne ustawienia konta lub zarządzać mailami. Gwiazdka informuje użytkownika o tym, że w jego skrzynce mailowej znajduje się nieprzeczytana wiadomość.



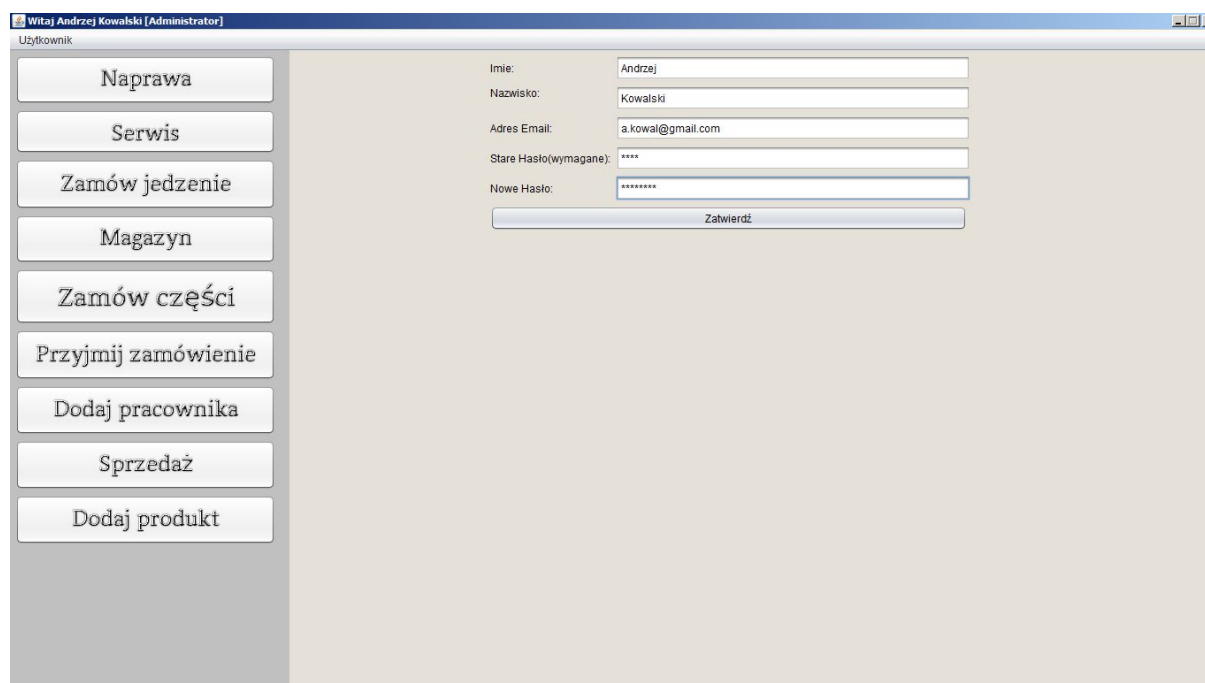
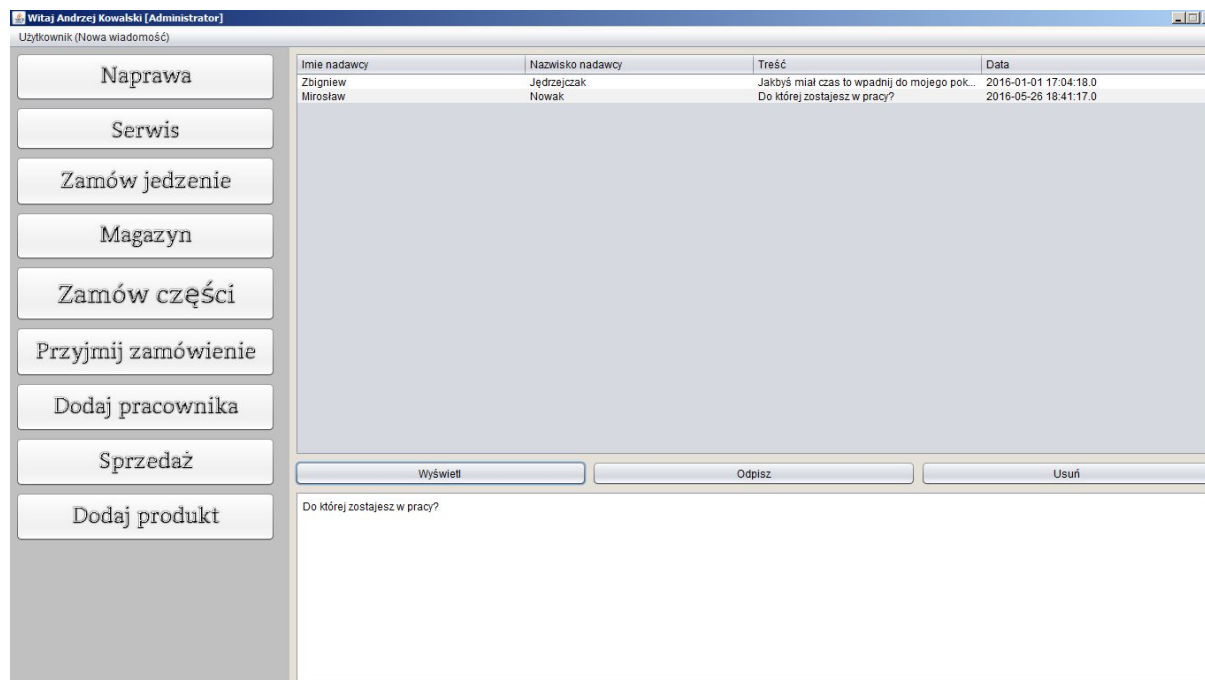
W rozwijalnym menu mamy widoczne opcje. Przydzielone są do nich odpowiednie skróty klawiszowe. Po kliknięciu w 'Napisz Wiadomość' pokazuje się to okno.



Z listy rozwijanej wybieramy adresata, na białym polu piszemy treść, a przycisk 'Wyślij' służy do wysyłania wiadomości. Zostaje ona zapisana w bazie i dostęp do niej ma tylko jej adresat.

Poniżej mamy skrzynkę odbiorczą. Możemy z jej poziomu wyświetlić zaznaczoną wiadomość, odpisać lub ją usunąć. Po każdym wyświetleniu lub usunięciu skrypt sprawdza, czy w skrzynce znajdują się nieprzeczytane wiadomości i jeśli znajdzie takowe, ustawia je na

liście w pierwszej kolejności. Jeśli użytkownik będzie chciał odpisać, program automatycznie przeniesie go do okna pisania wiadomości i sam wybierze adresata.



Powyższe okno służy do konfiguracji ustawień swojego konta. Możemy w nim zmienić takie dane jak imię, nazwisko, adres mailowy, oraz hasło. Niezbędnym warunkiem jest podanie swojego obecnego hasła.

Przycisk 'Wyloguj' przenosi do ekranu logowania i odłącza aktywnego użytkownika od aplikacji.

Przycisk 'Naprawa' służy do włączania następującego panelu.

Witaj Andrzej Kowalski [Administrator]

Użytkownik

Naprawa

Serwis

Zamów jedzenie

Magazyn

Zamów części

Przyjmij zamówienie

Dodaj pracownika

Sprzedaż

Dodaj produkt

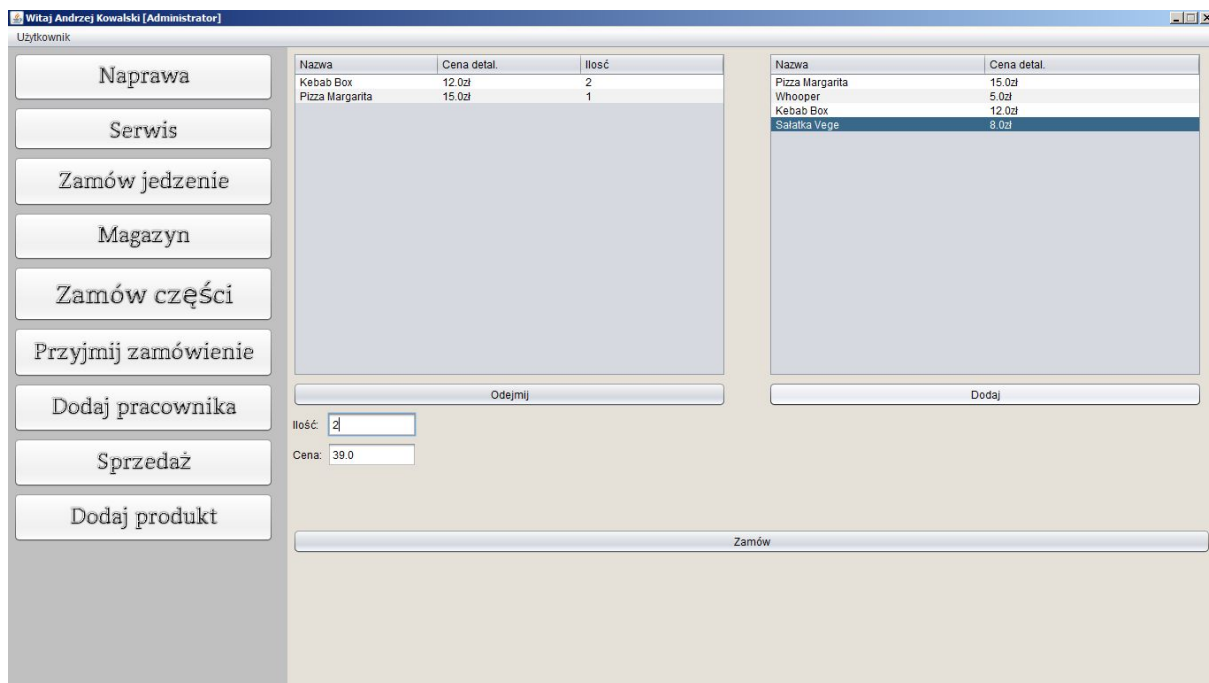
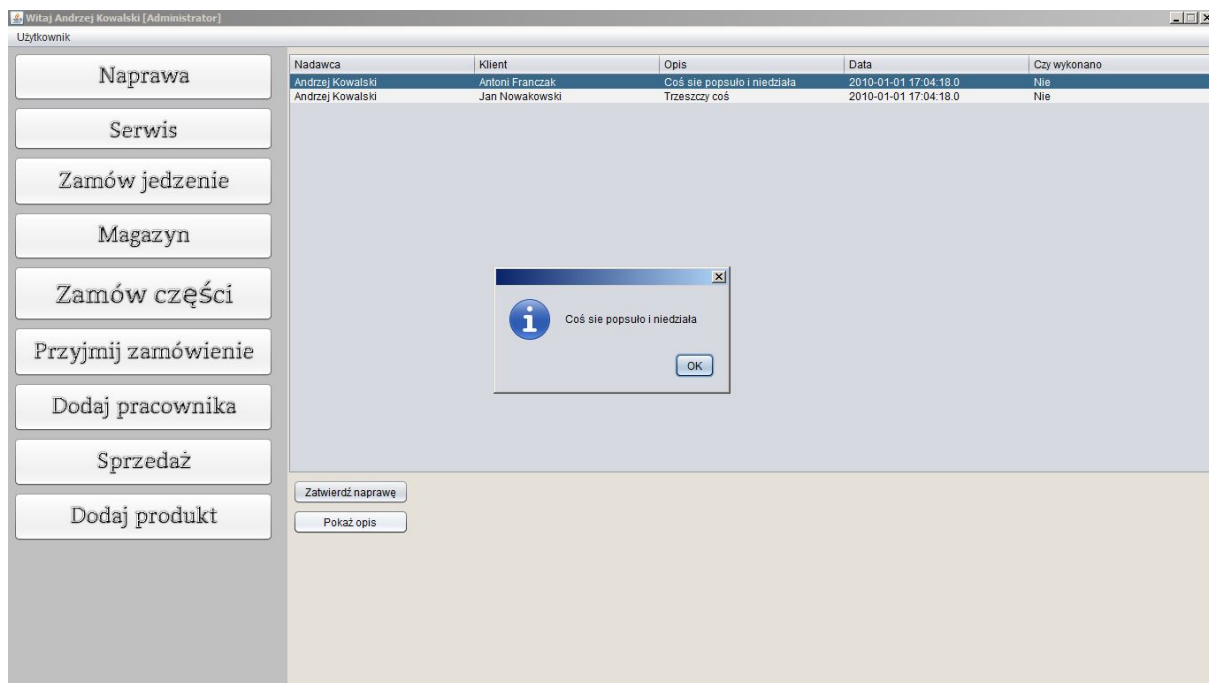
Imię klienta: Jan

Nazwisko klienta: Kowalski

Opis (opcjonalnie): Coś się zepsuło i pojawia się niebieski ekran.

Wyślij informacje

Jest to formularz, jaki składa klient przynoszący sprzęt do naprawy. Pracownik obsługujący wypełnia go danymi takimi jak imię i nazwisko oraz opcjonalnie podaje opis problemu. Formularz ten jest następnie wysyłany do bazy i dostęp do niego zyskują wszyscy technicy. Ich zadaniem jest wykonanie zamówienia i wysłanie potwierdzenia. Technik może też wyświetlić opis osobno. W tabeli zawarte są informacje o tym, który z pracowników nadał zlecenie, dane klienta, opis problemu, data utworzenia i powiadomienie o tym, czy zlecenie zostało już wykonane. Osoba, która dokonuje naprawy może dodać wpis o wykonaniu. Dostęp do tego okna ma również sprzedawca z tą różnicą, że może on tylko odczytywać dane bez ich modyfikowania.



Każdy użytkownik ma możliwość zamawiania jedzenia. Ilustruje to powyższe okno. Prawa tabela przechowuje spis rzeczy, które można zamówić wraz z ceną, prawa natomiast listę wybranych rzeczy. Podczas tworzenia zamówienia w każdej chwili można cofnąć swoją decyzję i usunąć z listy niepotrzebne przedmioty. Kliknięcie 'Zamów' powoduje utworzenie zamówienia, które następnie musi zostać przyjęte przez odpowiedniego pracownika.

Następne okno pozwala na przeglądanie zapasów magazynowych w firmie. Wyświetlone są w nim wszystkie produkty z uwzględnieniem ilości, VAT, ceny i kategorii. Puste pole u góry służy do wyszukiwania. Program sprawdza podaną frazę i wyświetla tylko te produkty, które zawierają ją w nazwie. Jeśli nic nie podano, algorytm zwraca wszystkie produkty w bazie. Okno to służy do szybkiego sprawdzenia stanu magazynowego oraz obecności i ilości produktów.

Nazwa	Cena detal.	VAT	Kategoria	Ilość
Nvidia MX400	450.00zł	0.23	Karty graficzne	48
Toshiba HDD 1TB	200.00zł	0.2	Dyski twarde	10
GOODRAM 1GB	200.00zł	0.3	Pamięć RAM	2
ART Keyboard	40.00zł	0.15	Klawiatury	15
Corsair PS	45.00zł	0.05	Zasilacze	84
Cooler Master 3000	140.00zł	0.02	Chłodzenie	77
Intel i3 3120m	600.00zł	0.01	Procesory	110
GIGABYTE P4PX-E	360.00zł	0.23	Płyty główne	2
LG Flatron	1200.00zł	0.23	Monitory	6
Razer Gaming Mouse	80.00zł	0.05	Myszki	44
ATI Radeon X300	250.00zł	0.23	Karty graficzne	3
Samsung 500GB	150.00zł	0.2	Dyski twarde	55
Kingston 4GB	100.00zł	0.3	Pamięć RAM	42
ART Keyboard	35.00zł	0.15	Klawiatury	155
Razor Supply	140.00zł	0.05	Zasilacze	8
Cooling Pro	80.00zł	0.02	Chłodzenie	17
Intel i7 6700HQ	700.00zł	0.01	Procesory	11
MSI X3E	230.00zł	0.23	Płyty główne	20
BenQ	500.00zł	0.23	Monitory	63
Art Mouse	20.00zł	0.05	Myszki	94

Jeśli użytkownik potrzebuje dokładniejszego narzędzia, może użyć wyszukiwarki zaawansowanej.

Nazwa	Cena detal.	VAT	Kategoria	Ilość
Nvidia MX400	450.00zł	0.23	Karty graficzne	50
Toshiba HDD 1TB	200.00zł	0.2	Dyski twarde	10
GOODRAM 1GB	200.00zł	0.3	Pamięć RAM	2
ART Keyboard	40.00zł	0.15	Klawiatury	15
Corsair PS	45.00zł	0.05	Zasilacze	84
Cooler Master 3000	140.00zł	0.02	Chłodzenie	77
Intel i3 3120m	600.00zł	0.01	Procesory	110
GIGABYTE P4PX-E	360.00zł	0.23	Płyty główne	2
LG Flatron	1200.00zł	0.23	Monitory	6
Razer Gaming Mouse	80.00zł	0.05	Myszki	44
ATI Radeon X300	250.00zł	0.23	Karty graficzne	3
Samsung 500GB	150.00zł	0.2	Dyski twarde	55
Kingston 4GB	100.00zł	0.3	Pamięć RAM	42
ART Keyboard	35.00zł	0.15	Klawiatury	155
Razor Supply	140.00zł	0.05	Zasilacze	8
Cooling Pro	80.00zł	0.02	Chłodzenie	17
Intel i7 6700HQ	700.00zł	0.01	Procesory	11
MSI X3E	230.00zł	0.23	Płyty główne	20
BenQ	500.00zł	0.23	Monitory	63
Art Mouse	20.00zł	0.05	Myszki	94

Za jej pomocą pracownik może zawęzić kryteria wyszukiwania podając nie tylko nazwę, ale także minimalną i maksymalną cenę, jak i ilość. Wynik można sprecyzować, wybierając odpowiednie kategorie produktów.

Następną funkcją programu jest możliwość zamawiania produktów do magazynu. Zaznaczając kategorię po lewej stronie, wybieramy konkretny produkt z listy rozwijanej. Następnie podajemy ilość i klikamy przycisk 'Dodaj'. Po prawej stronie wypełnia się lista z zamówionymi produktami. W prawym dolnym rogu zliczana jest cena z uwzględnieniem podatku. Poniżej przedstawione jest logo wybranej kategorii produktów. Kliknięcie przycisku 'Zamów' tworzy zamówienie, które z kolei zostaje zachowane w bazie i będzie czekało na akceptację przyjęcia.

Witaj Andrzej Kowalski [Administrator]

Użytkownik

Naprawa

Serwis

Zamów jedzenie

Magazyn

Zamów części

Przyjmij zamówienie

Dodaj pracownika

Sprzedaż

Dodaj produkt

Kategorie

- ☐ Karty graficzne
- ☐ Dyski twarde
- ☐ Pamięć RAM
- ☐ Klawiatury
- ☒ Zasilacze
- ☐ Chłodzenie
- ☐ Procesory
- ☐ Płyty główne
- ☐ Monitory
- ☐ Myszki

Corsair PS - 45.0zł 84


Ilość: 2

Dodaj

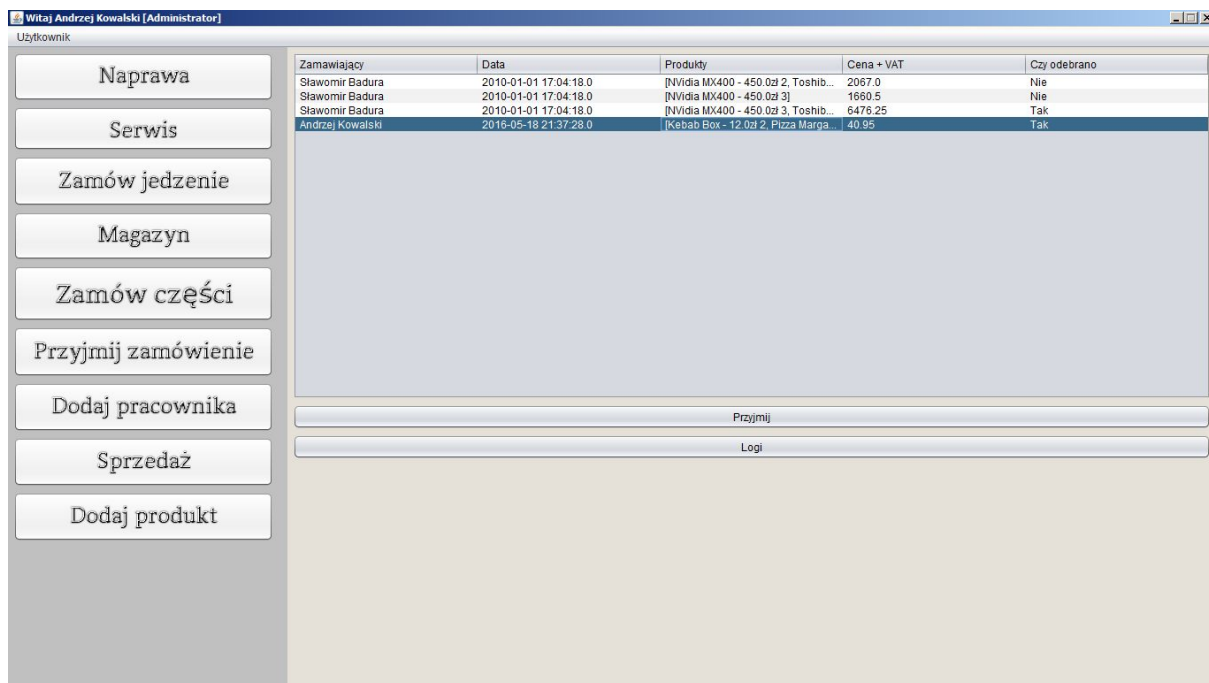
Nazwa	Cena detal.	VAT	Ilość
ATI Radeon X300	250.0zł	0.23	10
Corsair PS	45.0zł	0.05	2

Zamów

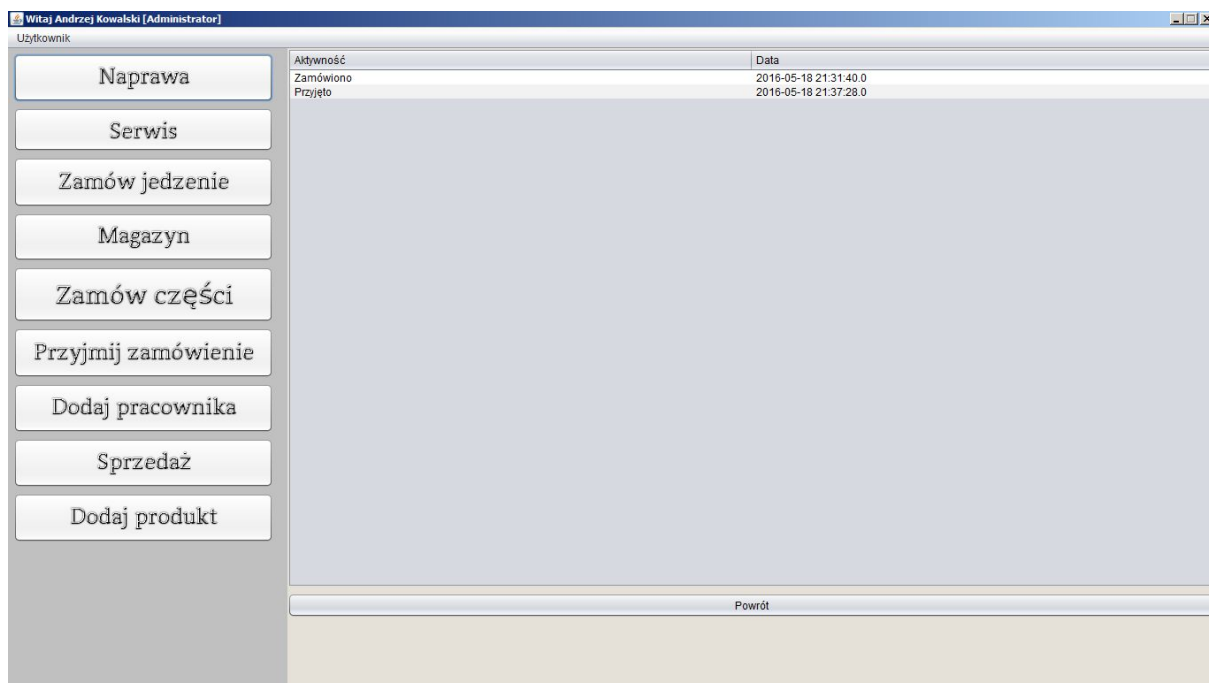
Cena: 3169.5



Przyjmowanie zamówień zachodzi w poniższym oknie. Po zaznaczeniu odpowiedniego wpisu można za pomocą jednego przycisku oznaczyć zamówienie jako odebrane.



Dodatkową opcją są logi. Można się z nich dowiedzieć, kiedy dokładnie zamówiono dostawę oraz kiedy została ona przyjęta.



Jedyną funkcją, którą może wykonywać tylko administrator, jest dodawanie nowego pracownika do firmy. Należy tutaj wpisać dane takie jak imię, nazwisko, adres mailowy, hasło oraz nadać stanowisko pracy. Poprawność adresu mailowego jest sprawdzana przy pomocy specjalnej biblioteki JavaMail.

Witaj Andrzej Kowalski [Administrator]

Użytkownik

Naprawa
Serwis
Zamów jedzenie
Magazyn
Zamów części
Przyjmij zamówienie
Dodaj pracownika
Sprzedaż
Dodaj produkt

Imię: Jan
Nazwisko: Kowalski
Adres mailowy: jan.kowalski@gmail.com
Hasło: qwerty
Powtórz hasło: qwerty
Stanowisko: Magazynier
Dodaj Pracownika

Witaj Andrzej Kowalski [Administrator]

Użytkownik

Naprawa
Serwis
Zamów jedzenie
Magazyn
Zamów części
Przyjmij zamówienie
Dodaj pracownika
Sprzedaż
Dodaj produkt

Nazwa	Cena detal.	VAT	Ilość
Toshiba HDD ...	200.0zł	0.2	2
Kingston 4GB	100.0zł	0.3	3

Ilość: 1
Dodaj
Zakończ

Nazwa	Cena detal.	VAT	Ilość
Intel i3 3120m	600.0zł	0.01	110
Intel i7 6700HQ	700.0zł	0.01	11

Jedną z najważniejszych funkcji jest sprzedaż części. Lewa tabela w oknie powyżej reprezentuje listę przedmiotów, które zamawia klient. Prawa natomiast to lista produktów w sklepie. Górne menu służy do wyboru kategorii produktów. Pole pod nim to miejsce do wpisywania ilości. Przycisk 'Zakończ' generuje fakturę w formacie Pdf.

Całość aplikacji bazuje na gotowych szablonach produktów. Jeśli nie mamy takiego i chcemy go dodać, używamy poniższego okna.

Witaj Andrzej Kowalski [Administrator]

Użytkownik

Naprawa

Serwis

Zamów jedzenie

Magazyn

Zamów części

Przyjmij zamówienie

Dodaj pracownika

Sprzedaż

Dodaj produkt

Nazwa: GeForce MX4400

Cena detaliczna: 170

Podatek: 0.23

Kategoria: Karty graficzne

Dodaj nowy produkt

Podajemy tutaj jego nazwę, cenę, podatek i kategorię. Od tej pory możemy zamawiać dany produkt, a po przyjęciu zamówienia jest on gotowy do sprzedaży.

7. Wnioski

Reasumując, wszystkie założenia programu zostały spełnione, co nie oznacza, że aplikacja nie powinna być rozwijana. Kolejne wersje oprogramowania mogą zostać wzbogacone w możliwość prowadzenia badań nad sprzedażą lub w mechanizmy ułatwiające komunikację między pracownikami jak na przykład chat na żywo, lub kalendarz wydarzeń. Przydatną funkcjonalnością byłoby też zarządzanie flotą przedsiębiorstwa, jak również stworzenia systemu rekrutacji dla nowych pracowników.

8. Bibliografia

- [1] <http://www.microsoftdynamicserp.pl/>
- [2] https://en.wikipedia.org/wiki/Enterprise_resource_planning
- [3] https://en.wikipedia.org/wiki/Economic_order_quantity
- [4] https://en.wikipedia.org/wiki/Toyota_Production_System
- [5] https://en.wikipedia.org/wiki/Material_requirements_planning
- [6] https://en.wikipedia.org/wiki/Manufacturing_resource_planning
- [7] https://en.wikipedia.org/wiki/Enterprise_resource_planning