

Aplikacja desktopowa dla wypożyczalni aut

Speed-Up

Do Nam Do Le

Michał Królikowski

Adam Pokorski

Jako cel obrano stworzenie aplikacji dla pracowników, dzięki której będą mogli zastąpić prace jaką wykonują na co dzień manualnie na system informatyczny, który pozwoli na szybsze i dokładniejsze zastąpienie czynności wykonywanych do tej pory. Jak wiemy technologia cały czas się rozwija, aby móc usprawnić i polepszyć warunki pracy konieczne jest zastosowanie zaawansowanego systemu. Aplikacja desktopowa wydaje się być idealnym zastosowaniem dla prężnie rozwijającej unikatowej firmy jaką jest Speed Up- car rental. Pozwala ona również zapobiec problemowi braku zdigitalizowania danych firmy i przeniesienia jej w zabezpieczone miejsce jakim jest właśnie nasz komputer. Zyskamy dzięki temu również przestrzeń którą musielibyśmy przeznaczyć na składowanie dokumentów.

Zasoby które zostaną zdigitalizowane to:

- Baza danych klientów (Imię nazwisko, adres, miasto, pesel, nr prawa jazdy)
- Baza aut należących do wypożyczalni

Korzyści wynikające z wprowadzenia takiego systemu są takie, że unikniemy błędów przy wydawaniu/obieraniu auta. System nie pozwoli nam na wprowadzenie błędnych danych auta tj. zły numer rejestracyjny, lub błędnych danych klienta w momencie odbioru. Będziemy mieli również wgląd do aut dostępnych co jest także kolejnym ułatwieniem, gdyż zawsze trzeba było to załatwiać papierkowo a teraz wystarczy nam jedno kliknięcie, zapobiegnie to wypożyczenia uszkodzonego auta co mogłoby skutkować zagrożeniem życia klienta bądź zwyczajnie naruszenie renomy firmy.

Za bezpieczeństwo odpowiadał będzie mechanizm logowania. Każdy nowo zatrudniony pracownik przejdzie szkolenie odnośnie funkcjonalności oprogramowania po czym uzyska swój indywidualny numer logowy. Numery będą nadawane przez administratora programu. Żadna osoba z zewnątrz nie mająca swojego numeru nie będzie w stanie zalogować się do aplikacji.

- Utworzenie formularzy jakimi są: ekran logowania, menu, flota, raporty, rezerwacje, zwroty, pomoc, cennik
- Logowanie- odpowiada za wprowadzenie indywidualnego numeru pracownika, który jest w systemowej bazie
- Menu- mamy tu wszystkie dostępne opcje jakie możemy zrobić w programie
- Raporty- formularz dzięki któremu możemy wygenerować sobie informacje z danymi firmy, na tą chwilę są to raporty takie jak: liczba pojazdów, liczba wynajętych aut, liczba wolnych, liczba uszkodzonych
- Rezerwacja - rozbudowany formularz z obowiązkowymi danymi klienta tj. imię nazwisko, adres, miasto, pesel, numer prawa jazdy, oraz wyborem wynajętego auta jak i terminy wynajęcia.

- Zwrot auta - tutaj będziemy mieli możliwość wybrania spośród już wcześniej wpisanych klientów, oraz zaznaczenie daty zwrotu i czy auto jest uszkodzone czy sprawne.
- Cennik - wersja podglądowa dla pracownika w celu przedstawienia ceny dla klienta.
- Pomoc - krótka informacja gdzie się zgłosić w razie problemów bądź pytań odnośnie aplikacji.

Są to założenia, które zostaną wdrożone i zaprogramowane na potrzebę projektu oraz wstępnie wystarczą na wystartowanie aplikacji gotowej do pracy. Z czasem planowany jest rozwój aplikacji o następujące funkcjonalności:

- Wprowadzenie większej ilości raportów do generowania, będą one również bardziej złożone
- Usprawnienie/rozwinięcie opcji logowania o indywidualne hasło dla użytkowników
- Kategoryzowanie aut
- Rozwinięcie bazy danych

Wyróżniamy również wymagania zgodności mające na celu wykazać, że konkretna aplikacja jest wytwarzana w sposób zgodny z ich wymaganiami zasadniczymi i szczegółowymi, określonymi w założeniach projektowych. Należą do nich:

- Weryfikacja użytkownika za pomocą identyfikatora nadawanego przez administratora aplikacji
- Prowadzenie ewidencji samochodów- dodawanie i usuwanie pojazdów z floty- zgodne z oknami rezerwacji oraz zwrotu.
- Rejestracja klientów- dodawanie klientów do bazy, by w łatwy sposób móc się do nich odwołać w innych częściach programu
- Wymaganie zgodności między oknami aplikacji- oznacza w praktyce to, że okno „rezerwacji” musi być zgodne oknem „floty”. Uniemożliwi to sytuacje w której można byłoby wypożyczyć niedostępne auto.
„zwrot”- „rezerwacja” i „flota”
- Informowanie klienta o stałym cenniku wynajmu aut
- Przyjmowanie zwrotów i rezerwacji poprzez zsynchronizowany z innymi funkcjami formularz- flota

By móc rozwijać aplikacje potrzebujemy też czasu i testów, różne sytuacje z codziennego użytkowania systemu pozwolą nam znaleźć więcej potrzeb a także mankamentów do wyeliminowania. Ważną rolę odegra tutaj właśnie użytkownik aplik

jakim jest pracownik firmy Speed Up- car rental. Będzie on nie tylko zasobem korzystającym z systemu ale również testerem. Aplikacja pozostawia miejsce do jej znacznego rozwoju, wprowadzania nowych funkcji dzięki swojej elastyczności. Korzystanie z niej nie wymaga specjalnych umiejętności, gdyż robiona jest w sposób przejrzysty a zarazem prosty.

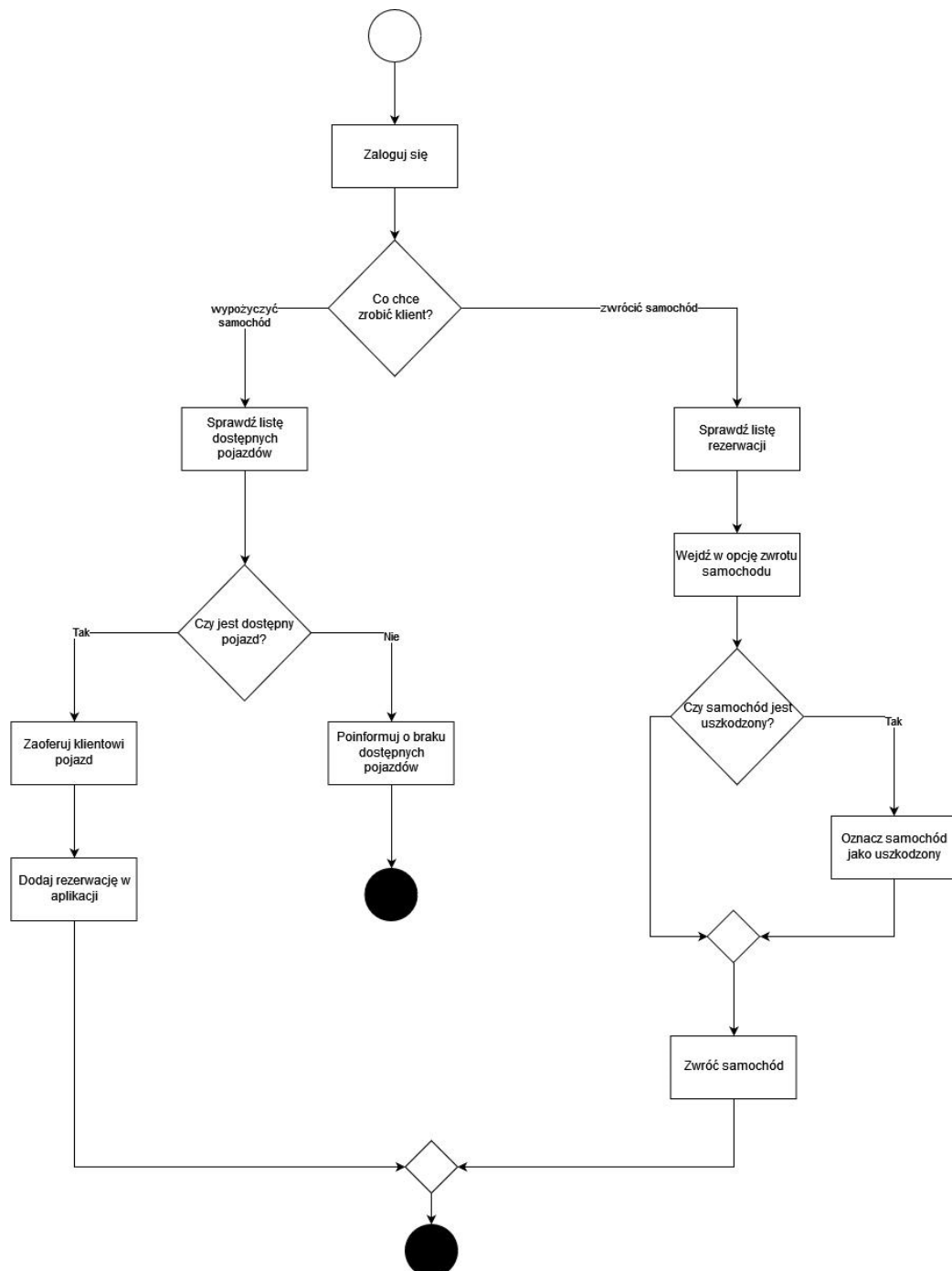
Założenia nawiązują do obecnych problemów logistycznych, które odpowiadają potrzebom grupy docelowej.

Aby lepiej zobrazować poruszoną tematykę w ramach projektu zostały wykonane następujące diagramy.

- diagram przypadków użycia
- diagram aktywności
- diagram dla przypadku dodawania auta
- aktywności dla rezerwacji/zwrotu auta

Diagramy:

Ilustracja 1 Diagram Przypadków użycia dla proponowanego systemu

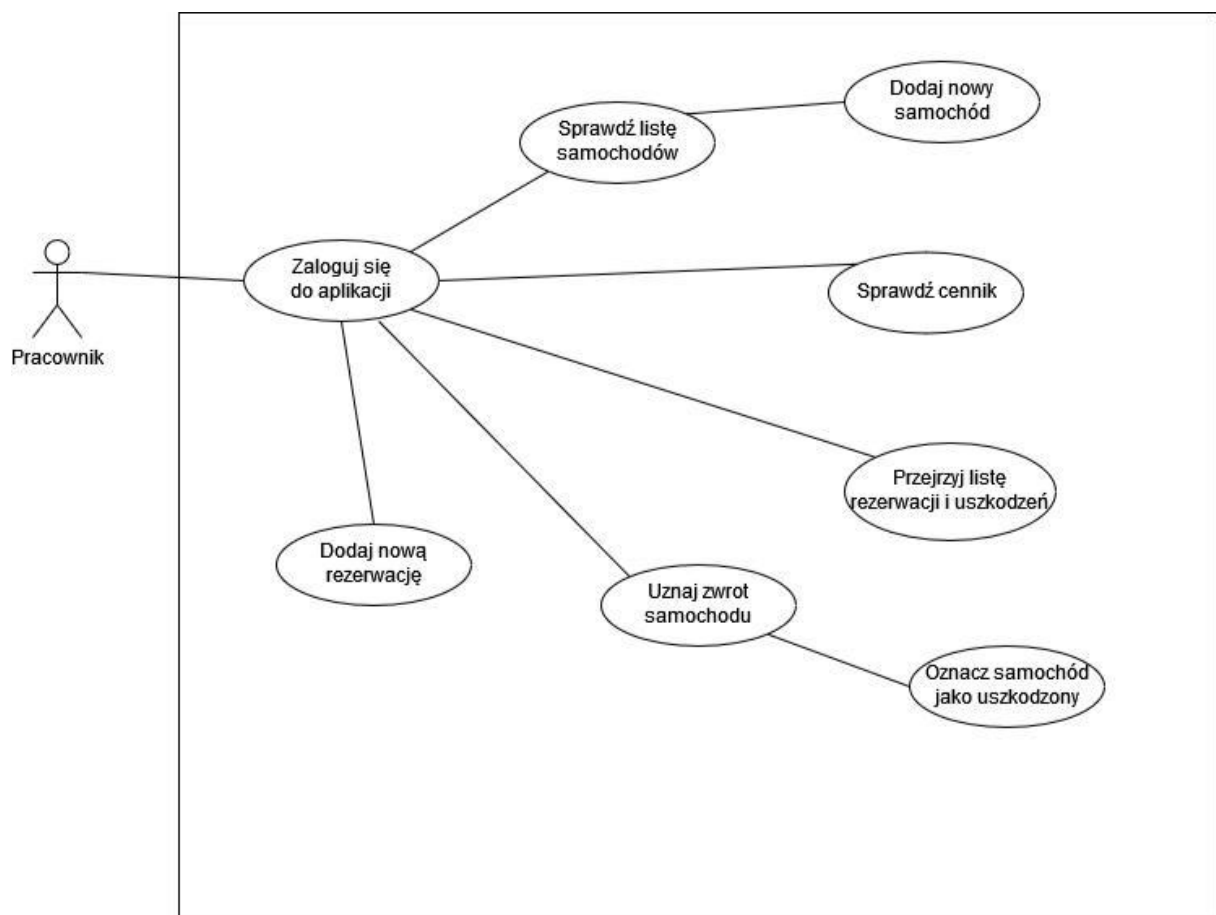


Aktor po zalogowaniu do aplikacji podejmuje decyzje czy chce zwrócić bądź wypożyczyć auto. W przypadku wypożyczenia zmuszony jest przejść przez listę pojazdów oraz sprawdzenie ich

dostępności. W przypadku gdy auto jest dostępne, wtedy może je zaoferować klientowi i dodać rezerwację. W momencie gdy nie ma auta żadanego przez klienta, pracownik zobowiązany jest go o tym poinformować. Istnieje również możliwość zwrotu wcześniej wypożyczonego pojazdu, odbywa się to w następującej kolejności:

- pracownik sprawdza listę rezerwacji i wybiera odpowiednią go pozycję.
- sprawdza czy samochód jest uszkodzony jeżeli nie to przyjmuje zwrot. W przypadku gdy auto jest uszkodzone, zaznacza w formularzu checkbox oraz sporządza notatkę.

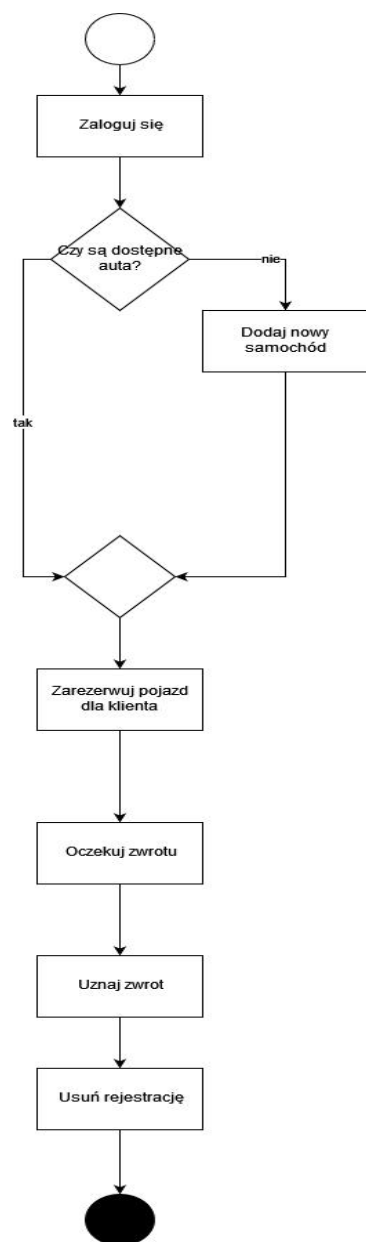
Ilustracja 2 Diagram aktywności proponowanego system



Jest to diagram obrazujący dostępne czynności, dla użytkownika danej aplikacji. Diagram aktywności umożliwia określenie tego, w jaki sposób system będzie osiągał swoje zamierzone cele. Użytkownikiem aplikacji jest pracownik firmy. Po zalogowaniu udostępnione mu są następujące funkcjonalności:

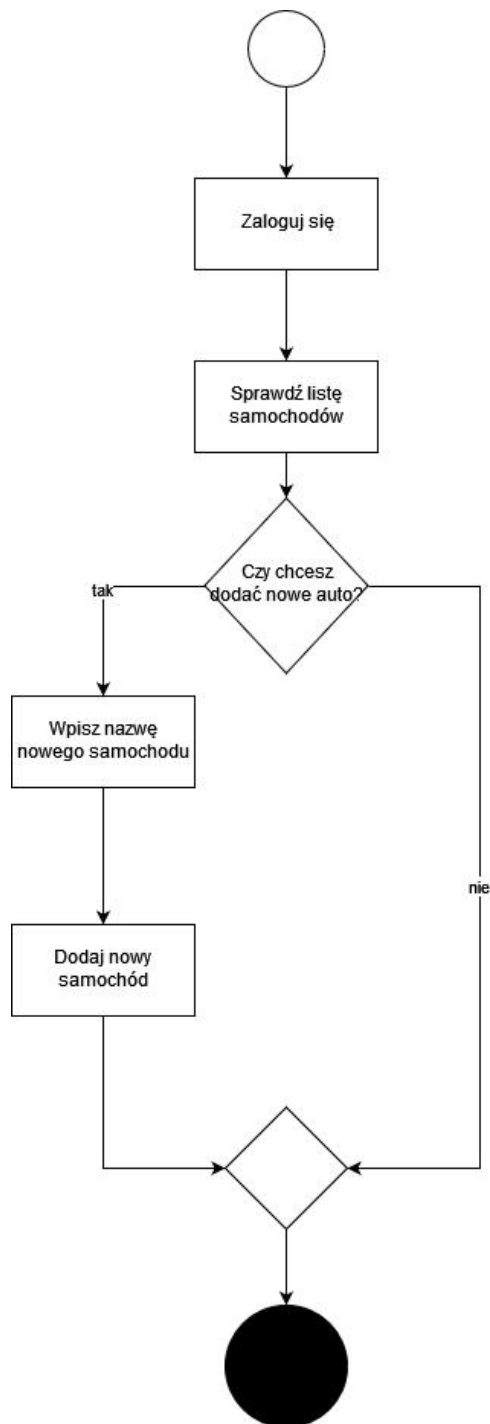
-sprawdzenie listy samochodów
-wgląd do cennika
-przegląd listy rezerwacji i uszkodzeń
pojazdów -dodanie nowej rezerwacji -dodanie
zwrotu pojazdu.

Ilustracja 2 Diagram dla przypadku użycia (dodawanie i zakończenie rezerwacji) dla proponowanego systemu



Powyżej przedstawiono diagram aktywności dla konkretnego zadania jakim jest rezerwacja wraz ze zwrotem pojazdu. Pracownik po zalogowaniu do programu sprawdza dostępne auta i z tego poziomu ma możliwość dodania nowego samochodu lub przejścia bezpośrednio do rezerwacji pojazdu dla klienta. Po upływie czasu najmu rozpoczyna się proces zwrotu auta którego skutkiem jest zamknięcie rezerwacji.

Ilustracja 3 Diagram aktywności dla dodawania auta w danym projekcie



Kolejny diagram aktywności jest poświęcony przypadkowi dodawania auta. Odbywa się to w sposób analogiczny do poprzednich wersji poprzez wcześniejsze zalogowanie pracownika do programu. Otwierając listę samochodów ma możliwość dodania nowego auta bądź wgląd na dostępne pojazdy. Wybierając opcje dodania pojazdów prosi nas o wpisanie nazwy oraz modelu auta a następnie zatwierdzenie poprzez kliknięcie przycisku „dodaj”.

Projektowanie aplikacji

Aplikacja została stworzona w celu przyspieszenia pracy oraz ze względu na bezpieczeństwo danych firmy. Wymagania sprzętowe są o tyle niskie, że pozwolą uruchomić nam aplikację nawet na starym sprzęcie, jedynym warunkiem jest system Windows 7 bądź nowszy. Instalacja gotowego produktu przebiega w sposób bardzo prosty, aplikacja jest pobierana zasobu internetowego i bezpośrednio uruchamiana z pliku exe. Do korzystania z programu potrzebujemy tylko podstawowe peryferia jakimi są monitor, myszka, klawiatura. Dostęp do Internetu nie jest wymogiem.

Program który wykorzystano do stworzenia aplikacji to Visual Studio. Jest to zintegrowane środowisko programistyczne dostarczone przez firmę Microsoft, powszechnie używane do tworzenia oprogramowania konsolowego oraz z graficznym interfejsem zwanym Windows Forms. To właśnie tego drugiego użyto do stworzenia projektu. By móc z niego korzystać wymagane jest doinstalowanie pakietu z narzędzia Visual Studio. Środowisko to ułatwia nam konfigurowanie, kompilowanie, debugowanie, tworzenie pakietów i wdrażanie aplikacji.

Okno logowania

Moduł logowania



Źródło: opracowanie własne z użyciem narzędzia Visual Studio

Każdy pracownik by móc korzystać z aplikacji jest zmuszony wprowadzić swój indywidualny numer nadawany przez administratora aplikacji. W momencie gdy wprowadzony numer jest błędny otrzymujemy komunikat „Wprowadzono niepoprawny numer pracownika”. Numer pracownika ma 6 cyfr, przykładowy numer wykorzystany do testów aplikacji to 654321. Po zalogowaniu przenosi nas bezpośrednio do menu.

Menu

Menu aplikacji



Źródło: opracowanie własne z użyciem narzędzia Visual Studio

Menu jest stworzone w prosty i intuicyjny sposób, aby żaden z pracowników nie miał problemu z użytkowaniem. Po zalogowaniu mamy dostęp do 6 zakładek tj.

- Flota
- Raporty
- Rezerwacja
- Zwrot
- Pomoc
- Cennik

Zamykanie aplikacji następuje po kliknięciu na przycisk X w prawym górnym logu. Pracownik z automatu zostaje wylogowany.

Inicjalizacja programu w module menu

```
public partial class Menu : Form
{
    public Menu()
    {
        private void button_rezerwacje_Click(object sender, EventArgs e)
        {
            Rezerwacja rezerwacja = new Rezerwacja();
            rezerwacja.Show();
        }
        private void menu_Load(object sender, EventArgs e)
        {
            try
            {
                FileStream fs = new FileStream("vehicleList.dat", FileMode.Open, FileAccess.Read);

                BinaryFormatter bf = new BinaryFormatter();

                Vehicle.vehicleList = (List<Vehicle>)bf.Deserialize(fs);

                fs.Close();
            }
            catch (Exception ex)
            {
                Vehicle vehicle1 = new Vehicle("Nissan Patrol", false, false, null);
                Vehicle vehicle2 = new Vehicle("Jepp Grand Cherokee", false, false, null);
                Vehicle vehicle3 = new Vehicle("Jepp Wrangler", false, false, null);
                Vehicle vehicle4 = new Vehicle("Toyota Land Cruiser", false, false, null);
                Vehicle vehicle5 = new Vehicle("Mitsubishi Pajero", false, false, null);
                Vehicle vehicle6 = new Vehicle("Mitsubishi Lancer", false, false, null);
                Vehicle vehicle7 = new Vehicle("Land Rover Discovery", false, false, null);
                Vehicle vehicle8 = new Vehicle("Nissan 370z", false, false, null);
                Vehicle vehicle9 = new Vehicle("Suzuki Samurai", false, false, null);
            }
            try
            {
                FileStream fs = new FileStream("clientList.dat", FileMode.Open, FileAccess.Read);

                BinaryFormatter bf = new BinaryFormatter();

                Reservation.clientList = (List<Reservation>)bf.Deserialize(fs);

                fs.Close();
            }
            catch (Exception ex2)
            {
                MessageBox.Show(ex2.Message, "Brak pliku clientList.dat - lista rezerwacji będzie pusta.")
            }
        }
    }
}
```

Jest to fragment kodu odpowiadający za inicjalizację programu. Wartości początkowe są domyślnie wczytywane do programu. Klasa FileStream jest odpowiedzialna za odczyt pliku startowego jakim jest vehicleList.dat. Wczytywane pliki są zserializowane i nie można w nie integrować z zewnątrz. VehicleList.Add użyto w celu uzupełnienia modułu floty dostępnymi autami. W przypadku braku pliku clientList.dat program pokaże błąd „Brak pliku clientList.dat – lista rezerwacji będzie pusta.”

Logika przycisków dla modułu menu

```
private void button_raporty_Click(object sender, EventArgs e)
{
    Raporty raporty = new Raporty();
    raporty.Show();
}
private void button_flota_Click(object sender, EventArgs e)
{
    Flota flota = new Flota();
    flota.Show();
}
private void button_zwrot_Click(object sender, EventArgs e)
{
    if (Reservation.clientList.Count == 0)
    {
        MessageBox.Show("Brak rezerwacji na obecny moment.");
    }
    else
    {
        Zwrot zwrot = new Zwrot();
        zwrot.Show();
    }
}
private void button_pomoc_Click(object sender, EventArgs e)
{
    Pomoc pomoc = new Pomoc();
    pomoc.Show();
}
private void button_cennik_Click(object sender, EventArgs e)
{
    Cennik cennik = new Cennik();
    cennik.Show();
}
```

Ta część kodu jest odpowiedzialna za logikę przycisków, pozwala uruchomić konkretny moduł. Button_Click reaguje na zdarzenie jakim jest kliknięcie przyciskiem myszy na dany przycisk.

zapisywanie danych modułu menu

```
private void Menu_FormClosing(object sender, FormClosingEventArgs e)
{
    BinaryFormatter bf = new BinaryFormatter();

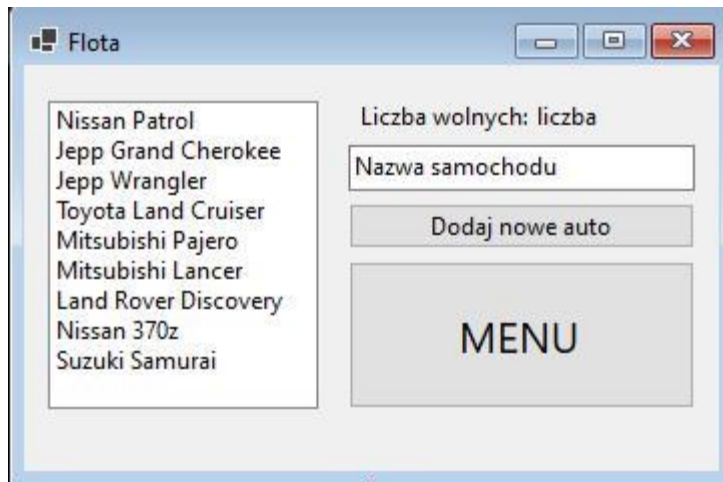
    FileStream fs = new FileStream("vehicleList.dat", FileMode.Create, FileAccess.Write);
    bf.Serialize(fs, Vehicle.vehicleList);
    fs.Close();

    FileStream fs2 = new FileStream("clientList.dat", FileMode.Create, FileAccess.Write);
    bf.Serialize(fs2, Reservation.clientList);
    fs2.Close();
}
```

Fragment odpowiedzialny za zapis danych do pliku, w momencie wyłączenia programu. FileStream nadpisuje plik vehicleList oraz clientList po czym zamyka aplikację.

Flota

Moduł flota



Źródło: opracowanie własne z użyciem narzędzia Visual Studio

W tym module wyświetla nam się ListBox z wszystkimi dostępnymi pojazdami jakimi dysponuje firma. Można również dodać auto wpisując jego nazwę i klikając button „dodaj nowe auto”. Z tego modułu mamy również możliwość powrotu do Menu.

inicjalizacja formatki flota

```
public Flota()
{
    InitializeComponent();
}

private void Raporty_Load(object sender, EventArgs e)
{
    listBox1.Items.Clear();
    for (int i = 0; i < Vehicle.vehicleList.Count; i++)
    {
        listBox1.Items.Add(Vehicle.vehicleList[i].vehicleName);
    }
    label2.Text = Vehicle.ReturnFreeCarsNumber();
}
```

Fragment odpowiadający za inicjalizację formatki. Pobiera nazwy pojazdów z listy i dodaje do licznika aut wolnych.

dodawanie auta w module flota

```
private void button1_Click(object sender, EventArgs e)
{
    Vehicle vehicle = new Vehicle(textBox1.Text, false, false,
    null); listBox1.Items.Clear();
    for (int i = 0; i < Vehicle.vehicleList.Count; i++)
    {
        listBox1.Items.Add(Vehicle.vehicleList[i].vehicleName);
    }
}
```

```

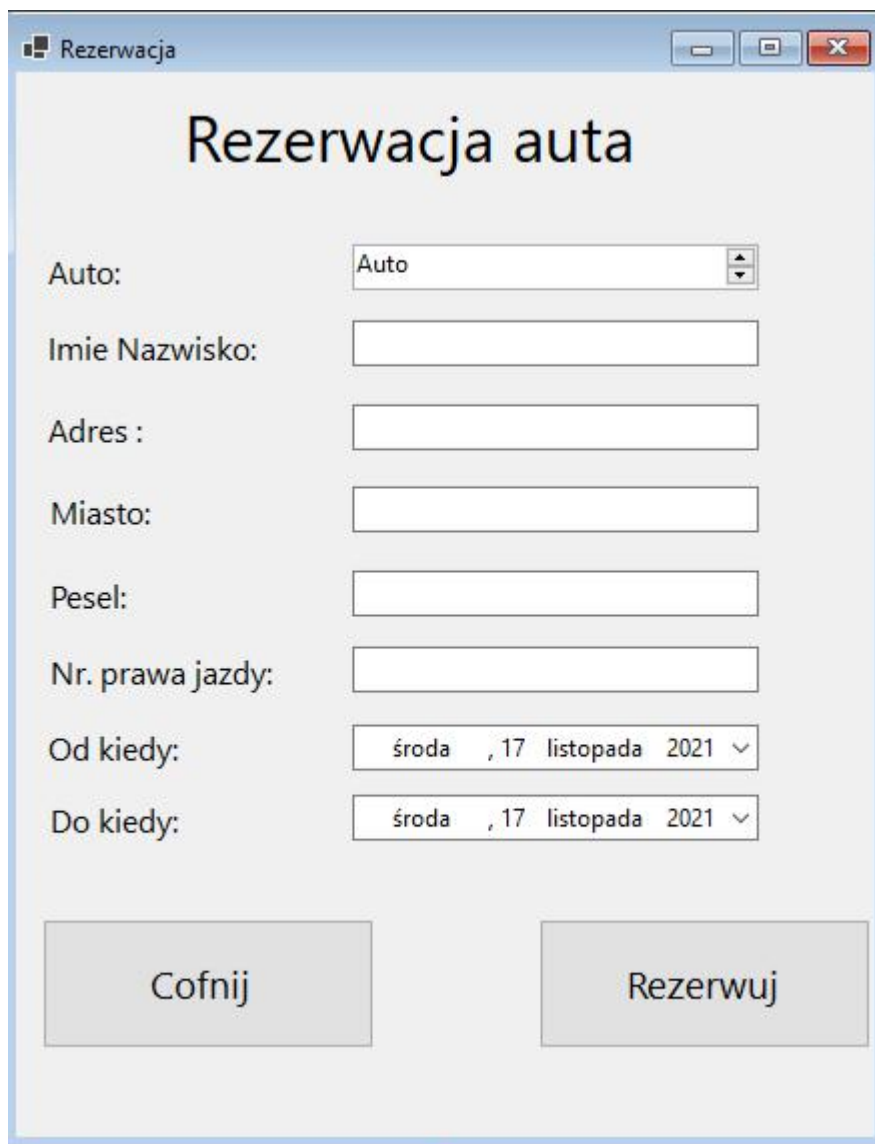
    }
    textBox1.Text = "Nazwa samochodu";
    MessageBox.Show("Pojazd został dodany");
}

```

Po kliknięciu na button1 „dodaj nowe auto” pojazd zostaje dodany do listbox. W tym samym czasie textbox jest czyszczony a pracownikowi wyświetla się komunikat „pojazd został dodany”

Rezerwacja

Moduł rezerwacja



The screenshot shows a Windows application window titled "Rezerwacja". Inside the window is a form titled "Rezerwacja auta". The form contains the following fields and controls:

- Auto:** A dropdown menu with "Auto" selected.
- Imie Nazwisko:** A text input field.
- Adres :** A text input field.
- Miasto:** A text input field.
- Pesel:** A text input field.
- Nr. prawa jazdy:** A text input field.
- Od kiedy:** A date picker showing "środa , 17 listopada 2021".
- Do kiedy:** A date picker showing "środa , 17 listopada 2021".
- Buttons:** Two buttons at the bottom: "Cofnij" (Cancel) and "Rezerwuj" (Reserve).

Źródło: opracowanie własne z użyciem narzędzia Visual Studio

Moduł odpowiedzialny za złożenie rezerwacji. Odbywa się ona w sposób następujący:

- 1) Pracownik wybiera auto z listy dostępnych pojazdów
- 2) Pracownik wprowadza dane klienta typu: imię nazwisko, adres, miasto, pesel oraz numer prawa jazdy

3) Pracownik wybiera okres od kiedy, do kiedy będzie trwał najem

Zapisanie rezerwacji finalizuje kliknięcie przycisku „rezerwuj”.

Wciśnięcie przycisku „cofnij” przenosi pracownika do menu

dodawanie rezerwacji

```
private void btn_rezerwuj_Click(object sender, EventArgs e)
{
    if (textBox1.Text != null || textBox2.Text != null || textBox3.Text != null || textBox4.Text != null
    || textBox1.Text != null)
    {
        if (Vehicle.vehicleList[domainUpDown1.SelectedIndex].isReserved != true
        && Vehicle.vehicleList[domainUpDown1.SelectedIndex].isDamaged != true)
        {
            Reservation reservation = new Reservation(textBox1.Text, textBox2.Text, textBox3.Text,
            textBox4.Text, textBox5.Text, domainUpDown1.SelectedIndex, dateTimePicker1.Value, dateTimePicker2.Value);
            Vehicle.ReserveVehicle(domainUpDown1.SelectedIndex);

            MessageBox.Show("Rezerwacja została dodana");
            this.Hide();
        }
        else
        {
            MessageBox.Show("Dany pojazd nie jest dostępny na ten moment.");
        }
    }
    else
    {
        MessageBox.Show("Należy uzupełnić wszystkie pola.");
    }
}
```

TextBoxy ustawione na „null” powodują że okienka są puste i nie posiadają żadnych nadpisanych treści. Auto wybieramy za pomocą narzędzia DomainUpDown gdzie wyświetlane są wszystkie auta. Do pliku clientList dodawany jest rekord z danymi nowo wpisanego klienta. Gdy pracownik przejdzie przez formularz poprawnie otrzyma komunikat „ Rezerwacja została dodane” w przeciwnym razie może zdarzyć się tak, że wybrany pojazd będzie niedostępny i wtedy pracownik dostanie o tym powiadomienie.

pobieranie nazw pojazdów

```
private void btn_rezerwacja_wroc_Click(object sender, EventArgs e)
{
    this.Hide();
}
private void Rezerwacja_Load(object sender, EventArgs e)
{
    for (int i = 0; i < Vehicle.vehicleList.Count; i++)
    {
        domainUpDown1.Items.Add(Vehicle.vehicleList[i].vehicleName);
    }
}
```

Wciśnięcie przycisku cofnij powoduje zamknięcie danego modułu i powrót do menu. Pętla for dodaje nazwy pojazdów do narzędzia domainUpDown.

Raporty

Moduł raporty

The screenshot shows a Windows application window titled "Raporty". Inside the window, there are two data grids. The first grid, titled "Rezerwacje", has columns: "Imię Nazwisko", "Adres", "Miasto", "PESEL", "Nr prawa jazdy", "Od", and "Do". The second grid, titled "Uszkodzenia", has columns: "Pojazd" and "Opis uszkodzenia". Below the grids, there is a button labeled "MENU".

Źródło: opracowanie własne z użyciem narzędzia Visual Studio

W danym module po wpisaniu rezerwacji wyświetli nam się w tabelce rekord. Każda osobna aktywna rezerwacja będzie miała swój rekord. Zawarte będą w nim dane takie jak: imię nazwisko, adres, miasto, pesel, numer prawa jazdy, od kiedy i do kiedy jest rezerwacja. Drugim raportem jaki obsługuje ten moduł jest raport uszkodzeń. Gdy w momencie zwrotu auta pracownik zaznaczy checkbox „uszkodzony” i napisze w notatce informacje co uległo zniszczeniu, wtedy dany pojazd zostanie wyświetlony w raporcie. Zawarto również przycisk cofający nas do menu.

uzupełnianie raportów

```
public Raporty()
{
    InitializeComponent();
}

private void btn_menuraporty_Click(object sender, EventArgs e)
{
    this.Hide();
}

private void Raporty_Load(object sender, EventArgs e)
{
    for (int i = 0; i < Reservation.clientList.Count; i++)
    {
        int index = dataGridView1.Rows.Add();
```



```

dataGridView1.Rows[index].Cells[0].Value = Reservation.clientList[i].name;
dataGridView1.Rows[index].Cells[1].Value = Reservation.clientList[i].address;
dataGridView1.Rows[index].Cells[2].Value = Reservation.clientList[i].city;
dataGridView1.Rows[index].Cells[3].Value = Reservation.clientList[i].pesel;
dataGridView1.Rows[index].Cells[4].Value = Reservation.clientList[i].licenceNumber;
dataGridView1.Rows[index].Cells[5].Value =
Reservation.clientList[i].reserveFrom.ToShortDateString(); dataGridView1.Rows[index].Cells[6].Value
= Reservation.clientList[i].reserveTo.ToShortDateString();
}

for (int i = 0; i < Vehicle.vehicleList.Count; i++)
{
    if (Vehicle.vehicleList[i].isDamaged)
    {
        int index = dataGridView2.Rows.Add();
        dataGridView2.Rows[index].Cells[0].Value = Vehicle.vehicleList[i].vehicleName;
        dataGridView2.Rows[index].Cells[1].Value = Vehicle.vehicleList[i].damageInfo;
    }
}

```

Pętla for porusza się po wszystkich rekordach klientów. Uzupełnia formatkę DataGridView danymi z rezerwacji oraz DataGridView2 w momencie gdy zwrócone auto jest uszkodzone. Wpisywane są takie dane jak nazwa auta i opis szkody. Druga pętla for jest odpowiedzialna za wyselekcjonowanie tylko aut uszkodzonych.

Zwrot

Moduł zwrotu

Źródło: opracowanie własne z użyciem narzędzia Visual Studio

Formularz odpowiedzialny za zwrot auta. Pracownik z listy wybiera imię i nazwisko klienta zwracającego pojazd, po czym wybiera datę zwrotu. Gdy zwrócone auto posiada jakieś

uszkodzenia, pracownik zmuszony jest zaznaczyć checkbox „uszkodzone” oraz sporządzić notatkę z opisem uszkodzenia. Po kliknięciu przycisku zwróć pracownik dostaje komunikat „zwrot został uwzględniony”.

zwrot pojazdów

```
private void btn_zwrot_Click(object sender, EventArgs e)
{
    int vehID = Reservation.ReturnVehicleID(domainUpDown1.SelectedIndex);
    Vehicle.UnReserveVehicle(vehID);
    if (checkBox1.Checked)
    {
        Vehicle.DamageVehicle(vehID);
        Vehicle.ChangeDamageInfo(vehID, textBox1.Text);
    }

    Reservation.RemoveReservation(domainUpDown1.SelectedIndex);

    domainUpDown1.Items.Clear();
    for (int i = 0; i < Reservation.clientList.Count; i++)
    {
        domainUpDown1.Items.Add(Reservation.clientList[i].name);
    }

    MessageBox.Show("Zwrot został uwzględniony");
}

private void Zwrot_Load(object sender, EventArgs e)
{
    for (int i = 0; i < Reservation.clientList.Count; i++)
    {
        domainUpDown1.Items.Add(Reservation.clientList[i].name);
    }
}
```

Zmienna vehID przechowuje dane aut wraz z przypisanym im klientem. Jeśli checkbox „uszkodzony” jest zaznaczony wtedy zapisuje do zmiennej wraz z notatką. Druga for jest odpowiedzialna za aktualizację listy klientów. Po wszystkim rezerwacja znika z listy.

Cennik

Moduł cennik

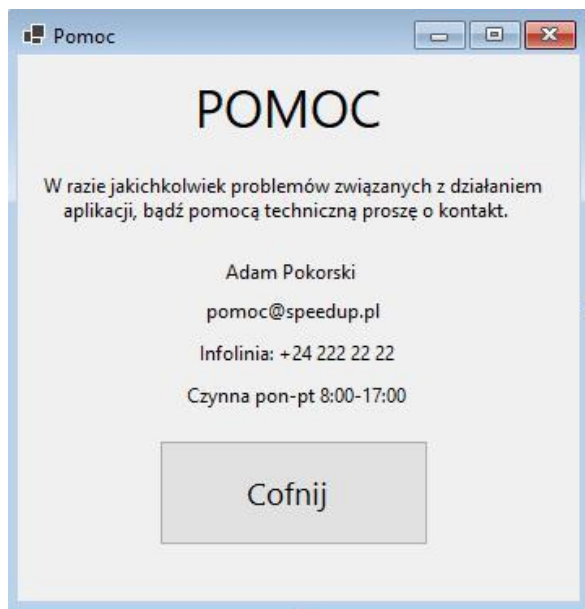


Źródło: opracowanie własne z użyciem narzędzia Visual Studio

Jest to moduł informacyjny składający się z Label-ów. Pracownik ma wgląd do cennika jaki obowiązuje w jego firmie. W tym module nie wykonują się żadne zadania, a po kliknięciu w przycisk cofnij przenosi pracownika do menu.

Pomoc

Moduł pomoc



Źródło: opracowanie własne z użyciem narzędzia Visual Studio

Tak jak w przypadku modułu cennika, moduł ten nie wykonuje żadnych zadań a służy jedynie jako informacja dla pracownika w jaki sposób skontaktować się z administratorem programu w celu pojawienia się jakiś problemów z działaniem systemu.

Podsumowanie

Rewolucja cyfrowa i coraz powszechniejsze przyjmowanie ekologicznego stylu życia doprowadziły do pojawienia się i szybkiego rozwoju wielu form współdzielonej mobilności, zagrażając nie tylko trwałości tradycyjnych modeli wynajmu samochodów, ale także ich istnieniu. Aby odnieść sukces na konkurencyjnym technologicznym rynku, należy przyjąć nowe technologie lub rozpocząć biznes oparty na technologii, taki jak wspólne korzystanie z samochodów.

Celem niniejszej pracy było przedstawienie projektu, który polega na zaproponowaniu prototypu mobilnego systemu wynajmu samochodów, który byłby zabezpieczony i umożliwił pracownikom wypożyczalni aut usprawnienie zarządzania firmą. Aplikacja została stworzona w celu przyspieszenia pracy oraz ze względu na bezpieczeństwo danych firmy. Program który wykorzystano do stworzenia aplikacji to Visual Studio.

Zaprojektowano rozwiązanie do wspólnego korzystania z samochodów, które pokazuje, jak można skutecznie osiągnąć możliwość prawidłowego zarządzania flotą. Istotnym elementem aplikacji było uwzględnienie poszczególnych funkcji:

- Funkcja zarządzania użytkownikami;
- Funkcja zarządzania płatnościami;
- Funkcje zarządzania flotą;
- Funkcja zarządzania raportami i analizami.

Usprawnienie zarządzania wypożyczalnią było głównym celem aplikacji, zatem aplikacja składa się z paru modułów, których zadaniem będzie polepszenie pracy konkretnych pracowników.

Wdrożenie aplikacji będzie przynosiło spore korzyści wynikające z wprowadzenia takiego systemu. System nie pozwoli na wprowadzenie błędnych danych auta lub błędnych danych klienta w momencie odbioru. Osoby zarządzające flotą będą miały wgląd do aut dostępnych co jest także kolejnym ułatwieniem, kolejnym atutem jest zapobieganie to wypożyczenia uszkodzonego auta co mogłoby skutkować zagrożeniem życia klienta bądź zwyczajnie naruszenie renomy firmy.

Należy zauważyć, że autor aplikacji powinien rozważyć rozbudowanie poszczególnych modułów w przypadku, gdy aplikacja zostanie wdrożona w wybranej firmie i zgłoszone zostanie zapotrzebowanie na poszczególne elementy.