Inżynieria
Oprogramowania

Dokumentacja Projektu

Aquadrom

Autorzy:
Anna Reichel
Magdalena Pąchalska
Sebastian Nalepka
Mateusz Ogiermann

27 stycznia 2015

Spis treści

1 Wstęp

1.1 Autorzy

Anna Reichel, Magdalena Pąchalska, Sebastian Nalepka, Mateusz Ogiermann
Informatyka sem. V
Wydział Matematyki Stosowanej
Politechnika Śląska

1.2 Wprowadzenie

Celem projektu było stworzenie aplikacji w związku z zaliczeniem przedmiotu Inżynieria Oprogramowania na V semestrze studiów inżynierskich na kierunku Informatyka. Do realizacji tego zadania przystąpiliśmy, dobierając się w 4-osobową grupę, w której rozdzielane były zadania, składające się na całość projektu.

1.3 Założenia

Założeniem naszego projektu było stworzenie aplikacji z bazą danych, która ma służyć w Parku Wodnym Aquadrom w Rudzie Śląskiej. Program ma zawierać funkcjonalności ułatwiające pracownikom Parku Wodnego wykonywanie swojej pracy, tzn:

- Przechowywanie i zarządzanie danymi pracowników,
- Obsługa harmonogramu godzin pracy,
- Tworzenie i zarządzanie notatkami ze zdarzeń.

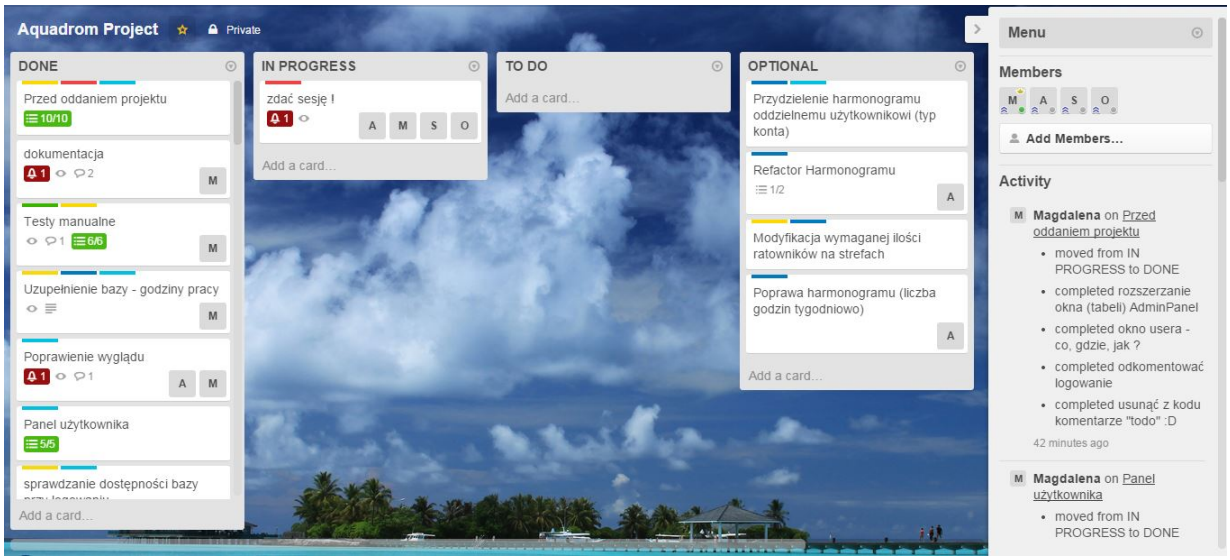
2 Realizacja projektu

2.1 Podział pracy

Rozpoczynając projekt grupa omówiła zagadnienia związane z planowaną aplikacją. Przedyskutowane zostały potrzeby użytkowników aplikacji, które przełożone zostały na planowane funkcjonalności aplikacji.
Na początku każdy z członków grupy wybrał wstępnie zadanie do realizacji, natomiast w trakcie tworzenia aplikacji zadania z listy "do zrobienia" były kolejno wykonywane przez autorów.

Przyjęliśmy metodykę pracy wzorowaną na SCRUM'ie – co tydzień lub dwa dostarczaliśmy nowych funkcjonalności. Ponadto projekt był stale na bieżąco konsultowany.

Zarządzanie wykonanymi i trwającymi czynnościami, rzeczami do zrobienia oraz listą opcjonalną dokonywało się dzięki aplikacji Trello dostępnej na www.trello.com. Tam także zadania były na bieżąco rozdzielane między członków grupy. Ostateczny wygląd naszej tablicy przedstawia rys. 1.



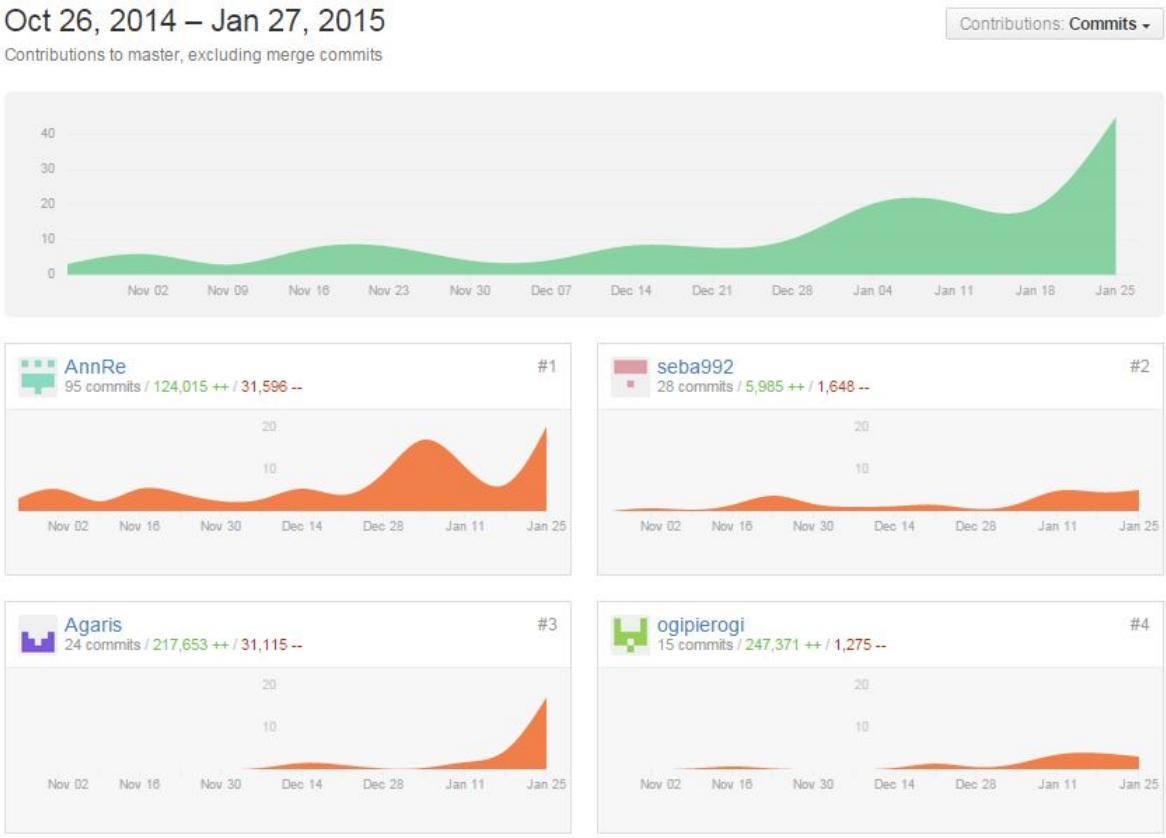
Rysunek 1: Podział zadań

Gdzie:

- A – Anna Reichel
- M – Magdalena Pąchalska
- S – Sebastian Nalepka
- O – Mateusz Ogiermann

Pełny wydruk wykonanych zadań wraz z podziałem pracy znajduje się w Załączniku nr 1 do Dokumentacji.

Udokumentowanie wykonanych zadań zostało też dokonane przez repozytorium stworzone na github.com, dzięki któremu możliwe było stworzenie statystyki widocznej na rys. 2.



Rysunek 2: Statystyka

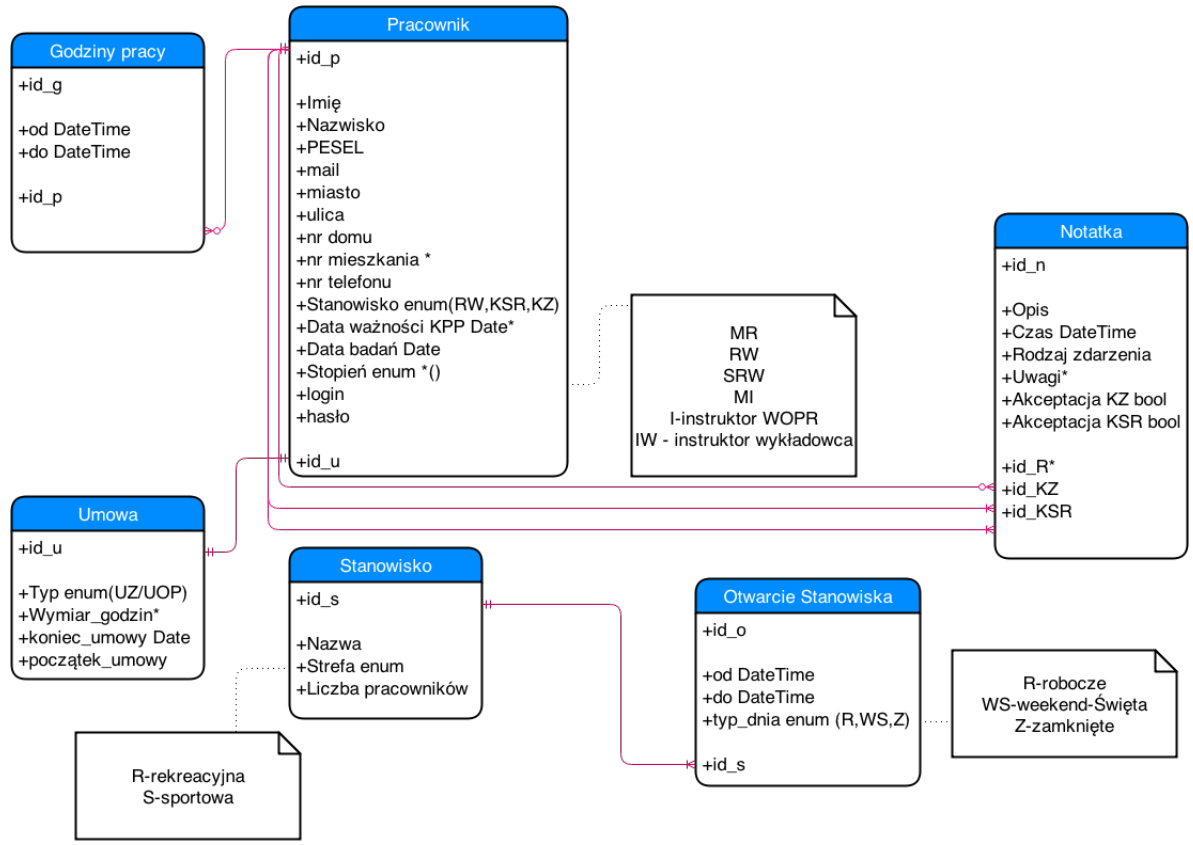
2.2 Aplikacje użyte przy realizacji projektu

Wykaz używanych przez nas aplikacji:

- Microsoft Visual Studio 2012
- Microsoft SQL Server Menagement Studio
- SourceTree (obsługująca repozytorium z github.com)
- Trello (trello.com)

2.3 Baza danych

Baza danych została stworzona w języku T-SQL przy pomocy aplikacji Microsoft SQL Server Menagement Studio. Struktura bazy została omówiona przez zespół na wstępie procesu realizacji projektu, czego wynikiem był schemat bazy przedstawiony na rys. 3.



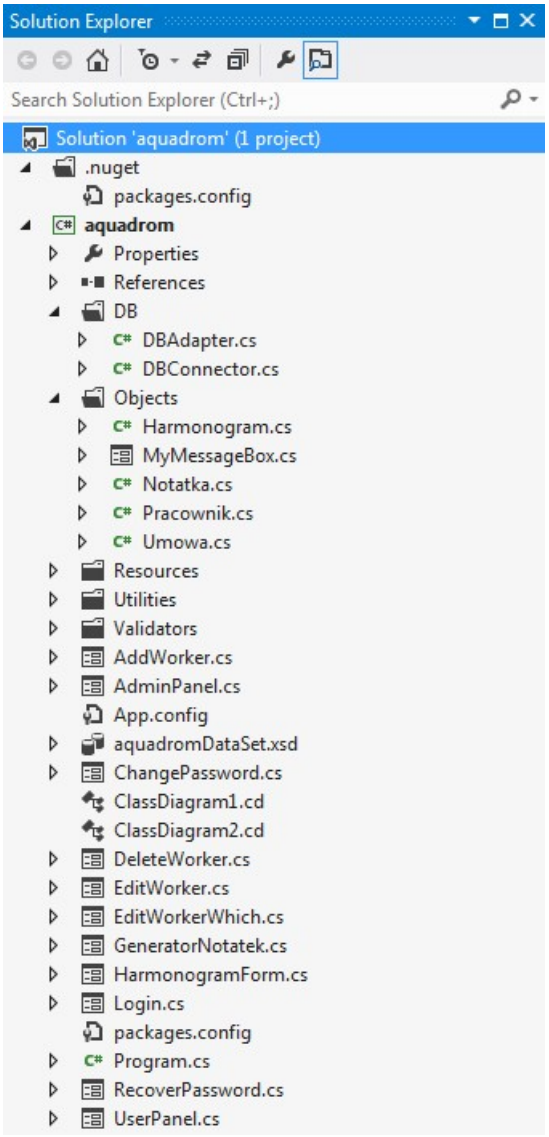
Rysunek 3: Schemat bazy danych

Jedynymi zmianami wprowadzonymi w trakcie tworzenia aplikacji było dodanie do tabeli *Pracownik* trzech kolumn: *Ostrzezenie_umowa*, *Ostrzezenie_KPP* i *Ostrzezenie_badania*. Związane są one z funkcją ostrzegania przez program o zbliżającej się dacie wygaśnięcia umowy, badań lub uprawnień pracownika. Dodała została również kolumna *Typ_konta*, z której program czerpie informację, czy zalogowany pracownik jest użytkownikiem czy administratorem.

2.4 Program

2.4.1 Podział na pliki

Podział projektu na pliki przedstawia rys. 4.



Rysunek 4: Podział aplikacji na pliki

2.4.2 Klasy i okna

2.4.2.1 Okno Harmonogramu (Harmonogram)

Okno odpowiedzialne za wyświetlanie harmonogramu wybranego miesiąca i roku na podstawie danych przechowywanych w bazie. Daje możliwość modyfikacji danych oraz walidacji czasu pracy w zależności od typu umowy. Sprawdza poprawność wprowadzanych danych i obecność odpowiednich pracowników na stanowiskach.

```
private string PoprawnieRozplanowanyDzien(DateTime day)
```

Sprawdza, czy w każdym momencie jest wystarczająca ilość pracowników oraz czy są KZ/KSR

```
private string pracownicyMajaOdpowiednieGodziny(DateTime time)
```

Kontroluje, czy w zależności od typu umowy pracownicy mają wypełnioną odpowiednią ilość godzin w harmonogramie.

```
public string poprawnieRozplanowanyMiesiac(DateTime time)
```

Sprawdza kolejne dni miesiąca pod kątem odpowiedniego rozplanowania.

```
private int GetNeededWorkersAmount(DateTime time)
```

Sprawdza ilu w danym czasie potrzebnych jest ratowników w pracy.

```
private string stanowiskaObsadzone(DateTime time)
```

Sprawdza czy w danym czasie na stanowiskach jest odpowiednia liczba ratowników, kierowników zmiany oraz kierowników sztabu ratunkowego.

```
private int GetNumberOfRescuresAtTime(DateTime time)
```

Zwraca liczbę ratowników na stanowistach w danym czasie.

```
private bool KZPresentAtTime(DateTime time)
```

Sprawdza czy w danym czasie obecny jest Kierownik Zmiany na pływalni.

```
private bool KSRandKZPresenceAtTime(DateTime time)
```

Sprawdza czy Kierownik Zmiany (KZ) i (KSR) obecni na stanowiskach w danym czasie.

```
public string Save()
```

Sprawdza, czy wszystkie godziny mają wprowadzone początek i koniec pracy i zapisuje zmiany do bazy.

```
public int GetColumnIndexForDate(DateTime date)
```

Zwraca numer kolumny, której tytuł zawiera zadaną datę.

```
private DateTime GetColumnDate(int columnIndex, int rowIndex)
```

Zwraca datę, umieszczoną w tytule tabeli dla komórki o danych współrzędnych.

```
private DateTime GetCellHourAtDate(DateTime oldDate, int columnIndex)
```

Uzupełnia wczytaną datę o dzień w zależności od kolumny dla której została wywołana metoda.

```
private DateTime GetCellDateTime(int columnIndex, int rowIndex)
```

Zwraca datę i godzinę dla danej komórki na podstawie nagłówka i wprowadzonej godziny rozpoczęcia lub zakończenia pracy.

```
private TimeSpan GetUserHoursAtDate(int rowIndex, DateTime date)
```

Zwraca liczbę godzin pracy danego pracownika w danym dniu.

```
public bool bothTimesAreReady(int columnIndex,int rowIndex)
```

Zwraca true jeżeli wypełnione są zarówno godzina rozpoczęcia i zakończenia pracy.

```
public bool onlyOneTimesAreReady(int columnIndex, int rowIndex)
```

Zwraca true, jeżeli wypełniona jest tylko jedna z dwóch godzin pracy (rozpoczęcia lub zakończenia).

```
public string ValidateCell(DataGridViewCellValidatingEventArgs e)
```

Sprawdza czy:

- dane zapisane są w formacie GG:mm

- godziny są z przedziału 8:00-22:00
- minuty podane są z dokładnością 15min

```
private string GetNazwisko(int rowIndex)
```

Pobiera nazwisko z tabeli harmonogramu w danym wierszu

```
private string GetImie(int rowIndex)
```

Pobiera imie z tabeli harmonogramu w danym wierszu

2.4.2.2 Harmonogram

Odpowiada za wyświetlanie siatki siatki godzin, obsługę zdarzeń wykorzystując metody klasy *Harmonogram.cs*. Metody pomocnicze:

```
private void FillFromDB()
```

Wypełnia tabelę danymi z bazy.

```
private int GetYearFromCombo()
```

Pobiera wybrany rok z listy rozwijanej.

```
private int GetMonthFromCombo()
```

Pobiera wybrany miesiąc z listy rozwijanej.

```
private void CreateDayColumns()
```

Inicjalizuje tabelę siatki, dodając kolumny dla każdego nia.

```
private void UpdateColumnsToDate()
```

Aktualizuje wyświetloną siatkędo wybranej daty.

```
private void HighlightCell(int row_i, int col_i)
```

Podświetla nieuzupełnioną komórkę w przypadku gdy uzupełniona jest tylko jedna część godzin dla danych osoby i dnia.

Obsługiwane zdarzenia:

```
private int GetFirstIndexOfPair(int columnIndex)
```

Ponieważ dzień miesiąca dzielony jest na godziny rozpoczęcia i zakończenia pracy, metoda sprawdza czy dana komórka jest rozpoczęciem, czy zakończeniem.

```
private void dataGridView1_CellValueChanged(object sender,
    DataGridViewCellEventArgs e)
```

Sprawdza poprawność wpisanych godzin, pod względem formatu oraz godzin otwarcia pływalni.

```
private void comboBoxYear_SelectedIndexChanged(object sender, EventArgs
    e)

private void comboBoxMonths_SelectedIndexChanged(object sender,
    EventArgs e)
```

Wywoływane po zmianie wartości w liście rozwijanej, powoduje aktualizację siatki godzin do wybranej daty.

```
private void buttonSave_Click(object sender, EventArgs e)
```

Po naciśnięciu przycisku zapisywania, sprawdzana jest poprawność, wyświetlane są ewentualne komunikaty i zapisywane są zmiany.

```
private void dataGridView1_CellValidated(object sender,
    DataGridViewCellEventArgs e)
```

Po usunięciu zawartości komórki, jej zawartość zamieniana jest na pusty ciąg.

```
private void btn_check_Click(object sender, EventArgs e)
```

Po naciśnięciu sprawdzana jest poprawność wypełnienia siatki godzin danego miesiąca.

```
private void HarmonogramForm_FormClosing(object sender,
    FormClosingEventArgs e)
```

Wywoływane jest po naciśnięciu przycisku zamykania. Jeśli zmiany nie zostały zapisane, wyświetlane jest okno dialogowe.

```
private void comboBoxMonths_MouseClick(object sender, MouseEventArgs e)
```

Naciśnięcie na linię rozwijanązmiany miesięcy. Przez zmianą sprawdzane jest, czy zapisane zostały zmiany.

```
private void dataGridView1_CellEndEdit(object sender,
    DataGridViewCellEventArgs e)
```

Zakończenie edycji komórki - sprawdzana jest poprawność formatu

2.4.2.3 Okno administratora (Admin Panel)

a) Główna tabela danych

b) Sprawdzenie ważności badań lekarskich, KPP oraz daty zakończenia umowy (podświetlanie)

c) Wysyłanie e-maila w chwili pojawienia się ostrzeżenia (zakończenie badań,KPP,umowy)

d) Sprawdzanie połączenia internetowego + stosowna informacja

```
public void AdminPanel_Load(object sender, EventArgs e)
```

Wypełnianie głównej tabeli zawierającej wszystkie potrzebne informacje dotyczące każdego pracownika oraz sprawdzane jest połączenie internetowe, o którym informacja umieszczona jest na pasku stanu.

```
private void UsuńToolStripMenuItem_Click(object sender, EventArgs e)
private void edytujUżytkownikaToolStripMenuItem_Click(object sender,
    EventArgs e)
private void PrzeglądajUżytkownikówToolStripMenuItem_Click(object
    sender, EventArgs e)
```

Po wybraniu z menu Zarządzaj pozycji 'Usuń użytkownika' lub 'Edytuj użytkownika' następuje wywołanie okna odpowiedniego okna, jeśli uprzednio żadne inne nie zostało włączone.

```
private void ColorCheckUser()
```

Funkcja sprawdza daty zakończenia badań, KPP oraz umowy kolejno wszystkich użytkowników i w chwili gdy zbliża się ich koniec, z 7 dniowym wyprzedzeniem informuje o tym. ID rekord tej osoby oraz kończąca się data zaznaczana jest na czerwono oraz wysyłana jest stosowna informacja mailowa informacja.

```
private void sendmail(int iterator, DateTime KPPdate, bool changeKPP,
    DateTime medicaldate, bool changemedi, DateTime concractdate, bool
    changeconcract )
```

W sytuacji zbliżającego zakończenia się daty badań, KPP lub umowy wysyłana jest wiadomość mailowa, gdy obecne jest połączenie internetowe oraz, gdy mail nie został już z tego samego powodu uprzednio wysłany.

```
private bool CheckInternetConnection()
```

Próbując połączyć się z stroną internetową Google.com sprawdzamy dostępność połączenia internetowego zwracając twierdzącą lub przeczącą odpowiedź.

2.4.2.4 Logowanie (Login)

a) Haszowanie hasła oraz sprawdzenia pary login:haslo z bazą danych

b) Sprawdzanie dostępności bazy oraz wystosowanie odpowiedniego komunikatu w chwili braku połączenia

```
public static String sha256_hash(String value)
```

Używając gotowej funkcji haszującej dla stringa podanego w jej argumencie uzyskujemy 256 bitowy skrót SHA – wszystkie hasła przetwarzane w programie są w postaciach niejawnych i tak też porównywane są ich zgodności.

```
private bool CheckBase()
```

Tworząc nowe tymczasowe połączenie sprawdzamy połączenie z bazą. W chwili negatywnego wyniku jej działania połączenie z bazą jest niemożliwe i wystosowywana jest odpowiednia dla użytkownika informacja.

```
private void LoginButton_Click(object sender, EventArgs e)
```

Kliknięcie przycisku zaloguj powoduje sprawdzenie dostępności bazy i w chwili jej obecności sprawdzane jest z dopasowanie wpisanej pary login:hasło z wszystkimi parami login:hasło znajdującymi się w bazie danych. W przypadku ich zgodności użytkownik przekierowany zostaje do odpowiedniego okna, które zależy od jego typu konta. W przypadku, gdy para nie została odnaleziona wystosowywana jest odpowiednia informacja o nieprawidłowym loginie lub haśle.

2.4.2.5 Dodawanie użytkownika (AddWorker)

a) Sprawdzanie poprawności zmienionych edytowanych danych

b) Utworzenie nowej umowy pracownika

c) Zapis wprowadzonych informacji do bazy danych

```
private void Add_Contract()
```

Dodaje umowę użytkownika do bazy danych.

```
private void Add_Employer()
```

Dodaje pracownika do bazy danych.

```
public string createPassword(int length)
```

Zwraca losowy ciąg znaków o podanej długości, składający się z małych i dużych liter oraz cyfr.

```
public void sendMail(string login, string haslo, string mail, string
    imie, string nazwisko)
```

Wysyła do pracownika wiadomość e-mail z danymi niezbędnymi do logowania.

2.4.2.6 Edycja użytkownika (EditWorker)

a) Wczytanie danych odpowiedniego pracownika oraz odpowiednia walidacja komórek

b) Sprawdzanie poprawności zmienionych edytowanych danych

c) Wprowadzenie edytowanej umowy i danych pracownika do bazy danych

```
private void EditWorker_Load(object sender, EventArgs e)
```

W czasie ładowania okna edycji użytkownika automatycznie wypełniane są wszystkie dane wybranego uprzednio użytkownika. Każda rozwijana lista zawiera wszystkie możliwe do wyboru wartości, każde pole tekstowe umożliwia wpisanie tylko danych we właściwym formacie i we właściwej długości. Sprawdzane są także wszystkie zależności, które mogą lub nie mogą wystąpić i zależnie od nich blokowane lub odblokowywane są możliwe do wyboru wartości.

```
private void TypUmowyComboBox_SelectedIndexChanged(object sender,
    EventArgs e)
```

W zależności od wybranego typu umowy blokowana lub odblokowywana jest do wyboru liczba godzin. W przypadku umowy zlecenia do bazy jako liczba godzin wysyłana jest wartość NULL.

```
private void StanowiskoUseraComboBox_SelectedIndexChanged(object
    sender, EventArgs e)
```

Dla stanowiska KZ data KPP oraz stopień nie obowiązuje. Funkcja ta w chwili wyboru KZ blokuje wyżej wymienione okna edycji.

```
private string TakeValue(DataTable dtlist, string what)
```

Funkcja zwracająca daną wartość kolumny z listy podanej w argumencie (tablica z jednym rekordem – danymi jednego pracownika)

```
private void EdytujUseraButton_Click(object sender, EventArgs e)
```

W chwili wprowadzenia wszystkich zmian po kliknięciu przycisku edytuj oraz potwierdzeniu wybranej opcji następuje dodatkowa walidacja Peselu, adresu e-mail oraz numeru telefonu. W chwili ich błędnego wprowadzenia wartości, użytkownik informowany jest co musi poprawić w celu poprawnej edycji. W sytuacji, gdy proces walidacji przejdzie pomyślnie wywoływana jest funkcja edytująca umowę EditConcart oraz pracownika EditEmployee. Gdy obie funkcje zostaną poprawnie wykonane edycja zakończona jest pomyślnie, w przeciwnym wypadku wyświetlana jest informacja o błędzie.

```
private bool EditEmployee()
```

Funkcja ta tworzy obiekt pracownik, który zawiera odpowiednio zwalidowane dane oraz dodaje je do bazy (edytowanie danych wybranego pracownika). W sytuacji, gdy aktualizacja danych przebiega niepomyślnie – wyświetlana jest stosowna informacja.

```
private bool EditContract()
```

Funkcja analogiczna do EditEmployee() z tą różnicą, iż tworzony obiekt dotyczy umowy wybranego użytkownika. Wszystkie dane poddane są walidacji a następnie aktualizowane są w bazie danych. W sytuacji, gdy operacja Update kończy się niepowodzeniem – wyświetlany jest komunikat. 4. Wybór pracownika do edycji (EditWhichWorker) a. Lista imion i nazwisk z przekazaniem parametru wybranego pracownika do edycji b. Wywołanie okna 'EditWorker' dla odpowiednio wybranego pracownika.

```
public void EditWorkerWhich_Load(object sender, EventArgs e)
```

Przy wywołaniu okna wyboru pracownika do edycji generowana jest lista wszystkich par 'Nazwisko Imię' oraz przekazana do rozwijanej listy.

```
private void ChooseButton_Click(object sender, EventArgs e)
```

W chwili kliknięcia Wybierz następuje wywołanie okna 'EditWorker' wybranego pracownika. Operacja ta wykonywana jest tylko w momencie, gdy uprzednio nie zostało otwarte już inne okno.

2.4.2.7 Usuwanie pracownika (DeleteWorker)

a) Lista imion i nazwisk z przekazaniem parametru wybranego pracownika do usunięcia

b) Usunięcie umowy wybranego pracownika

c) Usunięcie pracownika z bazy

```
public void DeleteWorker_Load(object sender, EventArgs e)
```

Przy wywołaniu okna wyboru pracownika do usunięcia generowana jest lista wszystkich par 'Nazwisko Imię' oraz przekazana do rozwijanej listy.

```
private void DeleteWorkerComboBox_SelectedIndexChanged(object sender,
    EventArgs e)
```

W chwili zmiany (wyboru) pracownika zapamiętywany jest jego numer ID oraz ID jego umowy.

```
private void DeleteButton_Click(object sender, EventArgs e)
```

W chwili kliknięcia Wybierz następuje wywołanie stosownego komunikatu (potwierdzenia dokonania wyboru). W chwili ponownej akceptacji ze strony użytkownika, wybrany przez niego pracownik jest usuwany z bazy danych (w kolejności usuń umowę, usuń pracownika) i okno zostaje automatycznie zamknięte.

2.4.2.8 Zmiana hasła użytkownika (changePassword)

```
public void changePassword()
```

Umożliwia zmianę hasła zalogowanego użytkownika.

2.4.2.9 Odzyskiwanie hasła (RecoverPassword)

```
public void newPassword()
```

Umożliwia przypomnienie hasła.

2.4.2.10 Generator notatek (generujNotatke)

```
public void generujNotatke()
```

Generuje dokument *.pdf z notatką-zdarzeniem.

```
private void CreateIfMissing(string path)
```

Tworzy folder „Notatki" jeśli nie istnieje lub został usunięty.

```
private void TekstNotatki_KeyDown(object sender, KeyEventArgs e)
```

Blokuje możliwość używania klawisza Enter.

2.4.2.11 Walidacja wprowadzanych danych (pracownikValidators)

```
public bool ValidatePesel(string pesel)
```

Sprawdza czy numer PESEL ma poprawny format.

```
private static string ObliczSumeKontrolna(string pesel)
```

Zwraca sumę kontrolną.

```
public bool ValidateNumber(string numer)
```

Sprawdza czy numer telefonu ma poprawny format.

```
public bool ValidateMail(string mail)
```

Sprawdza czy adres e-mail ma poprawny format.

```
private string OdwzorujDomene(Match match)
```

Zwraca domenę.

```
public string CaloscNaMale(string nazwa)
```

Zmienia wielkość liter w ciągu na małe.

```
public string PierwszyZnakNaDuzaLitere(string nazwa)
```

Zmienia wielkość pierwszej litery w ciągu na dużą.

```
public string ResztaZnakowNaMale(string litera)
```

Zmienia wielkość liter w ciągu na małe z pominięciem pierwszego znaku.

```
public string PoMyslnikuLubSpacji(string nazwa, string znaki,
    CultureInfo culture)
```

Zmienia wielkość liter w ciągu na małe z pominięciem pierwszego znaku oraz wielkość pierwszej litery w ciągu na dużą, po wystąpianiu znaku specjalnego.

```
public bool isNullOrNot(TextBox tekst, string nazwaPola)
```

Sprawdza czy pole tekstowe jest puste.

```
public bool isNullOrNot(string tekst, string nazwaPola)
```

Sprawdza czy podany ciąg znaków jest pusty.

```
public bool isNullOrNot(NumericUpDown tekst, string nazwaPola)
```

Sprawdza czy pole numeryczne jest puste.

```
public string deleteNumbers(TextBox tekst)
```

Usuwa cyfry z ciągu znaków.

```
public void tylkoLitery(KeyPressEventArgs e)
```

Umożliwia wpisywanie wyłącznie liter.

```
public void tylkoLiteryMyslnik(KeyPressEventArgs e)
```

Umożliwia wpisywanie wyłącznie liter i myślników.

```
public void tylkoLiteryMyslnikSpacja(KeyPressEventArgs e)
```

Umożliwia wpisywanie wyłącznie liter, myślników oraz spacji.

```
public void tylkoCyfryPlus(KeyPressEventArgs e)
```

Umożliwia wpisywanie wyłącznie liczb oraz plusów.

```
public void tylkoCyfry(KeyPressEventArgs e)
```

Umożliwia wpisywanie wyłącznie liczb.

```
public void tylkoCyfryLitery(KeyPressEventArgs e)
```

Umożliwia wpisywanie wyłącznie liter i liczb.

2.5 Testy

Poprawność działania aplikacji została przetestowana za pomocą testów automatycznych (przy użyciu pakietu NuGet). W trakcie powstawania programu były na bieżąco przeprowadzane testy manualne. Przed oddaniem projektu do oceny zostały przeprowadzone ostateczne testy manualne, potwierdzające poprawność zachowań aplikacji.

2.6 Bezpieczeństwo danych i aplikacji

Hasła użytkowników przechowywane w bazie danych programu są hashowane przy użyciu funkcji haszującej SHA-256.

Aplikacja poprzez walidację wprowadzanych danych została uodporniona na ataki typu SQL injection.

3 Podsumowanie

Wszystkie założenia projektu zostały spełnione. Praca w grupie umożliwiła nam zapoznanie się z aspektem tworzenia oprogramowania w zespole programistów. Niewielkie problemy z jakimi spotkaliśmy się podczas realizacji projektu zostały sprawnie rozwiązane, dzięki dobrej współpracy w grupie, a także dzięki naszemu repozytorium i jego obsłudze w programie SourceTree. Wspólnymi siłami osiągnęliśmy zamierzony cel – dostarczyliśmy aplikację, która może być z powodzeniem używana w Parku Wodnym Aquadrom w Rudzie Śląskiej.