Inżynieria
Oprogramowania

Dokumentacja Projektu

Aquadrom

Autorzy:
Anna Reichel
Magdalena Pąchalska
Sebastian Nalepka
Mateusz Ogiermann

25 stycznia 2015

Spis treści

1 Wstęp

1.1 Autorzy

Informatyka sem. V
Wydział Matematyki Stosowanej
Politechnika Śląska

1.2 Wprowadzenie

2 Część I – dla użytkowników

2.1 Opis programu

2.1.1 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 DodajUżytkownikówToolStripMenuItem_Click(object sender,
    EventArgs e)
```

CZĘŚĆ OGIEGO

```
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.

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

CZĘŚĆ OGIEGO

2.1.2 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 haszującej1 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.

2.1.3 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. 5. 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.

2.1.4 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 .

```
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-/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

2.2 Minimalne wymagania sprzętowe

2.3 Instrukcja obsługi

3 Część II – dla specjalistów

3.1 Program

3.1.1 Podział na pliki

3.1.2 Klasy

3.2 Baza danych

3.2.1 Struktura bazy danych

4 Wnioski

4.1 Zastosowania programu

4.2 Propozycje rozszerzeń programu