

PROJEKT INFORMATYCZNY

ORGANIZER

APLIKACJA WYKONANA W TECHNOLOGII ADOBE FLEX
Z WYKORZYSTANIEM LOKALNEJ BAZY DANYCH SQLITE

simplercode.com

Opublikowano na licencji

[Creative Commons Attribution-NoDerivs 3.0 Poland License](https://creativecommons.org/licenses/by-nd/3.0/pl/)

2008 / 2009

Spis treści

1. Cel i zakres pracy.....	2
2. Specyfikacja wymagań.....	2
2.1. Przypadki użycia.....	2
2.2. Wymagania funkcjonalne i pozafunkcjonalne.....	5
3. Projekt bazy danych.....	5
3.1. Opis tabel.....	5
3.2. Definicje tabel.....	6
4. Interfejs użytkownika.....	8
5. Testowanie interfejsu.....	9
6. Informacje dodatkowe.....	12

1. Cel i zakres pracy

Celem pracy jest stworzenie elektronicznego terminarza, czyli programu komputerowego umożliwiającego użytkownikowi prowadzenie osobistych notatek, tworzenie tygodniowych planów zajęć oraz gromadzenie danych adresowych znajomych.

Przystąpienie do pracy nad programem wiązało się z koniecznością stworzenia specyfikacji wymagań jakie program powinien spełniać. Wykorzystano tu metodę przypadków użycia oraz zastosowano podział na wymagania funkcjonalne i pozafunkcjonalne. Następnie przystąpiono do zaprojektowania schematu bazy danych służącej do gromadzenia danych użytkownika. Kolejnym etapem było zaprogramowanie aplikacji posiadającej interfejs graficzny umożliwiający korzystanie z bazy danych. Końcowym etapem była kontrola jakości wykonanego programu - przetestowanie wszystkich funkcji interfejsu graficznego.

2. Specyfikacja wymagań

Specyfikacja wymagań określa szczegółowo funkcje jakie przyszły program powinien posiadać, aby spełnić założony cel. Jest to forma kontraktu między programistą a przyszłym użytkownikiem programu.

2.1. Przypadki użycia

Metoda przypadków użycia polega na postawieniu się w rolę użytkownika przyszłego

systemu informatycznego i określeniu wykonywanych przez niego czynności będących przedmiotem rozwiązywanego przez system problemu.

UC1. Zapisywanie notatki.

Atrybuty:

Główny aktor: Użytkownik

Priorytet: Wysoki

Źródło: Michał

Główny scenariusz:

1. Użytkownik pragnie zapisać notatkę do terminarza.
2. Użytkownik otwiera terminarz w sekcji przeznaczonej na notatki.
3. Użytkownik wyszukuje stronę oznaczoną odpowiednią datą.
4. Użytkownik wpisuje notatkę w wolnym miejscu na stronie.

UC2. Tworzenie tygodniowego planu godzin.

Atrybuty:

Główny aktor: Użytkownik

Priorytet: Wysoki

Źródło: Michał

Główny scenariusz:

1. Użytkownik pragnie zapisać tygodniowy rozkład zajęć.
2. Użytkownik otwiera terminarz na przeznaczonej do tego tabeli.
3. Każdy wiersz tabeli przypisany jest jednej godzinie, a każda kolumna przypisana jest jednemu dniowi w tygodniu.
4. Użytkownik wpisuje zajęcia w odpowiednie kolumny i wiersze tabeli.

UC3. Odszukiwanie zapisanego adresu.

Atrybuty:

Główny aktor: Użytkownik

Priorytet: Wysoki

Źródło: Michał

Główny scenariusz:

1. Użytkownik pragnie odnaleźć adres znajomego, zapisany wcześniej w terminarzu.
2. Użytkownik otwiera terminarz w sekcji przeznaczonej do wpisywania adresów.

3. Korzystając z alfabetycznego indeksu umieszczonego na każdej ze stron, użytkownik odnajduje stronę z odpowiednią literą.
4. Użytkownik przegląda kolejno wpisy na otworzonej stronie. W zależności od rodzaju kontaktu (praca/rodzina) użytkownik sprawdza górną lub dolną część strony.
5. Użytkownik odnajduje wiersz z danymi poszukiwanej osoby.

UC4. Sprawdzanie dnia tygodnia na podstawie daty.

Atrybuty:

Główny aktor: Użytkownik

Priorytet: Średni

Źródło: Michał

Główny scenariusz:

1. Użytkownik mając dany rok, miesiąc i numer dnia w miesiącu, pragnie sprawdzić jaki to będzie dzień w tygodniu.
2. Użytkownik otwiera terminarz w sekcji w której znajduje kalendarz na dany rok.
3. Kalendarz składa się z 12 tabel z których każda przypisana jest do kolejnego miesiąca. Każda tabela składa się z 7 kolumn odpowiadających kolejnym dniom tygodnia. W każdej kolumnie wpisane numery dni w miesiącu.
4. Użytkownik odnajduje odpowiedni miesiąc a następnie sprawdza w której kolumnie znajduje się poszukiwany przez niego numer dnia.

UC5. Sprawdzanie wydarzeń wpisanych do terminarza.

Atrybuty:

Główny aktor: Użytkownik

Priorytet: Średni

Źródło: Michał

Główny scenariusz:

1. Użytkownik pragnie sprawdzić czy na dany dzień w roku wpisane są notatki.
2. Użytkownik otwiera terminarz w sekcji w której każda strona przeznaczona jest jednemu dniowi w roku.
3. Użytkownik na podstawie daty odszukuje stronę z notatkami na dany dzień.
4. Każdy wiersz na stronie przypisany jest do konkretnej godziny - użytkownik przegląda kolejno wiersze na znalezionej stronie.

2.2. Wymagania funkcjonalne i pozafunkcjonalne

Metoda określania wymagań funkcjonalnych i pozafunkcjonalnych polega na wyszczególnieniu funkcji, które program powinien wykonywać (wymagania funkcjonalne) oraz opisanu sposobu w jaki powinien je wykonywać (wymagania pozafunkcjonalne). Metoda ta w porównaniu z przypadkami użycia, charakteryzuje się mniejszą czytelnością. Trudniej jest za jej pomocą odzwierciedlić ogólny kształt systemu, ponieważ skupia się na wyróżnianiu konkretnych, odrębnych funkcji, które przyszedł program powinien dostarczać.

Wymagania funkcjonalne:

- wprowadzanie notatek tekstowych przypisywanych do konkretnej daty;
- przeglądanie istniejących wpisów na dany dzień, miesiąc, rok;
- edytowanie tekstu danej notatki oraz przypisanej do niej daty;
- przeglądanie alfabetycznego spisu zapisanych kontaktów;
- zmienianie wprowadzonych wcześniej danych adresowych;
- tworzenie planu tygodniowego z podziałem na godziny;
- możliwość zarządzania wieloma planami zajęć;
- wyświetlanie aktualnego kalendarza na dany miesiąc/rok;

Wymagania pozafunkcjonalne:

- możliwość wprowadzania danych bez uprzedniego przeszkolenia użytkownika;
- małe zużycie zasobów komputera, objętość programu rzędu kilku MB;
- zachowanie integralności gromadzonych danych;
- ładny i intuicyjny interfejs graficzny programu;
- dobra responsywność - brak oczekiwania na odpowiedź programu;

3. Projekt bazy danych

3.1. Opis tabel

Baza danych składa się z następujących tabel:

- **notatki** - tabela przeznaczona do gromadzenia notatek na dany dzień;
- **plany** - tabela przeznaczona do przechowywania informacji na temat tygodniowych zajęć;

- **zajecia** - każdy wiersz tabeli będzie zawierał dane na temat zajęcia przypisanego do danego planu; jednoznacznym identyfikatorem konkretnego zajęcia na planie godzin jest jego godzina, numer dnia w tygodniu oraz plan, do którego jest przypisane dane zajęcia; kluczem podstawowym tej tabeli będzie klucz złożony;
- **kontakty** - tabela ta przeznaczona jest do gromadzenia danych adresowych osób;
- **rodzajeKontaktow** - tabela przeznaczona do gromadzenia informacji o grupach kontaktów;

3.2. Definicje tabel

Definicje wszystkich tabel umieszczone zostały w obrębie pliku Creation.mxml. Zdeklarowane w nim zmienne, są następnie wykorzystane w kodzie źródłowym odpowiedzialnym za stworzenie bazy przy pierwszym uruchomieniu aplikacji.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:UIComponent xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:String id="notatki">
    CREATE TABLE IF NOT EXISTS notatki (
      id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL ,
      tresc TEXT NOT NULL ,
      data DATETIME NOT NULL ,
      utworzono DATETIME NOT NULL ,
      zaktualizowano DATETIME NULL)
  </mx:String>

  <mx:String id="plany">
    CREATE TABLE IF NOT EXISTS plany (
      id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL ,
      nazwa VARCHAR(30) NOT NULL ,
      opis TEXT NULL)
  </mx:String>

  <mx:String id="rodzajeKontaktow">
    CREATE TABLE IF NOT EXISTS rodzajeKontaktow (
      id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL ,
      nazwa VARCHAR(20) NULL ,
      opis TEXT NULL)
  </mx:String>
```

```

<mx:String id="kontakty">
CREATE TABLE IF NOT EXISTS kontakty (
  id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL ,
  imie VARCHAR(30) NOT NULL ,
  nazwisko VARCHAR(30) NOT NULL ,
  email VARCHAR(45) NULL ,
  miejscowosc VARCHAR(45) NULL ,
  adres VARCHAR(45) NULL ,
  telcom VARCHAR(7) NULL ,
  telkom VARCHAR(9) NULL ,
  rodzaj INTEGER NOT NULL ,
  notka TEXT NULL ,
  FOREIGN KEY (rodzaj) REFERENCES rodzajeKontaktow (id)
)
</mx:String>

```

```

<mx:String id="zajecia">
CREATE TABLE IF NOT EXISTS zajecia (
  opis TEXT NOT NULL ,
  godzina INTEGER NOT NULL ,
  dzien INTEGER NOT NULL ,
  plan INTEGER NOT NULL ,
  PRIMARY KEY (dzien, godzina, plan) ,
  FOREIGN KEY (plan) REFERENCES plany (id)
)
</mx:String>

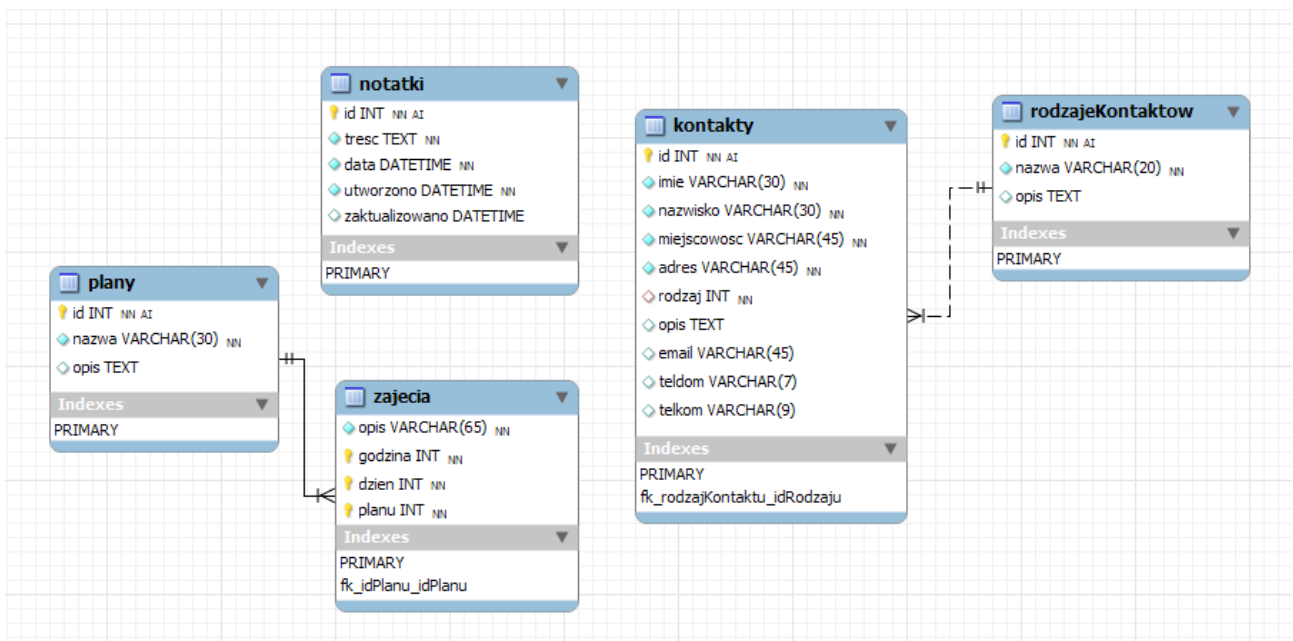
```

```

</mx:UIComponent>

```

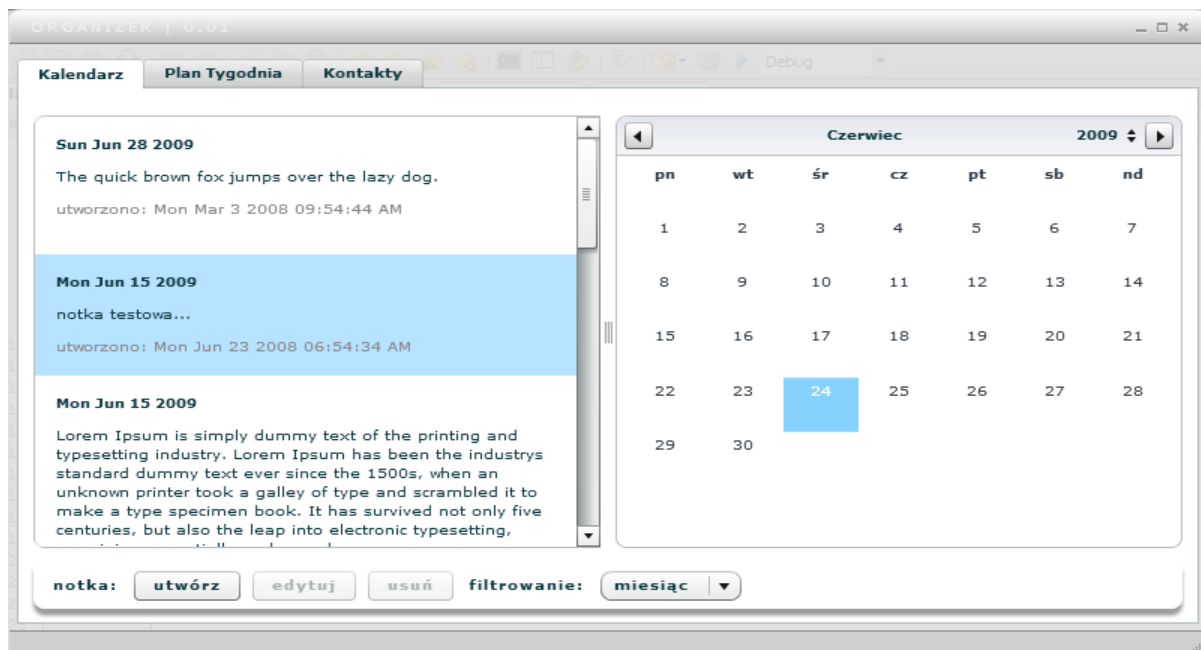
Schemat graficzny tabel.



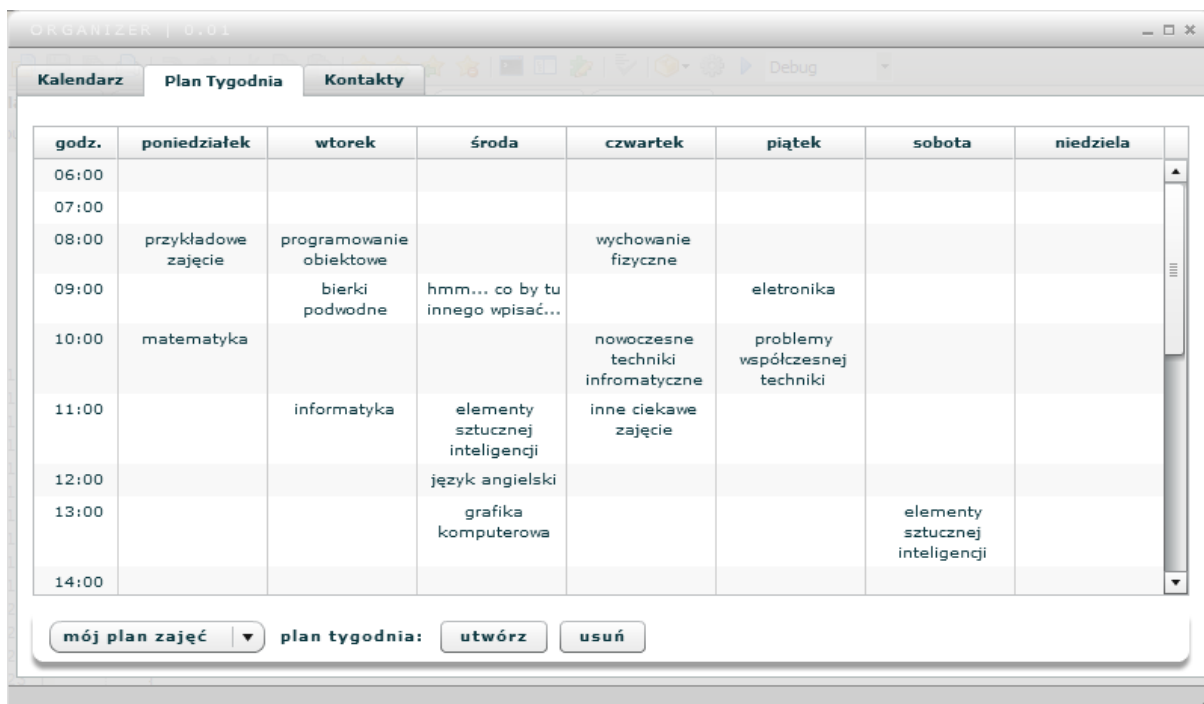
4. Interfejs użytkownika

Interfejs użytkownika zaprogramowano z wykorzystaniem standardowych komponentów środowiska Adobe Flex takich jak np. DateChooser, DataGrid, List, TabPanel, Form. Pomiędzy trzema głównymi ekranami aplikacji użytkownik nawiguje korzystając z zakładek umieszczonych pod paskiem tytułowym okna.

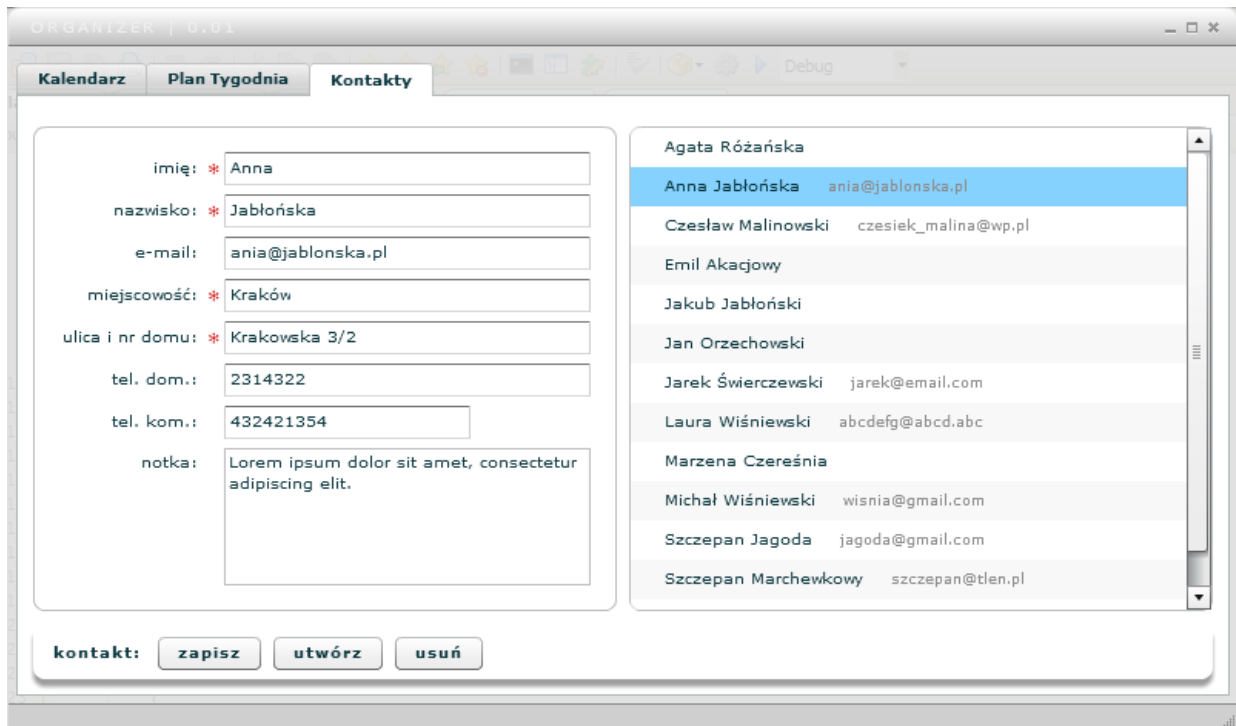
A. Kalendarz



B. Plan tygodnia.



C. Kontakty



5. Testowanie interfejsu

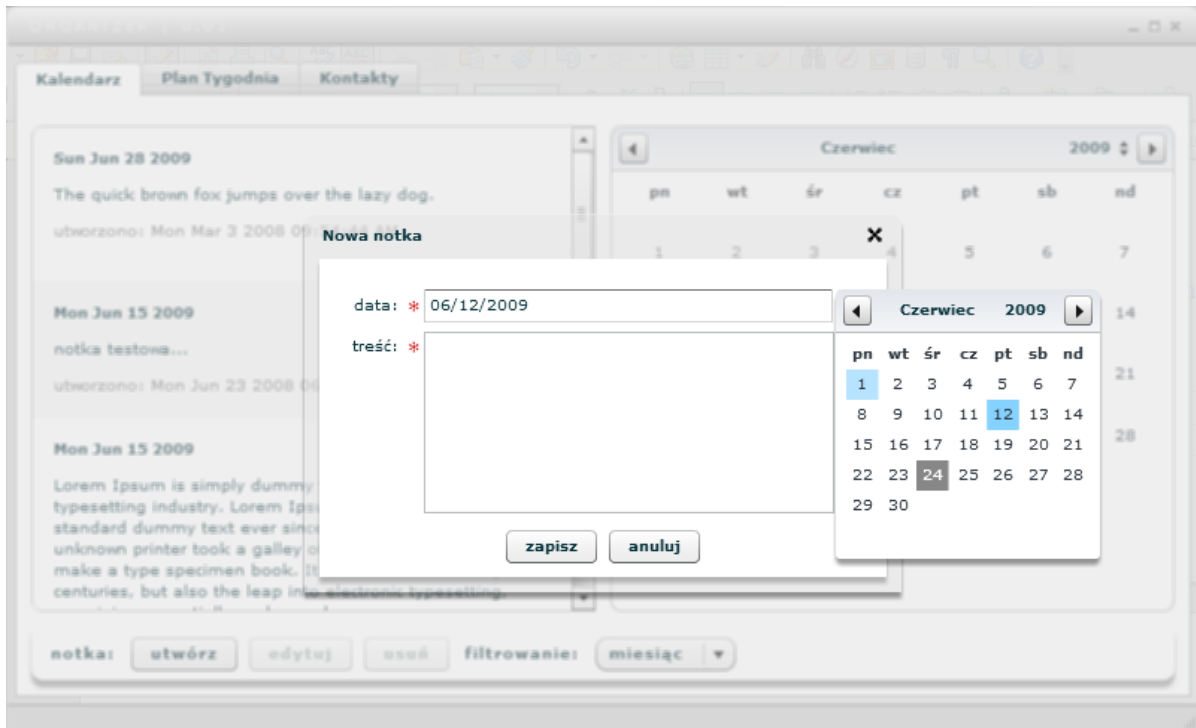
Jednym z ostatnich etapów pracy nad projektem, było przetestowanie interfejsu aplikacji. Testowanie ma na celu wykrycie potencjalnych błędów w działaniu programu, np. akceptację niepoprawnie sformatowanych danych w formularzu dodawania nowego kontaktu.

W celu zachowania integralności bazy danych niezbędne było zastosowanie walidacji we wszystkich polach służących do wprowadzania danych. Tam gdzie było to niezbędne posłużono się komponentem `StringValidator`, który umożliwia wprowadzenie ograniczeń na pola tekstowe, np. akceptację tylko i wyłącznie łańcuchów o długości z określonego przedziału. Oto przykładowa deklaracja komponentu sprawdzającego nazwę nowo wprowadzonego planu godzin:

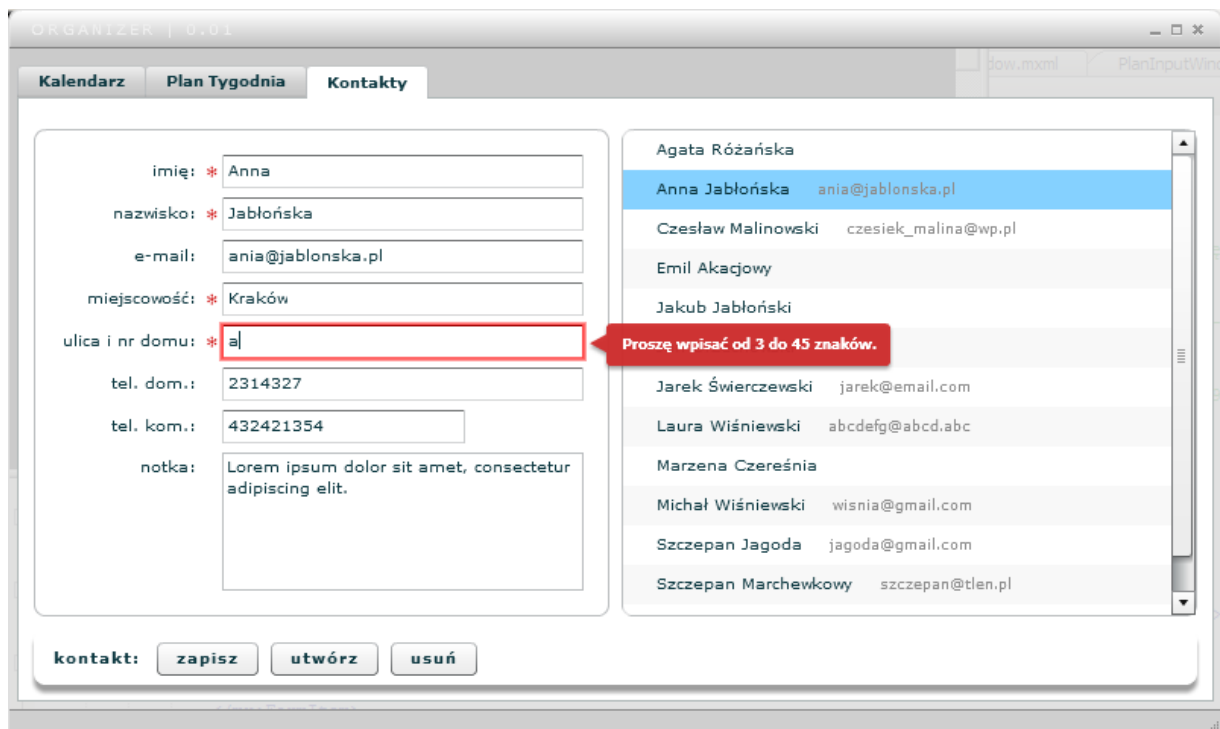
```
<mx:StringValidator id="nameValidator" source="{nameInput}" property="text" minLength="6"
    maxLength="30" tooShortError="{lengthError}" tooLongError="{lengthError}" required="true"
    requiredFieldError="{requiredError}" trigger="{saveButton}" triggerEvent="click"
    valid="dispatchEvent(new Event('savePlan'));" />
```

Pola testowano również na wypadek wprowadzenia wartości granicznych. Interfejs zaprojektowano w taki sposób, aby nie było możliwe wprowadzenie nieprawidłowych danych.

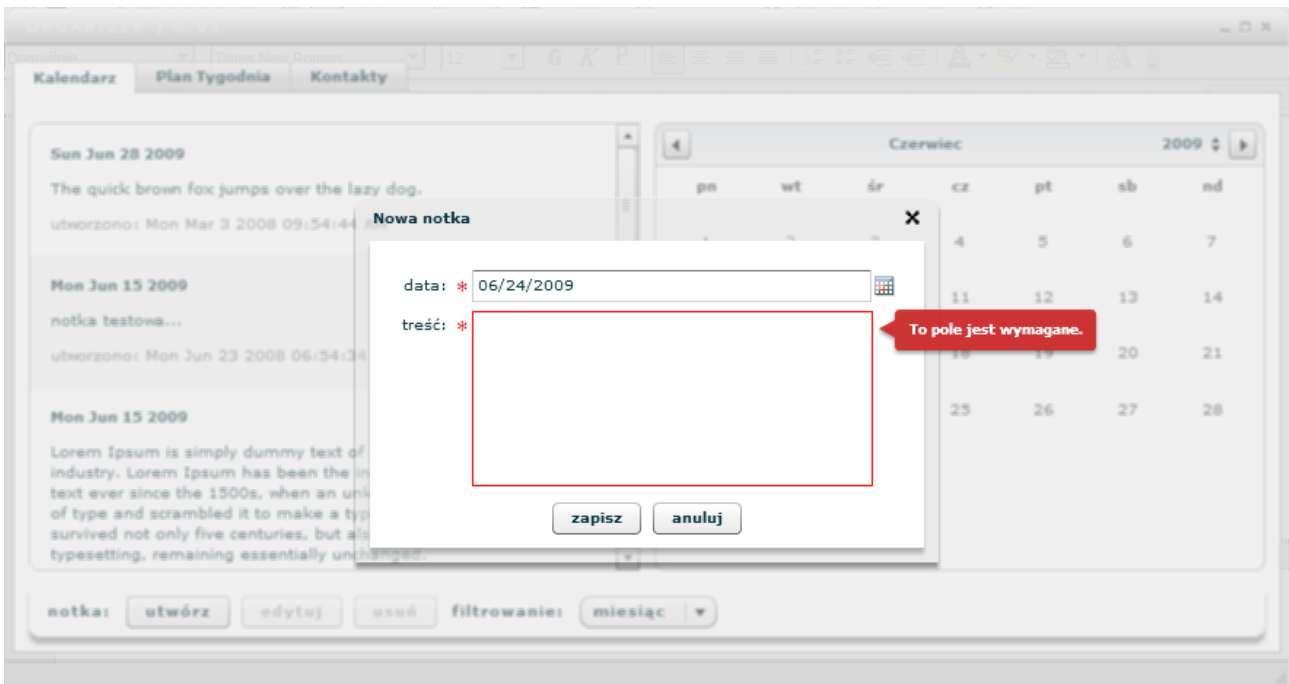
Przykładem takiego działania jest pole wprowadzania daty dla nowej notatki w kalendarzu. Jedynym sposobem wprowadzenia nowej wartości jest skorzystanie z podręcznego kalendarza, dzięki czemu użytkownik nie musi zwracać uwagi na prawidłowy format wprowadzanej daty:



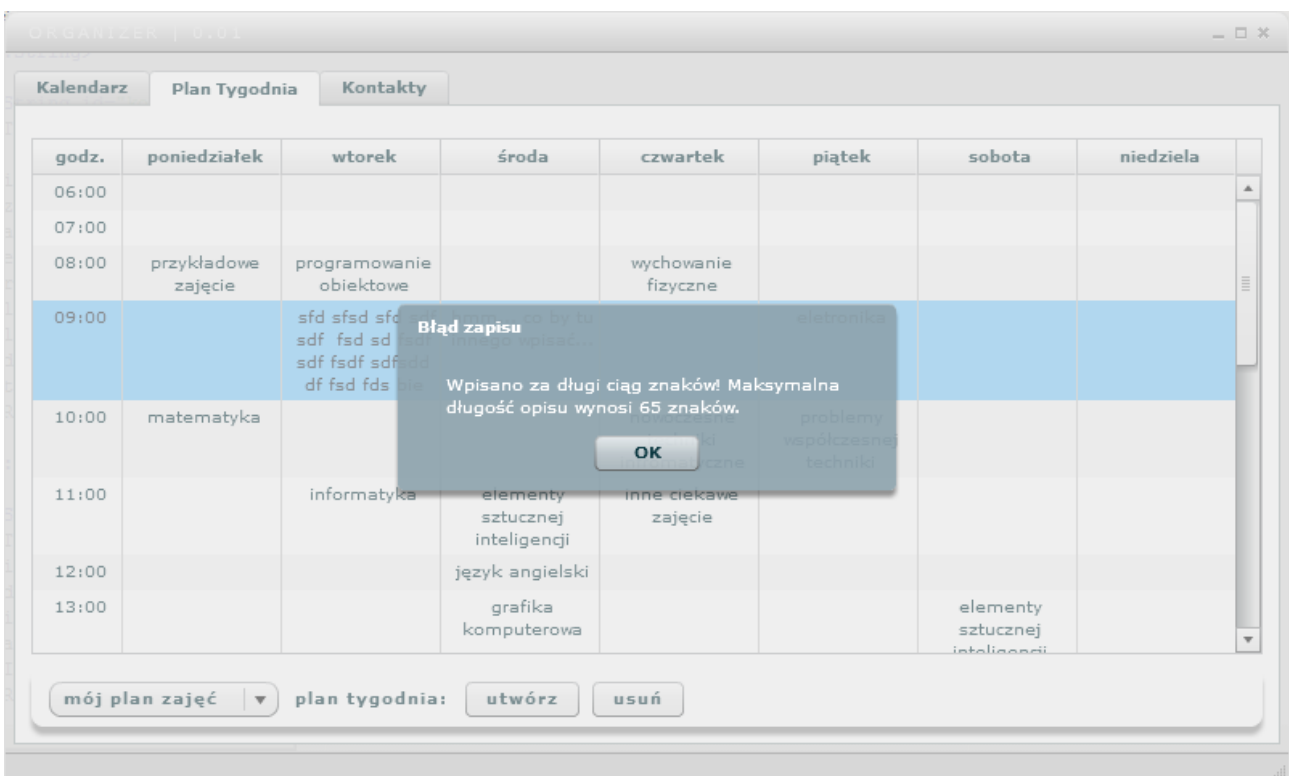
W przypadku wprowadzenia nie prawidłowych danych, użytkownik informowany jest o potrzebie poprawienia ich przyjaznym komunikatem:



Informacja o nie wypełnieniu wymaganego pola:



Okno informujące o wpisaniu zbyt wielu znaków jako opis na planie tygodnia:



6. Informacje dodatkowe

Na ilustracji obok przedstawiona jest struktura załączonego na płycie projektu. Aplikacja skompilowana została za pomocą Flex SDK 3.4.0 oraz AIR SDK 1.5.1. Jako edytor kodu źródłowego wykorzystano IDE FlashDevelop 3.0.0. Bazę danych zaprojektowano w programie MySQL Workbench 5.0 OSS.

Aplikacja dostarczona jest w formie pliku instalacyjnego z rozszerzeniem '.air'. Środowisko uruchomieniowe Adobe Air znajduje się na dołączonej płycie z projektem.

W trakcie pierwszego uruchomienia baza danych automatycznie zostanie wypełniona przykładowymi danymi. Baza danych w formie pliku na dysku twardym komputera tworzona jest w następującej lokalizacji (system operacyjny MS Windows XP):

```
C:\Documents and Settings\{użytkownik}\Application Data\com.michal.Organizer\Local Store\data.db.
```

