Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Diář

2017

Petr Úlehla

Vedoucí práce: RNDr. Arnošt Večerka

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Petr Úlehla
Název práce:	Diář
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2017
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	RNDr. Arnošt Večerka
Počet stran:	44
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Petr Úlehla
Title:	Diary
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2017
Study field:	Applied Computer Science, full-time form
Supervisor:	RNDr. Arnošt Večerka
Page count:	44
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

K této práci byla vytvořena aplikace reprezentující diář. Je určena pro běžné uživatele, kteří chtějí mít své plány a kontakty na jednom místě. Při realizaci byly použity různé programovací jazyky pro webová prostředí, zejména HTML, CSS, JavaScript a C#.

Synopsis

To this work was created application representing a diary. This application is intended for ordinary users, which want to have their plans and contacts in one place. For realization was used different programming languages for websites, especially HTML, CSS, JavaScript and C#.

Klíčová slova: web; webové technologie; HTML; CSS; JavaScript; C#

Keywords: website; web technology; HTML; CSS; JavaScript; C#

Mé poděkováné patří vedoucímu práce, RNDr. Arnoštu Večerkovi, za pomoc při vytváření této práce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

Seznam zdrojových kódů

1 Úvod

Cílem této práce bylo vytvořit aplikaci reprezentující diář. Je vytvořena pro běžné uživatele a tedy i osobní použití. Taková aplikace by měla zaznamenávat různé plány (akce) uživatelů a vytvořit jednoduchý přehled o tom, co mají v plánu. Zároveň musí uchovávat různé kontakty.

1.1 Zásady pro vypracování

Povinné požadavky:

- Kalendář
- Vkládání profilu uživatele
- Aktualizace profilu
- Evidence akcí (plánů)
- Vkládání plánu činností (akcí)
- Členění plánů (akcí) dle kategorií
- Upozornění na časové kolize akcí
- Uchovávání kontaktů

Volitelné požadavky:

- Správa kontaktů
- Posílání zpráv kontaktům

1.2 Typ aplikace

Jelikož nebylo specifikováno, o jaký druh aplikace se má jednat, bylo tedy na mém výběru, jaký typ aplikace si zvolit. Tato aplikace šla naprogramovat více způsoby. Jednoznačnou funkcí této aplikace by měl být nějaký databázový systém pro uchovávání dat. Nejdříve bylo zapotřebí si zvolit typ aplikace, který by nejlépe vyhovoval zadaným požadavkům. Jednoznačnou funkcí této aplikace by měl být nějaký databázový systém pro uchovávání dat. Na výběr jsem měl ze tří možností. První z nich byla klasická aplikace (program) pro stolní počítače. Bohužel program pro stolní počítače není vhodný, jelikož diář by měl být přístupný odkudkoliv. Uživatel by měl být tedy schopen dostat se do této aplikace, i když nemá přístup ke stolnímu počítači. Další variantou byla mobilní aplikace. Jelikož je více druhů těchto aplikací (pro mobilní systémy Android, Windows Phone, iOs,...) a zároveň by tyto aplikace měli mít přístup k internetu, zvolil jsem si tedy třetí možnost a to byla webová aplikace.

Webová aplikace je přístupná ze všech zařízení, co mají přístup k internetu. Jedná se o složitější návrh webové stránky s vlastním systémem, který tuto stránku ovládá. Druhou věcí tedy bylo, zvolit si takový jazyk (technologii), ve kterém tuto aplikaci naprogramuji. Jelikož jsem chtěl využít nějaké nové technologie s velkým potenciálem do budoucna, zvolil jsem si technologii ASP.NET s architekturou MVC (Model-View-Controller). Tato technologie využívá kombinaci několika programovacích jazyků, které spolu vytváří plně funkční web. Mezi tyto programovací jazyky, využívané v této technologii, patří C#, HTML, CSS, JavaScript, XML či různé rozšíření (knihovny, metody...) těchto jazyků jako Razor, jQuery, AJAX.

2 Teoretický úvod

Jelikož cílem této práce bylo vytvořit aplikaci, nikoliv algoritmus, je tu teoretický úvod o technologiích, metodách a programovacích jazycích, které jsem použil při vytváření této aplikace.

2.1 Technologie

2.1.1 ASP.NET

ASP.NET je součást **.NET Frameworku** pro tvorbu webových aplikací a služeb. Je nástupcem technologie ASP (Active Server Pages) a přímým konkurentem JSP (Java Server Pages). Ačkoliv název ASP.NET je odvozen od starší technologie pro vývoj webů ASP, obě technologie jsou velmi odlišné. ASP.NET je založen na CLR (Common Language Runtime), který je sdílen všemi aplikacemi postavenými na .NET Frameworku. Programátoři tak mohou realizovat své projekty v jakémkoliv jazyce podporujícím CLR, např. Visual Basic .NET, JScript.NET, **C#**, Managed C++, ale i mutace Perlu, Pythonu a další. Aplikace založené na ASP.NET jsou také rychlejší, neboť jsou předkompilovány do jednoho či několika málo DLL souborů, na rozdíl od ryze skriptovacích jazyků, kde jsou stránky při každém přístupu znovu a znovu parsovány. [1]

Koncept ASP.NET WebForms ulehčuje programátorům přechod od programování klasických aplikací pro Windows do prostředí webu: stránky jsou poskládány z objektů, ovládacích prvků (Controls), které jsou protějškem ovládacích prvků ve Windows. Při tvorbě webových stránek je tedy možné používat ovládací prvky jako tlačítko (Button), nápis (Label) a další. Těmto prvkům lze přiřazovat určité vlastnosti, zachytávat na nich události atd. Tak, jako se ovládací prvky pro Windows samy kreslí do formulářů na obrazovku, webové ovládací prvky produkují HTML kód, který tvoří část výsledné stránky poslané do klientova prohlížeče. [1]

2.1.2 Model-view-controller

Model-view-controller (**MVC**) (někdy také nesprávně označovaná jako Model-2) je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní. MVC je často chápán jako **návrhový vzor**, nicméně se týká architektury aplikací mnohem více než klasický návrhový vzor. [2]

Obecně řečeno, vytváření aplikací s využitím architektury MVC vyžaduje vytvoření tří komponent, mezi které patří:

- **Model** (model), což je doménově specifická reprezentace informací, s nimiž aplikace pracuje.
- **View** (pohled), který převádí data reprezentovaná modelem do podoby vhodné k interaktivní prezentaci uživateli.

- **Controller** (řadič), který reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu.

Komponenty řadič a pohled jsou ve standardním rozdělení vrstev na prezentační, doménovou a datovou obvykle zařazovány jako prezentační vrstva. V MVC je tato prezentační vrstva rozdělena mezi komponenty řadič a pohled, nicméně nejdůležitější rozdělení je mezí prezentací a doménovou vrstvou. [2]

Ačkoliv může být koncept MVC realizován různým způsobem, obecně platí tento princip: [2]

1. Uživatel provede nějakou akci v uživatelském rozhraní (např. stiskne tlačítko).

2. Řadič obdrží oznámení o této akci z objektu uživatelského rozhraní.

3. Řadič přistoupí k modelu a v případě potřeby ho zaktualizuje na základě provedené uživatelské akce (např. zaktualizuje nákupní košík uživatele).

4. Model je pouze jiný název pro doménovou vrstvu. Doménová logika zpracuje změněná data (např. přepočítá celkovou cenu, daně a expediční poplatky pro položky v košíku). Některé aplikace užívají mechanizmus pro perzistentní uložení dat (např. databázi). To je však otázka vztahu mezi doménovou a datovou vrstvou, která není architekturou MVC pokryta.

5. Komponenta pohled použije zaktualizovaný model pro zobrazení zaktualizovaných dat uživateli (např. vypíše obsah košíku). Komponenta pohled získává data přímo z modelu, zatímco model nepotřebuje žádné informace o komponentě View (je na ní nezávislý). Nicméně je možné použít návrhový vzor pozorovatel, umožňující modelu informovat jakoukoliv komponentu o případných změnách dat. V tom případě se komponenta view zaregistruje u modelu jako příjemce těchto informací. Je důležité podotknout, že řadič nepředává doménové objekty (model) komponentě pohledu, nicméně jí může poslat příkaz, aby svůj obsah podle modelu zaktualizovala.

6. Samotnému konečnému zobrazení výsledku uživateli ještě může u webaplikací předcházet odpověď ze serveru na klienta, aby si ihned vyžádal obnovení stránky (client side redirect, životnost 0, takže okamžitý): Tím je zaručeno, že při obnovení stránky uživatelem (refresh, F5 v prohlížeči) nevyvolá na serveru požadovanou akci opakovaně, ale že se jedná pouze o obnovení pohledu, nyní už bez požadavku na změnu dat (modelu). Účelem je změna URL a dat http requestu, aby poslední v řadě již nebyl „server-side data-affecting“ (ovlivňující model), ale pouze „read-only“ (pouhé zobrazení). Celý tento client-refresh (změna URL) se děje automaticky a bez povšimnutí uživatelem. Uživatelské rozhraní čeká na další akci uživatele, která celý cyklus zahájí znovu.

[2]

2.1.3 ASP.NET MVC

Tento **framework** umožňuje tvorbu webových aplikací podle softwarové architektury Model-view-controller. ASP.NET MVC má představovat alternativu oproti **WebForms**. Na rozdíl od WebForms aplikace vytvořené pomocí ASP.MVC nevyžadují ViewState a dají se snadněji testovat. Dalším rozdílem MVC oproti WebForms je nezávislost na Javascriptu (událostní model WebForms javasript vyžaduje) a logika bližší klasickému komunikačnímu modelu webu. [1]

2.2 Programovací jazyky

2.2.1 HTML

HyperText Markup Language (zkratka HTML) je v informatice název značkovacího jazyka používaného pro tvorbu webových stránek, které jsou propojeny hypertextovými odkazy. HTML je hlavním z jazyků pro vytváření stránek v systému **World Wide Web**, který umožňuje publikaci dokumentů na Internetu. Jazyk je aplikací dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka. [3]

2.2.2 CSS

Kaskádové styly (v anglickém originále Cascading Style Sheets se zkratkou **CSS**) jsou v informatice jazyk pro popis způsobu zobrazení elementů na stránkách napsaných v jazycích HTML, XHTML nebo XML. Hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. Starší verze HTML obsahují celou řadu elementů, které nepopisují obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí. [4]

2.2.3 JavaScript

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk. Používá se jako interpretovaný programovací jazyk pro WWW stránky, často vkládaný přímo do HTML kódu stránky. Jsou jím obvykle ovládány různé interaktivní prvky GUI (tlačítka, textová políčka) nebo tvořeny animace a efekty obrázků. [5]

Jeho syntaxe (zápis zdrojového textu) patří do rodiny jazyků C/C++/Java. Ale JavaScript je od těchto jazyků zásadně odlišný, sémanticky (vnitřně) jde o jiný jazyk. Standardizovaná verze JavaScriptu je pojmenována **ECMAScript** a z ní byly odvozeny i další implementace, jako je například ActionScript. [5]

Program v JavaScriptu se obvykle spouští až po stažení WWW stránky z Internetu (tzv. na straně klienta), na rozdíl od ostatních jiných interpretova-

ných programovacích jazyků (např. PHP a ASP), které se spouštějí na straně serveru ještě před stažením z Internetu. Z toho plynou jistá bezpečností omezení, JavaScript např. nemůže pracovat se soubory, aby tím neohrozil soukromí uživatele. JavaScript je možné použít i na straně serveru. [5]

- **jQuery** - je javascriptová knihovna obsahující mnoho vylepšení a usnadnění. Nejedná se tedy o nový jazyk.
- **AJAX** (Asynchronous JavaScript and XML) - je v informatice obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich kompletního znovunačítání za pomoci asynchronního zpracování webových stránek pomocí knihovny napsané v JavaScriptu. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů. [6]

2.2.4 C#

C# (vyslovované anglicky jako C Sharp) je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework. Microsoft založil C# na jazycích C++ a Java (a je tedy nepřímým potomkem jazyka C, ze kterého čerpá syntaxi). [7]

Standard ECMA definuje současný design C# takto: [7]

- C# je jednoduchý, moderní, mnohoúčelový a objektově orientovaný programovací jazyk.
- Jazyk a jeho implementace poskytuje podporu pro principy softwarového inženýrství, jakými jsou kupř. hlídání hranic polí, detekce použití neinicializovaných proměnných a automatický garbage collector. Důležité jsou také jeho vlastnosti jako robustnost, trvanlivost a programátorská produktivita.
- Jazyk je vhodný pro vývoj softwarových komponent distribuovaných v různých prostředích.
- Přenositelnost zdrojového kódu je velmi důležitá, obzvláště pro ty programátory, kteří jsou obeznámeni s C a C++.
- Mezinárodní podpora je též velmi důležitá.
- C# je navržen pro psaní aplikací jak pro zařízení se sofistikovanými operačními systémy, tak pro zařízení s omezenými možnostmi.
- Přestože by programy psané v C# neměly plýtvat s přiděleným procesorovým časem a pamětí, nemohou se měřit s aplikacemi psanými v C nebo jazyce symbolických adres.

C# lze využít k tvorbě databázových programů, webových aplikací a stránek, webových služeb, formulářových aplikací ve Windows, softwaru pro mobilní zařízení (PDA a mobilní telefony) atd. [7]

3 Programátorská dokumentace

Celý program (aplikaci) lze rozdělit do dvou částí. Jednou z nich je **MS SQL** databáze a druhou samotný program.

3.1 Databáze

V databázi jsou uložena pouze textová data, rozdělená do 5 samostatných tabulek:

- **users** - Obsahuje základní informace o uživatelském účtu (identifikátor, email, křestní jméno, příjmení, heslo)
- **user-data** - V této tabulce jsou uloženy dodatečné informace o uživateli (identifikátor, povolení ke zveřejnění údajů, telefonní číslo, datum narození, uživatelův identifikátor)
- **contacts** - Zde jsou ukládány uživatelské kontakty (identifikátor, uživatelův identifikátor, jméno, email, telefonní číslo, bydliště a datum narození)
- **category** - Tabulka pro ukládání kategorií pro akce (identifikátor, uživatelův identifikátor, název kategorie)
- **action** - Tabulka pro ukládání akcí (identifikátory - (vlastní, uživatelův, kategorie), název akce, popis akce, začátek akce a konec akce)

Column Name	Data Type	Allow Nulls
id	int	☐
user_id	int	☐
category_id	int	☑
name	nvarchar(150)	☐
text	nvarchar(MAX)	☑
from_date	datetime	☐
from_to	datetime	☐

Obrázek 1: Ukázka definované tabulky

Nepoužíval jsem žádné vazby mezi tabulkami, jelikož tyto „kolize“ řeším v programu.

3.2 Propojení s databází

K propojení programu a databáze používám framework určený přímo pro platformu .NET, a to **NHibernate**. Tento framework využívá mapovacích souborů (ve formátu XML) a má vlastní způsob zadávání dotazů na databázi. Místo klasických dotazů pro SQL databáze, používá zřetězení několika metod v jazyce C#.

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2" assembly="
       MVCDiary" namespace="MVCDiary.Models">
3
4    <class lazy="false" name="DiaryAction" table="action">
5      <id name="Id" column="id">
6        <generator class="native" />
7      </id>
8      <property name="UserId" column="user_id" />
9      <property name="CategoryId" column="category_id" />
10     <property name="Text" column="text" />
11     <property name="Name" column="name" />
12     <property name="From" column="from_date" />
13     <property name="To" column="from_to" />
14   </class>
15
16 </hibernate-mapping>
```

Zdrojový kód 1: Mapovací soubor **NHibernate** pro tabulku action

Základem **NHibernate** je soubor, ve kterém je nakonfigurován přístup k databázi, tj. k jaké databázi se připojuje, název databáze a přihlašovací údaje pro přístup k databázi. Další částí jsou již výše zmíněné mapovací soubory, které propojují Modely programu s jednotlivými tabulkami. Tyto soubory se nachází v programu.

3.3 Program

Struktura aplikace je rozdělena do dvou částí:

- Úvodní stránka
- Uživatelské prostředí

3.3.1 Úvodní stránka

Na úvodní stránce se řeší přihlášení, registrace a zapomenuté heslo. Přihlašování požaduje uživatelův email a heslo. Na této stránce se používají 2 kontrolery, z nichž jeden má na starost události ohledně přihlašování, zatímco ten druhý ohledně registrace.

- **Home** - První část tohoto kontroleru je přihlašování. K přihlašování používám upravenou třídu **MembershipProvider**, která obsahuje metody pro přihlášení.

```
1  public ActionResult SignIn(string email, string password)
2        {
3           if (Membership.ValidateUser(email, password))
4           {
5              FormsAuthentication.SetAuthCookie(email, false);
6              return RedirectToAction("Index", "Home", new { area = "
                   User" });
7           }
8           TempData["message-error"] = "Špatně zadaný email nebo heslo
                ";
9           return RedirectToAction("Index");
10       }
```

Zdrojový kód 2: Přihlašování s využitím třídy Membership

V tomto kontroleru se také vyskytuje generátor nového hesla, jelikož si o něj uživatel může zažádat, pokud zapomene heslo. Metoda vygeneruje nové pětimístné heslo, uloží ho do databáze a emailem ho pošle uživateli.

Samozřejmě nechybí i přesměrování přihlášeného uživatele, tj. když se na tuto stránku dostane uživatel, který je již přihlášený, okamžitě ho to přesměruje do Uživatelského prostředí.

- **Registration** - Tento kontroler zajišťuje registraci nového uživatele. Obsahuje 2 funkce. První je samotná registrace, ve které se pomocí hašovací funkce zakóduje heslo, vytvoří se nový uživatel a následně pro něj vytvoří záznam v tabulce user-data, kde se budou ukládat jeho osobní informace.

  Druhou funkcí je funkce na ověření, že zadaný email již někdo používá. Samotné ověření vstupů má na starost třída **DataAnnotations**. Tato třída nastaví parametry modelu, kde jednotlivé vlastnosti modelu (Email, SurName, password,...) odpovídají vstupním polům. DataAnnotations zahrnují i již zmíněnou metodu na ověření emailu.

```
1   [Required(ErrorMessage = "Je vyžadován email")]
2   [EmailAddress(ErrorMessage = "Zadejte emailovou adresu")]
3   [Remote("UniqueEmailAddress", "Registration", HttpMethod = "POST",
        ErrorMessage = "Účet s tímto emailem již existuje")]
4   public override string Email { get; set; }
```

Zdrojový kód 3: Ukázka využití DataAnnotations

Když jsou nastaveny parametry (podmínky) vstupů, zbývá je pouze aplikovat na vstupní řetězce. K tomuto používám silné typování proměnných.

To znamená, že v pohledu (v zobrazené HTML stránce) nastavím parametr **model** na vybranou třídu, čímž propojím model s pohledem. (např: @model MVCDiary.Models.DiaryUser)

```
1 @Html.TextBoxFor(x => x.Email, new { @autocomplete = "off",
     @placeholder = "Email", @type = "email", @maxlength = "50" })
2 @Html.ValidationMessageFor(x => x.Email, null, new { @class = "mes"
     })
```

Zdrojový kód 4: Ukázka využití silně typované proměnné

V hlavní struktuře programu mám soubory, jako jsou modely, objekty, mapovací soubory, skripty, css styly a samozřejmě pohledy.

- **Modely** - Jsou to třídy, které definují struktury. (u uživatele určují, že má vlastnost jako Email, Křestní jméno,...). S pomocí NHibernate jsou propojené přímo s tabulkami z databáze.
- **Objekty** - V objektech jsou nastaveny metody dotazovaní k databázi. Jak jsem již psal dříve, NHibernate má vlastní sadu na dotazování se k databázi. Všechny dotazy jsou uloženy v těchto třídách, kde jedna třída odpovídá jedné tabulce z databáze.

```
1        public IList<DiaryAction> GetActionsByUserId(int userId)
2        {
3           return session.CreateCriteria<DiaryAction>()
4              .Add(Restrictions.Eq("UserId", userId))
5              .List<DiaryAction>();
6        }
```

Zdrojový kód 5: Dotazování pomocí NHibernate

Výše uvedený zdrojový kód vrací seznam akcí, podle uživatelova identifikátoru.

- **Mapovací soubory** - Jak již bylo zmíněno, jedná se soubory, které propojují Modely s jednotlivými tabulkami.
- **Skripty** - Tato složka obsahuje všechny javascriptové soubory, zejména jQuery knihovny. Většina těchto souborů jsou speciálně vytvořená pro ASP.NET, kde ulehčují práci metody AJAX. Všechny tyto knihovny jsou volně stáhnutelné a ulehčují práci s JavaScriptem. V této složce je jeden můj vlastní soubor, ve kterém mám vytvořené vlastní funkce.
- **CSS styly** - Složka obsahuje pouze dva styly, kde jeden je pro úvodní stránku a druhý pro uživatelské prostředí.

- **Pohledy** - Složka, ve které jsou uloženy zobrazované stránky pro úvodní stránku. Vytvořil jsem šablonu, do které se generují různé pohledy jako pohled pro přihlašování, registraci a zapomenuté heslo.

3.3.2 Uživatelské prostředí

V ASP.NET MVC se používají tzv. oblasti, které rozdělují strukturu na různé části. Tyto části mají vlastní kontrolery, pohledy i modely. Využil jsem tyto oblasti na oddělení úvodní stránky od uživatelského prostředí.

Všechny části uživatelského prostředí ovládá jeden kontroler, který má několik akcí. Tato oblast (uživatelské protředí) používá svoji vlastní šablonu, takže se liší od úvodní stránky.

Šablona obsahuje tyto neměnné části:

- Obsahuje levou navigaci, která slouží pro přepínání úvodní stránky, stránky kalendáře a stránky kontaktů. Ikony těchto odkazů jsou vytvořeny pomocí jednoduché SVG grafiky.

```
<a href="/User" class="tooltip">
  <span class="tooltiptext">Domů</span>
        <svg height="30" width="30">
           <circle cx="50%" cy="50%" r="15" />
          <text x="9" y="22" font-size="18px">D</text>
     </svg>
</a>
```

Zdrojový kód 6: Odkazy pomocí SVG grafiky

Vy výše uvedém kódu lze vidět, že jsem vytvořil i třídu **tooltip**, která slouží k tomu, že po najetí na odkaz se zobrazí jeho popisek.

- Další částí této šablony je postranní menu (vedle výše uvedené navigace), ve kterém je rozdělení jednotlivých funkcí aplikace. Tj. rozdělení kalendáře, kontaktů, nastavení i úvodní stránky uživatelského prostředí.

- V šabloně se vyskytuje i horní menu, ve kterém lze nalézt Odhlášení, Nastavení, Vyhledávání a jméno přihlášeného uživatele. Po stistknutí tlačítka Odhlášení, se zavolá následující funkce:

```
            FormsAuthentication.SignOut();
             Session.Clear();
             return RedirectToAction("Index", "Home");
```

Zdrojový kód 7: Odhlášení

Výše uvedný kód zajistí, že se odstraní informace o přihlášeném uživateli a přesměrují ho zpět na úvodní stránku, tedy na stránku přihlášení. Další částí menu je tlačítko nastavení, které má pouze za úkol otevřít sekci Nastavení.

Dále se tu nachází i vstupní pole s tlačítkem Vyhledávání. Toto vyhledávání slouží k hledání uživatelových akcí (plánů). Vyhledávání je řešeno pomocí AJAXu. Pokud je pole prázdné, vyhledávání neproběhne. Pokud ale uživatel něco zadá, zavolá se metoda **SearchResult**, která prohledá databázi a vrátí uživatelovy hledané akce. Výsledky se ukládají do „modal okna“. Toto „modal okno“ je součástí šablony. Z počátku není viditelné, ale jakmile metoda **SearchResult** skončí, výsledek uloží do tohoto „okna“ a JavaScript ho zviditelní.

```
1  $(document).on("click", ".sclick", function () {
2        if ($('#s-box').val().length != 0) {
3           $.ajax({
4              url: "/User/SearchResult",
5              type: "POST",
6              data: { term: $('#s-box').val() },
7              success: function (data) {
8                 $('#searchId').html(data);
9                 $('#searchId').siblings(".top").children("h4").text("
                     Výsledky hledání pro: \"" + $('#s-box').val() + "
                     \"")
10                $('.modal-search').css("display", "block");
11             }
12          });
13       }
14    });
```

Zdrojový kód 8: Vyhledávání pomocí AJAXu

- Poslední „neměnnou“ částí šablony jsou tzv. „modal okna“. Již jsem se o nich zmýnil u vyhledávání. Z počátku jsou to neviditelné bloky, které slouží k zobrazení akcí. Aplikace obsahuje několik funkcí, které vrací např. seznam nějakých akcí. Výsledná data se uloží do těchto „oken“ a ihned poté se okna zobrazí. Vše probíhá pomocí AJAXu, takže nedochází ke zbytečnému znovunačtení aktuální stránky.

Samotná šalbona má i několik dalších prvků. Obsahuje i odkazy pro generování nového obsahu.

```
1     @RenderSection("User", true)
```

Zdrojový kód 9: Příklad vkládání nového obsahu do šablony

Tento kód vygeneruje do šablony sekci „User“. Parametr „true“ znamená, že tato sekce je povinná a musí být definována. Těchto „generátorů“ je v šabloně více. Rozlišují se způsoby použití. Podle způsobu použití se rozeznávají generátory pro obsah, jméno přihlášeného uživatele, navigace, atd...

- **Home** - Jak jsem již zmínil, uživatelské prostředí obsahuje pouze jeden kontroler. Tento kontroler má více funkcí. Jednou z nich je zobrazování stránek. Jsou to vlastně metody, které jako výsledek vrací pohled. To znamená, že po skončení této metody se zobrazí příslušná stránka. Kontroler celkově zobrazuje 4 stránky. Je to stránka Nastavení, Kontakty, Kalendář a úvodní stránka uživatelského prostředí. Do každé z těchto stránek se načítají další podstránky, které pouze upravují její obsah. Další částí kontroleru jsou různé funkce a metody pro mazání, editaci a vytváření akcí nebo kontaktů. Zároveň si tento kontroler uchovává spoustu informací, jako jsou uživatelovy akce, kontakty, kategorie, jeho vlastní údaje, návratovou stránku a spoustu dalších věcí.

- **Odkazování v uživatelském prostředí** - Na této stránce jsou dva druhy odkazování. Jedním z nich je odkazování na Nastavení, Kontakty, Kalendář a úvodní stránku uživatelského prostředí. Tohle jsou jediné „zobrazovací odkazy“, které znovu načítají stránku. Jelikož má hlavní stránka uživatelského prostředí adresu „/User“, po načtení těchto stránek se mění adresa např: na „/User/Calendar“ atd... Jedná se tedy o klasické odkazování jako na běžných webových stránkách.

  Druhý typ odkazování je pomocí AJAXu. AJAX pouze nahradí část stránky, takže není nutné její znovunačtení. Největší výhodu AJAXu je to, že se nemusí znovu načítat celá stránka, což zmenšuje velikost načítaných dat a zároveň urychluje běh celé aplikace. AJAX využívám uvnitř jednotlivých stránek (Kalendáře, Kontaktů,...)

Nyní se dostáváme k popisu hlavních částí celé aplikace. Celé uživatelské prostředí se dá rozdělit na 4 části. A to na Nastavení, Kontakty, Kalendář a Úvodní stránku.

1. **Nastavení** - Nastavení obsahuje tři části. Změnu osobních údajů, změnu hesla a smazání účtu. U změny nastavení osobních údajů jde u obyčejný formulář, který po vyplnění zavolá akci **SaveChanges**.

```
1        [HttpPost]
2        public ActionResult SaveChanges(DiaryUserData diaryUserData,
             string firstName, string surName);
```

Zdrojový kód 10: Úprava osobních údajů

Opět v pohledu používám silně typované proměnné, takže pohled vrací celou třídu, obsahující všechny data, které se budou ukládat do tabulky user-data. Jelikož může uživatel i měnit své jméno a příjmení, tak metoda dostává na vstupu i tyto údaje, které se uloží do jiné tabulky, konkrétně do tabulky users. Na začátku kódu používám „metodu“ [**HttpPost**], která zajišťuje, že tato metoda půjde zavolat jen pomocí formuláře. Takhle se zabrání nežádoucím zavolání této metody. Jakmile metoda upraví hodnoty v databázích, vyplní TempData (zpráva o výsledku) a přesměruje se zpět na stránku Nastavení.

Další funkcí je změna hesla uživatele. Na vstupu je opět formulář, který očekává původní heslo a dvakrát nové heslo.

```
1       public ActionResult ChangePassword(string password, string
            newPassword1, string newPassword2)
```

Zdrojový kód 11: Změna hesla

Poslední částí nastavení, je možnost smazat svůj účet. Jedná se pouze o tlačítko, které po stisknutí zavolá metodu na smazání profilu. Ještě předtím, než se ale tato metoda zavolá, tak jsem pomocí JavaScriptu zajistil, že po kliknutí vyskočí otázka na uživatele, zda opravdu chce smazat svůj účet. Tohle by mělo zabránit nechtěnému kliknutí.

```
1   @onsubmit = "return confirm(\"Opravdu chcete smazat svůj účet?\")"
```

Zdrojový kód 12: Potvrzení smazání

Pokud to uživatel potvrdí, pak metoda smaže uživatele z databáze a s ním i jeho akce a kontakty.

2. **Kontakty** - Stránka pro správu kontaktů. Oproti nastavení obsahuje tato stránka i nástrojovu lištu s funkcemi jako Přidání nového kontaktu, upravení kontaktu a smazání kontaktu. Tato stránka obsahuje vlastní „modal okna“ pro vytváření a editaci kontaktů.

   - **Nový kontakt** - Po kliknutí na tlačítko Přidat nový kontakt, se objeví „modal okno“ s formulářem pro vyplnění údajů nového kontaktu. Všechny vstupy (jako v celé aplikaci) jsou oštřené pomocí HTML podmínek, jako je maxlength, a další. Kontroler tedy nemá moc práce s kontrolou těchto vstupů. Po kliknutí na odeslání formuláře se zavolá funkce z kontroleru **AddContact**.

     Tato funkce přijímá jméno, email, bydliště, telefonní číslo a datum narození. Jelikož údaje byly ověřeny již v pohledu pomocí HTML

```
1   public ActionResult AddContact(string name, string email, string
       place, string phone, DateTime? date);
```

Zdrojový kód 13: Přidání nového kontaktu

podmínek, tak tato funkce pouze uloží hodnoty do databáze a následně přesměruje uživatele zpět na stránku kontaktů.

- **Úprava kontaktů** - Tato funkce funguje podobně jako vytváření kontaktů. Opět se otevírá ve vlastním modal okně, který obsahuje formulář pro kontakt. Jedná se AJAXový odkaz, který jako parametr obsahuje identifikátor upravovaného kontaktu. Tento parametr obdrží funkce v kontroleru, která podle tohoto identifikátoru najde v databázi odpovídající kontakt. Tento kontakt pošle pohledu (speciální stránka na editaci), která do formuláře vyplní již zadané informace o kontaktu. Jakmile uživatel doplní do formuláře potřebné věci, tak odešle formulář. Teprve v tuhle chvíli se zavolá funkce **EditContact**, která aktualizuje údaje o kontaktu v databázi a pak následně přesměruje uživatele zpět na stránku Kontakty.

```
1   @model MVCDiary.Models.DiaryContact
2   @{
3       Layout = null;
4       MVCDiary.Models.DiaryContact contact = ViewBag.Contact;
5   }
6   @using (Html.BeginForm("EditContact", "Home", FormMethod.Post, new
        { @class = "changes" }))
7   {
8       <label for="names">Jméno</label>
9       @Html.TextBoxFor(x => x.Name, new { @Value = contact.Name,
            @type = "text", @id = "names", @autocomplete = "off",
            @maxlength = "50", @required = "required" })
10
11      <label for="em">Email</label>
12      @Html.TextBoxFor(x => x.Email, new { @Value = contact.Email,
            @type = "email", @id = "em", @autocomplete = "off",
            @maxlength = "35", @required = "required" })
13
14      ...
15
16      <button type="submit">Uložit kontakt</button>
17  }
```

Zdrojový kód 14: Stránka pro editaci kontaktu

Jak lze vidět, opět je použita silně typovaná proměnná, která uchovává údaje o kontaktu. Tato stránka přijme upravovaný kontakt a ve for-

muláři vyplní jednotlivá pole. Teprve po kliknutí na „Uložit kontakt“ se zavolá metoda „EditContact“, která upraví kontakt v databázi.

- **Mazání kontaktů** - Jedná se pouze o odkaz s nastavenými parametry. Tento odkaz si uchovává identifikátor kontaktu a po stisknutí tohoto odkazu se zavolá metoda **DeleteContact**, která vybere kontakt podle identifikátoru z databáze a následně ho smaže.
- **Zobrazení kontaktů** - Kontakty se zobrazují v samotném obsahu stránky. Každý z kontaktů má samostatný blok, ve kterém se vypisují údaje jako jméno, email, atd. Samotný pohled získává z kontroleru seznam kontaktů, které má zobrazit. Poté pomocí cyklu projede každý prvek z pole a vypíše ho jako kontakt. I když pohled je ve výsledku HTML kód, Razor v něm umožňuje používat i C#.

  Kvůli jednoduchosti jsem přidal ke kontaktům i odkazy na mazání kontaktů. Po najetí na kontakt se v pravém horním rohu zobrazí ikona pro smazání. Po kliknutí na tento odkaz se zavolá jiná funkce na smazání kontaktu, která obsahuje jeho identifikátor. Podle následujícího identifikátoru funkce smaže vybraný kontakt.

```
1 DiaryContact da = diaryContactObject.GetContactById(int.Parse(id));
2 diaryContactObject.Delete(da);
3 TempData["message-success"] = "Vybraný kontakt byl úspěšně smazán";
4 return RedirectToAction("Contacts", "Home");
```

Zdrojový kód 15: Smazání kontaktu

- **Výběr kontaktů** - Samozřejmou funkcí je vícenásobný výběr kontaktů a manipulace s nimi. Tuhle funkci jsem vytvořil pomocí JavaScriptu. Nejdříve bylo zapotřebí si vytvořit pole, nebo podobnou strukturu, do které se budou ukládat identifikátory kontaktů.

```
1   var selectedContacts = new Set();
```

Zdrojový kód 16: Pole pro ukládání identifikátorů

Dále po kliknutí na jednotlivý kontakt se zavolá JavaScriptová funkce, která manipuje s identifikátory kontaktu. Nejdříve JavaScript získá identifikátor vybraného kontaktu, pak mu přenastaví třídu na **select**. Tato třída má v CSS pouze jinou barvu pozadí, aby bylo vidět, že je vybraný kontakt skutečně vybraný a že se tedy liší od ostatních. Poté se do již vytvořeného pole přidá, případně odebere identifikátor kontaktu. Jako poslední část se zavolají metody **showEdit** a **showDelete**. Tyto metody mají na starost úpravu odkazů v nástrojové liště.

Jde o to, že pokud není vybraný žádný kontakt, nejsou odkazy (tlačítka) pro úpravu a mazání kontaktů dostupné. Tyto metody nastaví, jestli se vybrané odkazy zobrazí, nebo ne. Zároveň přenastavují jejich parametry na požadované identifikátory. Tlačítko Odebrat kontakt je dostupé tehdy, když je označen jeden nebo více kontaktů, zatímco tlačítko Upravit kontakt je dostupné pouze tehdy, pokud je označený pouze jeden kontakt.

```
1   var id = $(this).children('.idecko').val();
2           if ($(this).hasClass("select")) {
3              $(this).removeClass("select");
4              selectedContacts.delete(id);
5              showEdit();
6              showDelete();
7           }
8           else {
9              $(this).addClass("select");
10             selectedContacts.add(id);
11             showEdit();
12             showDelete();
13          }
```

Zdrojový kód 17: Pole pro ukládání identifikátorů

- Další část stránky kontaktů je levé menu. V něm jsou odkazy pro zlepšení hledání mezi kontakty. Obsahuje odkazy: „Od nejnovějších“, „Od nejstarších“, „Dle jména vzestupně“ a „Dle jména sestupně“. Již podle názvu je jasné, že tyto odkazy mají pouze za úkol seřadit kontakty na stránce. Jsou to AJAXové odkazy (jak jsem již zmínil), které nahrazují pouze samotný obsah stránky. Zůstává i nástrojová lišta. Všechny tyto odkazy fungují stejně. Po kliknutí zavolají jim příslušnou funkci v kontroleru. Liší se ovšem pohledem, který tyto akce vrací. Před výpisem se pomocí C# seřadí do požadované pozice.

```
1   contacts = contacts.OrderBy(x => x.Name).ToList();
```

Zdrojový kód 18: Seřazení kontaktů

- **Posílání emailů** - Tohle je poslední funkce pro kontakty. Emaily opět obsahují vlastní modal okno. Email, vypsaný na stránce v kontaktu je AJAXový odkaz, který se odkazuje na akci v kontroleru. Tato akce vrací speciální stránku, která se díky AJAXu vygeneruje do příslušného modal okna. V modal okně bude tedy formulář, se dvěma vstupy. Jeden vstup reprezentující předmět emailu a druhý vstup text emailu.

Po kliknutí se zavolá metoda **SendMailContact**. Tato metoda doplní text, který se bude posílat o kontaktní informace odesílatele. Poté vytvoří s využitím třídy **Mailer** parametry zprávy. Následně odešle email a vrátí se zpět na stránku kontakty.

Třída **Mailer** využívá SmtpClienta. U tohoto klienta je nastavená emailová adresa, ze které se bude zpráva posílat, heslo k přístupu na tento email a nastavení odeslání. Metoda **SendMailContact** tedy vyplní potřebné informace (text, předmět,...) a zavolá funkci třídy **Mailer** „Send“, která odešle email.

```
1  public void Send()
2        {
3          SmtpClient smtpClient = new SmtpClient();
4            smtpClient.Host = MailHost;
5            smtpClient.Port = MailPort;
6            smtpClient.EnableSsl = MailSSL;
7            smtpClient.DeliveryMethod = SmtpDeliveryMethod.Network;
8            smtpClient.UseDefaultCredentials = false;
9            smtpClient.Credentials = new NetworkCredential(FromEmail,
                 FromPassword);
10
11           using (var message = new MailMessage(FromEmail, ToEmail))
12           {
13                message.Subject = Subject;
14               message.Body = Text;
15                message.IsBodyHtml = IsHtml;
16                smtpClient.Send(message);
17           }
18       }
```

Zdrojový kód 19: Posílání emailu

3. **Kalendář** - Hlavní funkce celé aplikace. Tato stránka slouží pro vytváření, úpravu či mazání akcí (plánů).

   - **Přidávání, úprava a mazání akcí** - Všechny tyto funkce se téměř neliší od funkcí pro správu kontaktů. Ovšem volají jiné metody, případně používají vlastní modal okna.
   - **Zobrazení akcí** - Oproti kontaktům se akce nezobrazují podle jména nebo stáří. Akce se zobrazují podle toho, kdy probíhají. V levém menu jsou odkazy „Kalendář“, „Probíhající akce“, „Nadcházející akce“ a „Již skončené akce“. Poslední tři odkazy volají funkce kontroleru. Všechny tyto funkce posílají zpět pohledu seznam akcí. V pohledu (za využití Razoru a tedy i C#) program získá aktuální datum a čas. Nyní nastane jednoduchá podmínka, která ověří, zda jsou nějaké probíhající, nadcházející nebo již uplynulé akce. Tohle je jediná věc, ve které se

tyto akce liší. Dále pohledy vypíší následující akce do obsahu. Po najetí na vypsanou akci se kromě smazání akce zobrazí i tlačítko na úpravu akce. Opět se jedná o AJAXový odkaz, který funguje stejně jako úprava akce v nástrojové liště. Má vlastní modal okno, do kterého se generuje výsledek akce odkazu. V modal okně vygeneruje formulář, který po vyplnění zavolá metodu **EditAction**, která vybranou akci upraví a uloží do databáze.

Dalším rozdílem oproti kontaktům je kategorie. Některé akce můžou patřit do kategorie, kterou jim uživatel přiřadil. Po kliknutí na tuto kategorii vyskočí modal okno, ve kterém budou zobrazeny akce patřící do této kategorie. Opět se jedná o stejný princip jako v případě editace. Kategorie mají vlastní modal okno. Po kliknutí na odkaz se zavolá funkce z kontroleru, která vygeneruje do modal okna všechny akce patřící do této kategorie. Tentokrát jediným tlačítkem zůstává tlačítko na zavření modal okna, jelikož neobsahuje žádný formulář (podobně jako u vyhledávání).

- **Výběr akcí** - Výběr jsem popsal již u kontaktů. Výběr akcí a kontaktů se příliš neliší. Rozdíl je v tom, že výběr akcí používá vlastní JavaScriptové pole, aby nedocházelo ke kolizím.
- **Kalendář** - Samotná reprezentace kalendáře. K tomuto využívám nástroj patřící do jQuery UI, **DatePicker**. Je to simulace kalendáře, která se hojně využívá na různých webových stránkách. Jako parametr pohled dostává seznam akcí uživatele.

```
1       <div id="calendar"></div>
```

Zdrojový kód 20: Vložení kalendáře

Jak jde vidět, kalendář nemá s HTML téměř nic společného. Vše se nastavuje přímo v jQuery.

Jelikož **DatePicker** zobrazuje měsíční kalendář, tak v Razoru pokaždé vyfiltruji ty akce, které nepatří do aktuálního měsíce. Nejprve si vytvořím nové pole akcí, do kterého si uložím všechna data patřící do aktuálního měsíce včetně těch, které se nachází v rozsahu více dní. Tímto způsobem dostaneme 2 pole akcí, které stačí sloučit a tím získáme požadované akce, které patří do aktuálního měsíce.

Nyní se můžeme podívat na nastavení celého kalendáře. Nejprve bylo zapotřebí „počeštit“ celý kalendář a nastavit mu formát datumu. Ve funkci **beforeShowDay** se nastavily data kalendáře, ve kterých probíhá nějaká akce. Tyto akce jsem získal metodou **JSON**. Jinými slovy jsem získal data z Razoru a uložil je do JavaScriptového pole.

V této chvíli jsou v kalendáři vyznačené datumy, ve kterých probíhá nějaká akce. Zbývá tedy tyto akce zobrazit. Opět jsem využil dalšího

```
1   DateTime[] times = actions.Select(x => x.From).ToList().Distinct()
       .ToArray().Union(newList.Distinct()).ToArray();
2     string[] dates = times.Select(x => x.ToString("yyyy-MM-dd")).
         ToArray();
```

Zdrojový kód 21: Vyfiltrování akcí

modal okna. V nastavení kalendáře v metodě **onSelect**, která reaguje na kliknutí na příslušné datum, jsem nastavil viditelnost modal okna a zavolal metodu AJAX, která zavolala další funkci z kontroleru. Tato funkce vyplnila modal okno akcemi na vybraný den.

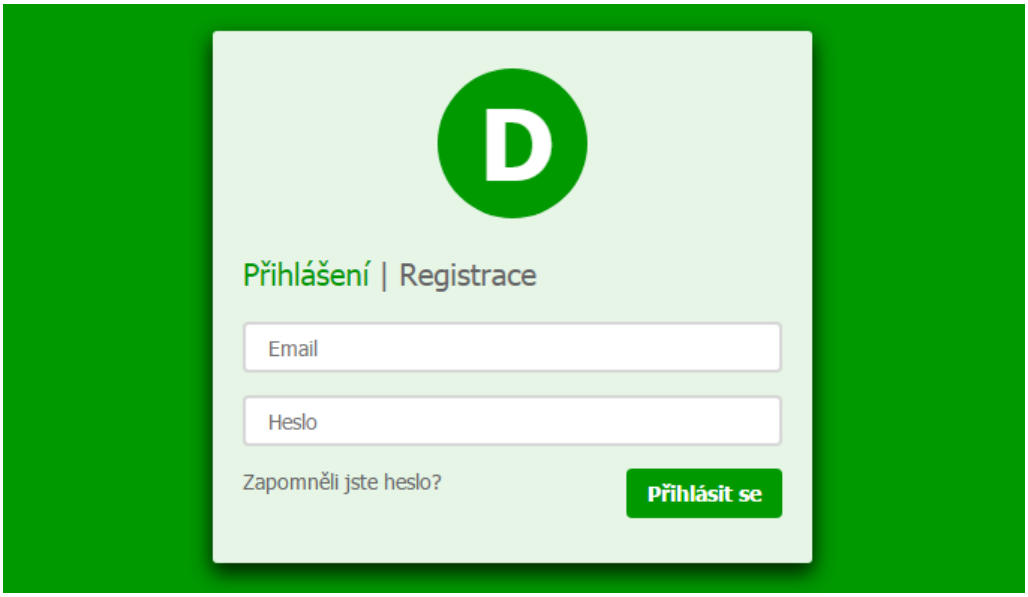
Poslední věc, která zbývá ke splnění požadavků je upozornění na kolizi akcí. Bylo zapotřebí vhodně zvolit tato upozornění. Zvolil jsem tedy nový blok, na stránce u kalendáře. Upozornění tedy bude vyhodnocovat vždy jen jeden měsíc, podle toho, jaký je zrovna vybrán v kalendáři. Bylo tedy potřeba zjistit, jaký je aktuální měsíc v kalendáři. To se dá zjistit zachycením kliknutí na přepínání měsíců v kalendáři (viz. kód výše). Jakmile jQuery zachytí změnu měsíce, zvýší (sníží) datum dle kliknutí. Proměnná datum je ovšem moje vlastní proměnná vytvořená při znovu načtení stránky.

Poté se použije metoda AJAX, která zavolá funkci z kontroleru **SetWarnings**. Tato funkce má jediný cíl, a to zjistit, u kterých akcí dochází k časové kolizi ve vybraném měsíci. Výsledek ukládá do bloku pro upozornění kolizí.

4. **Hlavní stránka uživatelského prostředí** - Tato stránka neobsahuje nic nového, než je uvedeno v předchozích stránkách. Hlavní stránka informuje o nádcházejících akcích i posledně přidaných kontaktech. Ovšem obsahuje i odkazy na smazání a úpravu akcí a odkaz na smazání kontaktu. V tomhle případě se volá funkce z kontroleru, která smaže kontakt (nebo akci) a odkáže uživatele zpět na hlavní stránku.

4 Uživatelská příručka

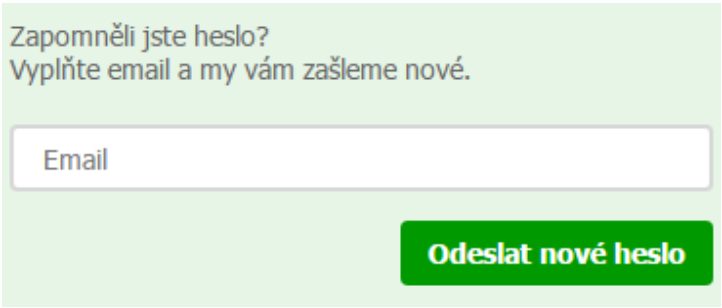
4.1 Úvodní stránka



Obrázek 2: Přihlašovací stránka

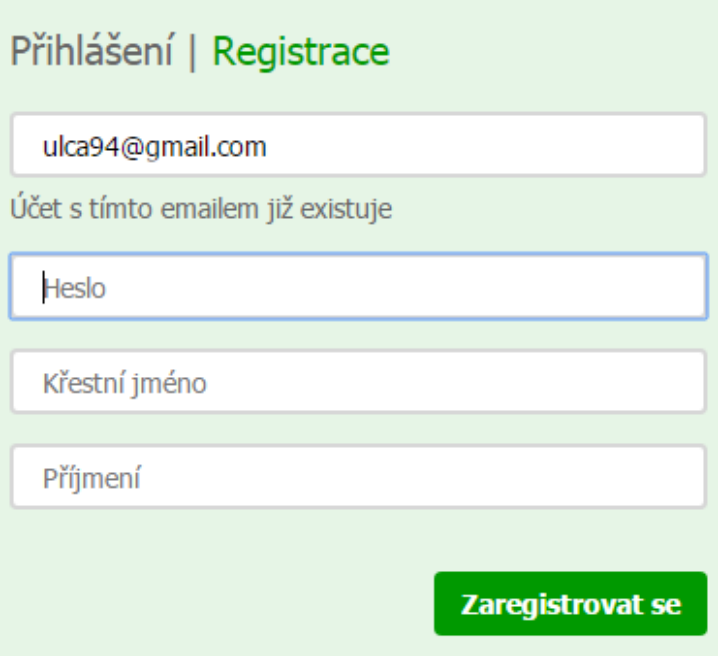
Úvodní stránka celé aplikace funguje jako přihlašovací stránka. Pokud se uživatel pokusí, např. přes webovou adresu, dostat dovnitř aplikace, aplikace zjistí, jestli je uživatel přihlášen. Pokud není, aplikace ho přesměruje zpět na úvodní stránku. Na této stránce se nachází formulář pro vyplnění přihlašovacích údajů. 2

Po úspěšném přihlášení bude uživatel přesměrován do uživatelského prostředí. Pokud zadá špatné údaje, objeví se zpráva, že uživatel zadal špatný email nebo heslo.



Obrázek 3: Zapomenuté heslo

Pokud uživatel zapomene své heslo, má možnost zažádat o nové. Po kliknutí na **Zapomněli jste heslo** se objeví formulář pro vyplnění emailu. 3 Na tento email bude uživateli zasláno nové heslo, které má neomezenou dobu platnosti.

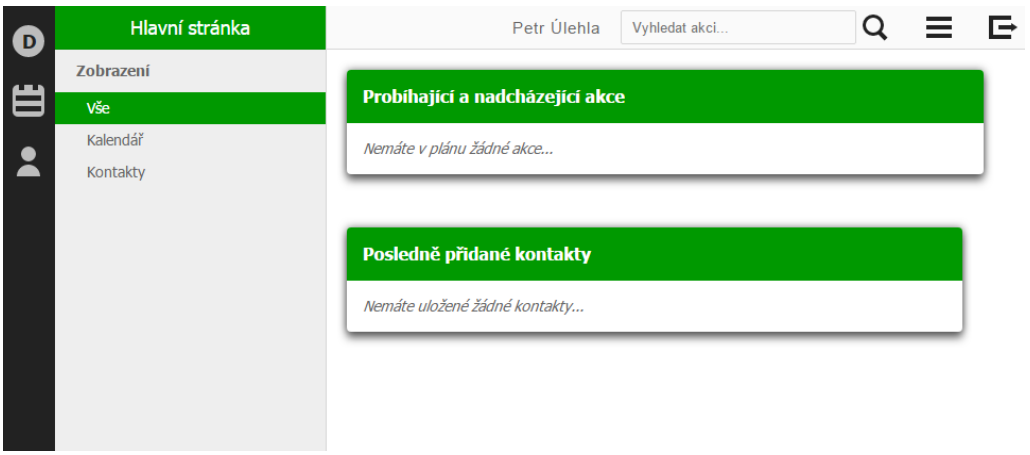


Obrázek 4: Registrace nového uživatele

Po kliknutí na tlačítko **Registrace** se objeví formulář pro registraci. Obsahuje 4 povinné vstupní pole. Je to pole pro zadání emailu, hesla, jména a příjmení. 4 Okamžitě po vyplnění hodnot se uživateli zobrazí, jestli zadanou hodnotu zadal správně, případně jak má taková hodnota vypadat (délka řetězce atd...)

Pos úspěšném přihlášení bude uživatel přesměrován na uživatelského prostředí.

4.2 Uživatelské prostředí



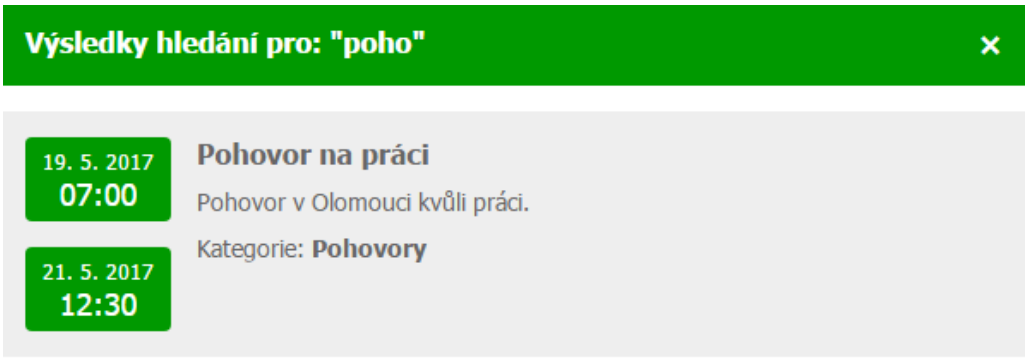
Obrázek 5: Úvodní stránka uživatelského prostředí)

Uživatelské prostředí má vlastní vzhled s různými ovládacími prvky. Zároveň se toto prosředí dá rozdělit do 4 samostaných stránek, který tento vzhled využívají. 5 Jsou to stránky **Hlavní stránka**, **Kalendář**, **Kontakty** a **Nastavení**.

Každá stránka má vlastní navigační menu a obsah. Na stránce jsou tedy 2 části neměnící svůj obsah a 2 části, které se mění podle toho, kam se uživatel chce odkázat.

Neměnné části vzhledu uživatelského prostředí:

- **Levé navigační menu (tmavé)** - Toto menu slouží k odkazování mezi Hlavní stránkou uživatelského prostředí, kalendářem a kontakty. Po najetí změní odkaz barvu a ukáže se nápověda, kam odkaz odkazuje. Po kliknutí se změní sousední šedé menu a obsah stránky.
- **Horní menu** - V tomto menu je uvedeno jméno přihlášeného uživatele, pole pro vyhledávání plánů (akcí) uživatele, tlačítko nastavení a odhlášení. Tlačítko nastavení funguje stejně jako tlačítka v levém navigačním menu. To znamená, že po kliknutí se změní šedé menu a obsah stránky. Po kliknutí na tlačítko Odhlášení bude uživatel odhlášen a přesměrován na přihlašovací stránka celé aplikace.

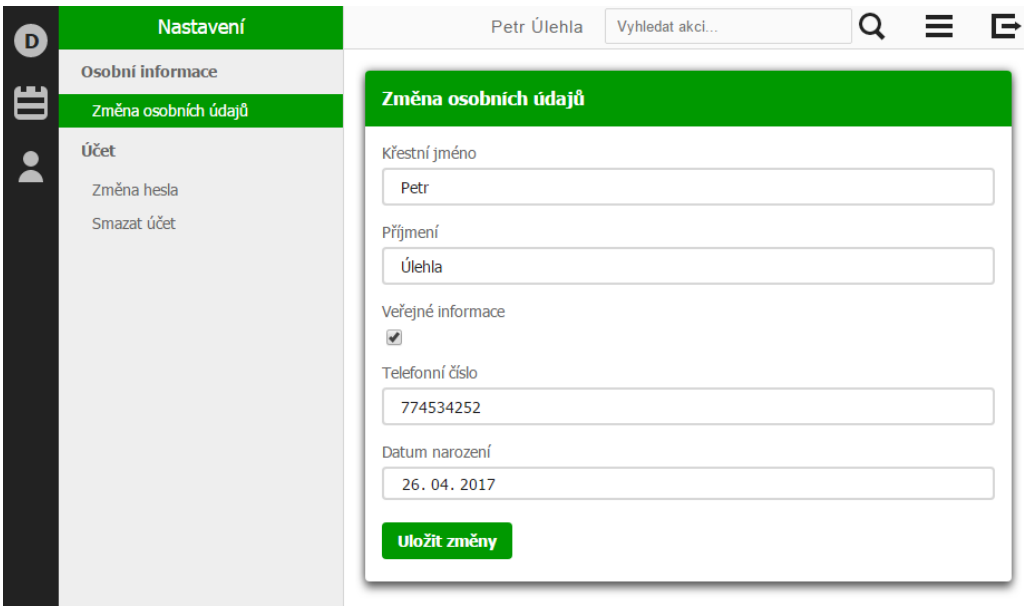


Obrázek 6: Ukázka vyhledávání plánů (akcí)

Toto menu obsahuje i vstupní pole pro vyhledávání akcí. Po stisknutí tlačítka vyhledávání se zobrazí vyskakovací okno 6 ve kterém budou vypsány plány (akce) odpovídající hledanému výrazu. Pokud ovšem neexistuje odpovídající akce, do vyskakovacího okna se napíše, že žádná akce nebyla nalezana.

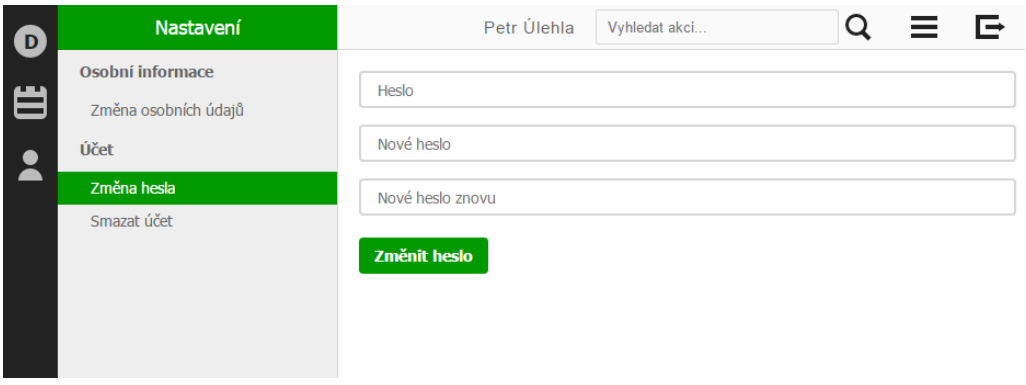
Jak jsem se již zmínil, toto uživatelské prostředí se dá rozdělit na čtyři části podle toho, co tyto části zobrazují. Po zobrazení těchto částí se nahradí obsah stránky a šedé menu. Šedé menu pak slouží k orientaci na vybrané stránce (k orientaci u kalendáře, nastavení, kontaktů). Po stisknutí odkazů v tomto menu se mění pouze obsah stránky.

1. **Nastavení**



Obrázek 7: Úprava osobních údajů

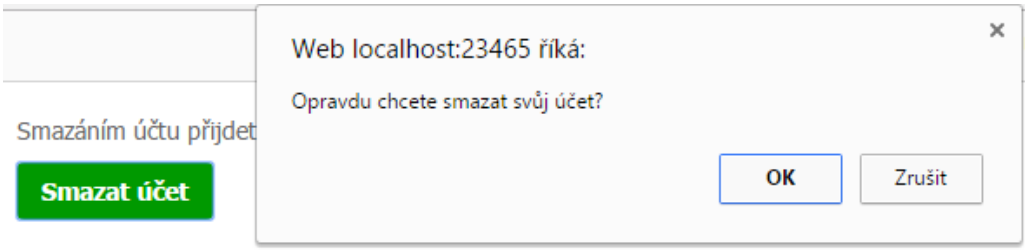
Po stisknutí tlačítka Nastavení se zobrazí následující stránka. 7 Na této stránce je již zobrazený formulář pro změnu osobních údajů. Uživatel si může tyto údaje vyplnit dle libosti. Povinnými údaji jsou pouze Křestní jméno a Příjmení. Dále si uživatel může zaškrtnout možnost „Veřejných informací“. Tato možnost slouží pro posílání emailů jednotlivým kontaktům. Pokud je toto pole zaškrtnuté, email se odešle s údaji uživatele, tj. s Telefonním čísle, věkem, jménem a emailem (pokud jsou tyto údaje vyplněné). Jinak bude odesílat pouze jméno a email uživatele.



Obrázek 8: Nové heslo

Po kliknutí v šedém menu na **Změnu hesla** se změní obsah stránky na jiný. Stávající formulář se tedy nahradí novým formulářem pro změnu hesla.

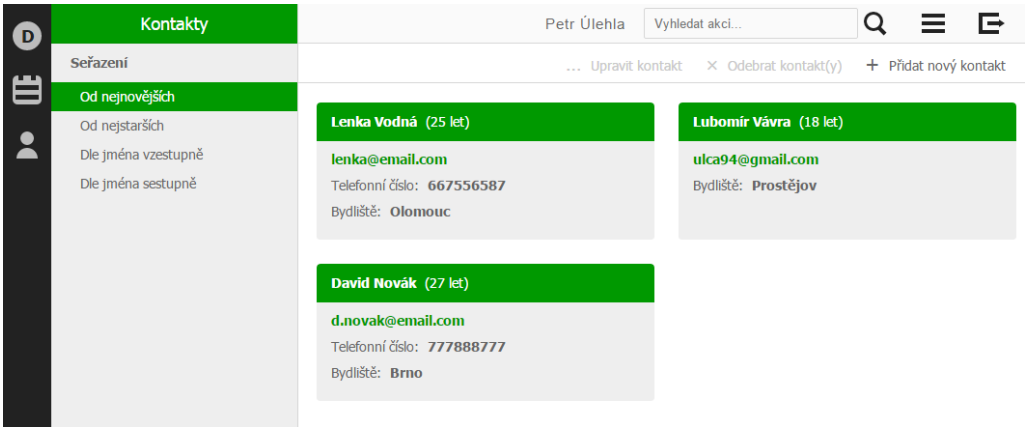
Uživatel se zadá dvoje původní heslo a dvakrát nové heslo. Pokud vše vyplní správně, tak po kliknutí se objeví zpráva, že heslo bylo úspěšně změněno.



Obrázek 9: Smazání účtu

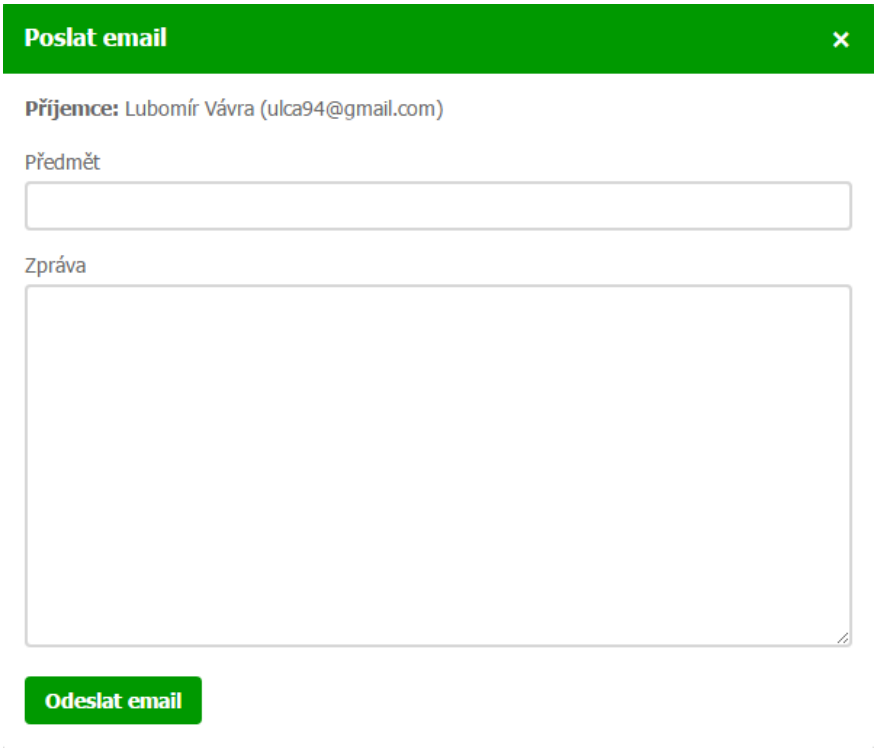
Poslední částí Nastavení je **Smazání účtu**. Tato funkce slouží k trvalému smazání účtu a všech jeho uložených údajů. Ještě před smazáním se ovšem aplikace dotáže, jestli chce uživatel opravdu svůj účet smazat, jelikož se uživatel mohl splést. 9

2. **Kontakty**



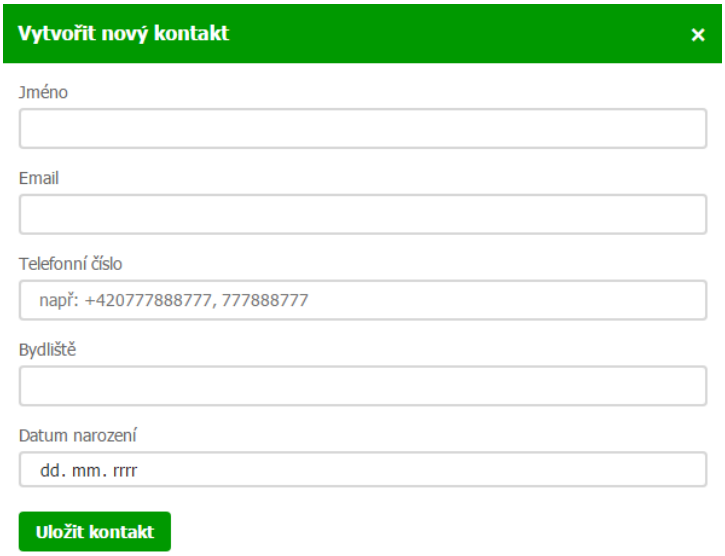
Obrázek 10: Zobrazení kontaktů

Na hlavní stránce kontaktů se zobrazují všechny uživatelovy kontakty (jméno kontaktu, email, případně i věk, telefonní číslo a bydliště). Na předchozím obrázku jsou již některé kontakty vložené v databázi. 10 V šedém navigačním menu může uživatel přepínat zobrazení kontaktů podle stáří nebo podle jména. Po najetí na libovolný kontakt se v pravém horním rohu objeví symbol pro smazání kontaktu. Před smazáním se aplikace uživatele zeptá, zda-li chce vybraný kontakt smazat. Pokud to uživatel potvrdí, vybraný kontakt se vymaže z databáze a kontakt již nebude uživateli dostupný. Další součástí zobrazeného kontaktu je odkaz v podobě emailu.



Obrázek 11: Posílání emailů

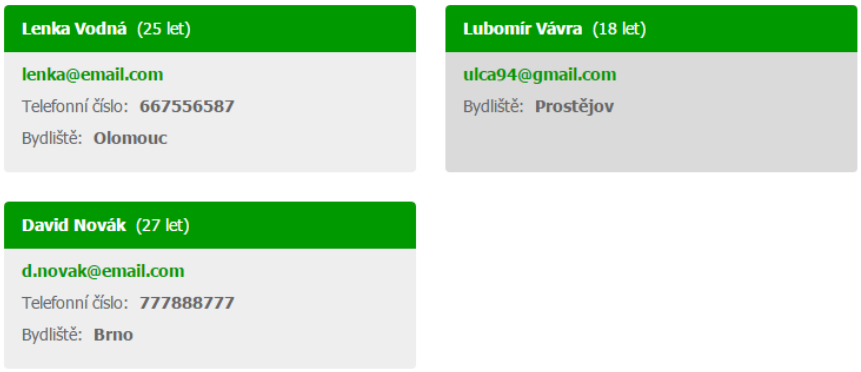
Po kliknutí na tuto emailovou adresu vyskočí okno, kde uživatel bude moci vyplnit předmět emailu a jeho text. Po kliknutí na „Odeslat email“ bude kontaktu poslán email s počátečním textem, ve kterém je napsáno, že tento email byl poslán z aplikace Diary.



Obrázek 12: Přidání nového kontaktu

Stránka **Kontakty** i **Kalendář** obsahují nahoře v obsahu stránky nástrojovou lištu. V této liště jsou odkazy **Přidat nový kontakt**, **Upravit kontakt** a **Odebrat kontakt**.

Hlavní funkcí této stránky (kontaktů) je přidávání nového kontaktu. Po kliknutí na tlačítko **Přidat nový kontakt** (vpravo nahoře) vyskočí okno formulářem pro vyplnění údajů nového kontaktu. 12 V tomto okně může uživatel zadat jméno kontaktu, email, telefonní číslo, bydliště a datum narození. Za povinné údaje jsou považovány Jméno a Email. Po kliknutí na **Uložit kontakt** se kontakt uloží do databáze. Tohle okno zmizí a na stránce již bude zobrazený nový kontakt.



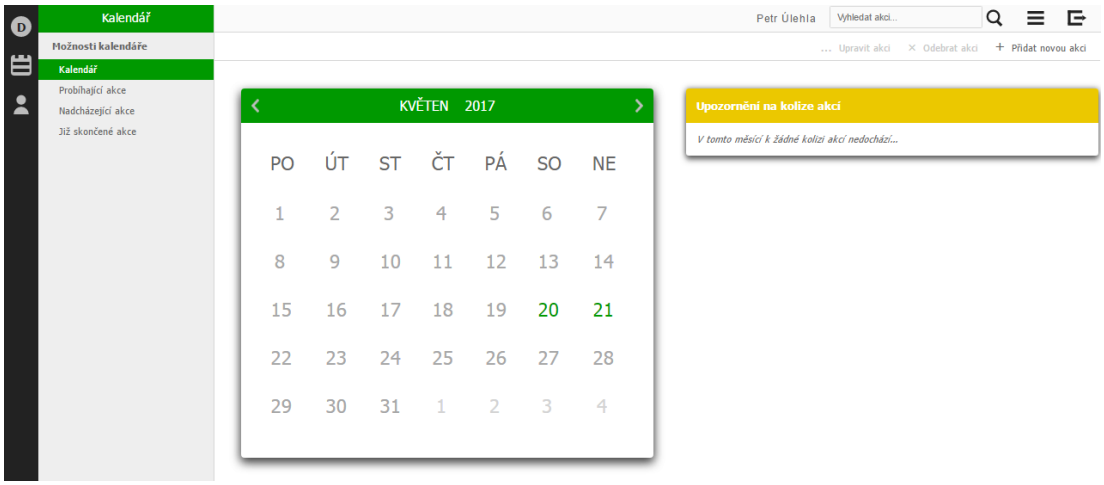
Obrázek 13: Vybrané kontakty

Obrázek 14: Úprava kontaktu

Jak jsem již zmínil, v nástrojové liště jsou tři odkazy, z nichž odkazy **Upravit kontakt** a **Odebrat kontakt** nejsou z počátku klikatelné. Jejich funkčnost je ovlivněna výběrem kontaktů. Po kliknutí na libovolný kontakt se změní jeho barva pozadí, což znamená, že následující kontakt je vybrán. 13 Pokud je tedy vybraný jeden kontakt, jsou oba odkazy dostupné. Pokud je vybrán více jak jeden kontakt, je dostupný pouze odkaz pro Odstranění kontaktů, jelikož aplikace povoluje pouze úpravu jednoho kontaktu.

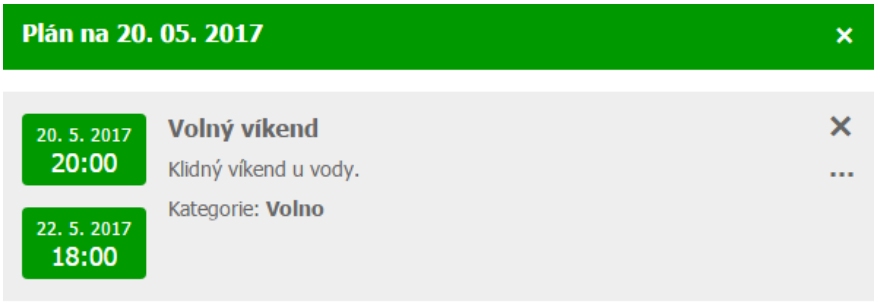
Po kliknutí na **Odebrání kontaktu** se vybrané kontakty odstraní. Po kliknutí na odkaz **Úpravy kontaktu** vyskočí okno s vyplněným formulářem vybraného kontaktu. 14 Uživatel zadá do formuláře nové údaje a po kliknutí se údaje o uživateli aktualizují.

3. **Kalendář**



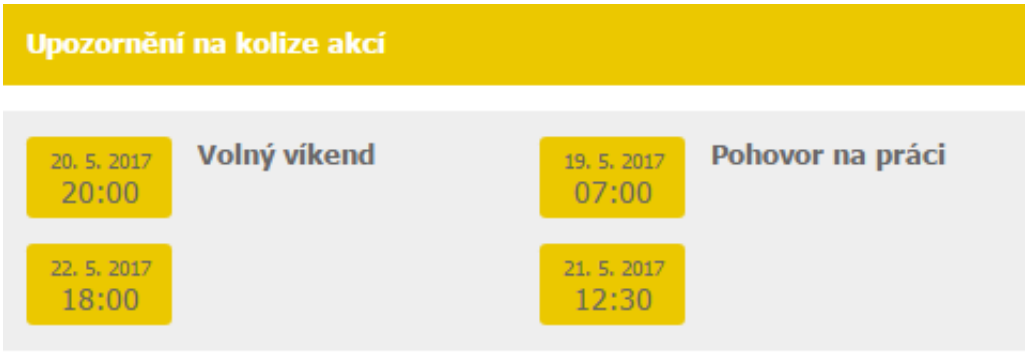
Obrázek 15: Stránka kalendáře

V šedém postranním menu jsou odkazy na kalendář, nadcházející, již uplynulé a probíhající akce. 15 Opět obsahuje nástrojovou lištu, jako v případě kontaktů. Funguje stejným způsobem. Stránka se liší především v zobrazení svých prvků (plánů či kontaktů)



Obrázek 16: Plán na jeden den

Kalendář zobrazuje data po měsících. Pokud je datum označeno zeleně, zanemná to, že v tomto datu je plán nějaké akce. Po rozkliknutí tohota data vyskočí okno, obsahující plán na vybraný den. 16



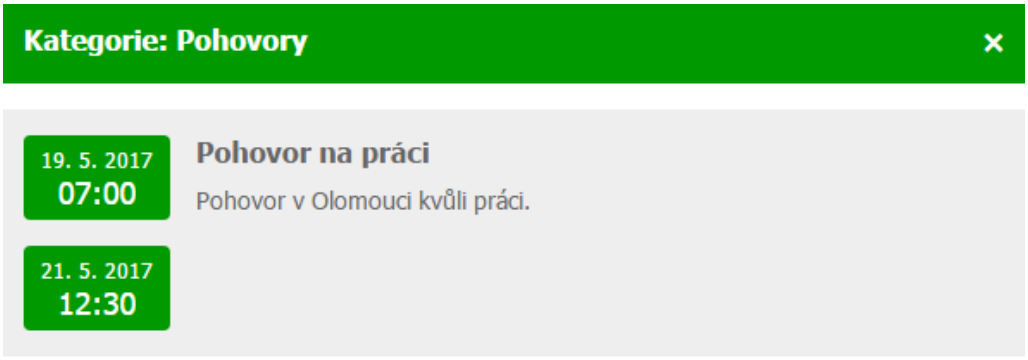
Obrázek 17: Kolize akcí

U kalendáře se vyskytuje okno pro upozornění na kolizi akcí. V tomto okně se vypíší akce (plány), které se časové překrývají. 17

Obrázek 18: Nadcházející akce

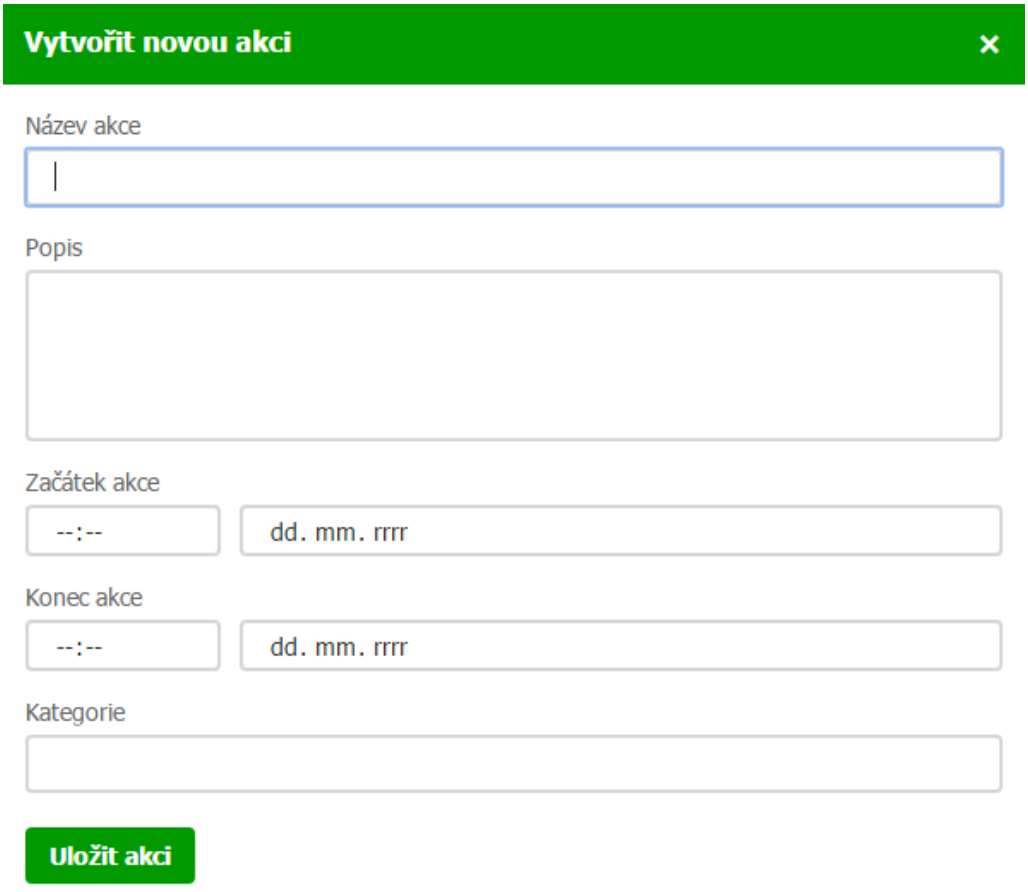
Stránka kalendář,jak jsem se již zmínil, obsahuje odkazy na probíhající, již uplynulé a nadcházející akce. Tyto akce se stejně jako v plánu na den a ve vyhledávání zobrazují pod sebe, podle data, kdy mají tyto akce proběhnout. 18

Úprava, odebrání a výběr akcí (plánů) je téměř stejný jako u kontaktů. Opět je nejsou dostupné všechny tlačítka a výběrem se mění barva pozadí vybrané akce.



Obrázek 19: Kategorie akcí

Na obrázku 16 můžeme vidět, že ve výpisu těhcto akcí jsou i nějaké odkazy. Jsou to odkazy na smazání a úpravu akcí (plánů) ale také obsahuje odkaz na kategorie. Po kliknutí na kategorii vyskočí nové okno, kde jsou vypsané všechny akce patřící do této kategorie. 19

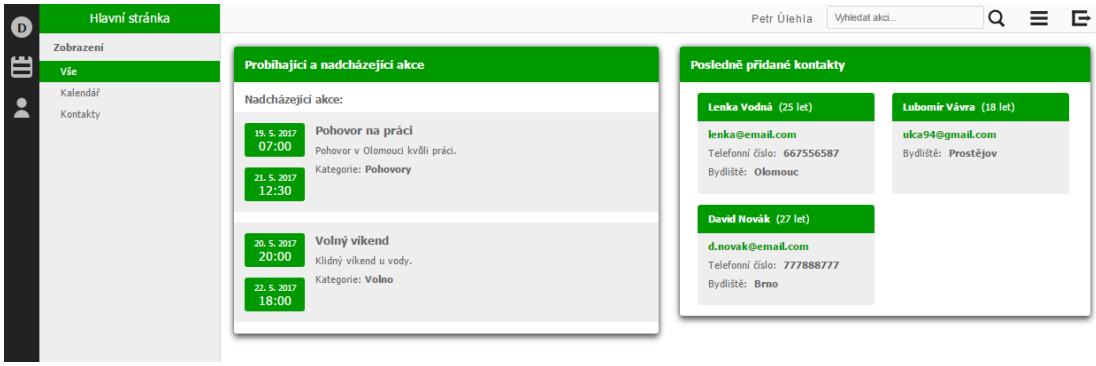


Obrázek 20: Přidání nové akce

Jak můžete vidět na obrázku 20, vytvoření akce má několik částí. Je to

název akce, popis akce, začátek a konec akce, včetně kategorie. Povinnými údaji je název akce, začátek a konec akce. Maximální rozsah je akce 7 dní. Kategorie je tedy volitelná.

4. **Hlavní stránka**



Obrázek 21: Hlavní stránka s vloženými údaji

Na obrázku 5 můžeme vidět hlavní stránku, když uživatel nemá uložené žádné akce (plány) nebo kontakty. Na obrázku 21 má uživatel vypsané jak kontakty, tak i jeho akce. Kontakty jsou seřazeny podle posledních přidaných, kde maximální počet zobrazení je 4 (nejvíce 4 kontakty). Akce jsou vypsané v samostatném bloku. Jedná se o akce, které právě probíhají, nebude ještě nezačaly. V šedém menu je rozdělení na Kalendář a Kontakty. Jsou to ty samé údaje jako na stránce 21. Pouze oddělují kontakty od akcí.

```
var array = @Html.Raw(Json.Encode(dates));
   $('#calendar').datepicker({
      inline: true,
      firstDay: 1,
      showOtherMonths: true,
      dayNamesMin: ['Ne', 'Po', 'Út', 'St', 'Čt', 'Pá', 'So'],
      monthNamesShort: ["Leden", "Únor", "Březen", "Duben", "Květen
          ", "Červen", "Červenec", "Srpen", "Září", "Říjen", "
          Listopad", "Prosinec"],
      dayNames: ['Neděle', 'Pondělí', 'Úterý', 'Středa', 'Čtvrtek',
          'Pátek', 'Sobota'],
      monthNames: ["Leden", "Únor", "Březen", "Duben", "Květen", "Č
          erven", "Červenec", "Srpen", "Září", "Říjen", "Listopad",
          "Prosinec"],
      dateFormat: "yy-mm-dd",
      prevText: "",
      nextText: "",
      beforeShowDay: function (date) {
         var string = jQuery.datepicker.formatDate('yy-mm-dd', date)
             ;
         return [array.indexOf(string) != -1, 'specialDay'];
      },
      onSelect: function (date) {
         $('.modal-actions').css("display", "block");
         $('#conts').siblings(".top").children("h4").text("Plán na "
              + date.substring(8, 10) + ". " + date.substring(5, 7)
             + ". " + date.substring(0, 4));
         $.ajax({
            url: "/User/ActionsByDate",
            type: "POST",
            data: { id : date},
            success: function (data) {
               $('#conts').html(data);
            }
         });
      }
   });
```

Zdrojový kód 22: Vyfiltrování akcí

```
var m = $('#calendar').datepicker('getDate').getMonth();
      var y = $('#calendar').datepicker('getDate').getFullYear();
      var newDate = new Date(y, m, 1);
      $(document).on('click', '.ui-datepicker-next', function () {
        newDate.setMonth(newDate.getMonth() + 1);
          var date = newDate.toString();
          $.ajax({
               url: "/User/SetWarnings",
               type: "POST",
               data: { id : (newDate.getMonth() + 1).toString(), id2
                    : newDate.getFullYear().toString() },
               success: function (data) {
                    $('#kol').html(data);
               }
          });
      });
```

Zdrojový kód 23: Kolize akcí

```
@for (int i = 0; i < newActs.Count - 1; i++)
{
     for (int j = i + 1; j < newActs.Count; j++)
     {
          if (!(newActs[i].To.Date < newActs[j].From.Date ||
              newActs[i].From.Date > newActs[j].To.Date))
          {
              if (!((newActs[i].To.Date == newActs[j].From.Date) &&
                  (newActs[i].To.TimeOfDay <= newActs[j].From.
                  TimeOfDay)))
              {
                   if (!((newActs[i].From.Date == newActs[j].To.Date)
                        && (newActs[i].From.TimeOfDay >= newActs[j].
                       To.TimeOfDay)))
                   {
          ...kód...
```

Zdrojový kód 24: Zjišťování kolizí

Závěr

Cílem této bakalářské práce bylo vytvořit reprezentaci klasického diáře.

Tato aplikace tedy umožňuje uchovávat uživatelovy akce (plány) a třídit je do kategorií. Samozřejmě nechybí ani jejich úprava či smazání. Tato aplikace také zajišťuje, že se uživatel dozví, jestli u nějakých akcí (plánů) nenastala časové kolize (jsou ve stejnou dobu) a dá mu vědět, že se tyto akce překrývají.

Tato aplikace také umožňuje uchovávat kontakty, upravovat je nebo odstranit.

Výhodou této webové aplikace je možnost k ní přistoupit z jakéhokoliv zařízení, které má přístup k internetu.

Conclusions

The goal of this bachelor thesis was to create representation of classic diary.

This application allows store users's actions (plans) and categorized them. Of course editing and deleting isn't missing. This application ensures, that user founds out, if at actions (plans) doesn't happen time collision (actions are running at the same time) and application let user know, that these actions coincides.

This application allows store user's contantcs, manage contacts a remove contacts.

Advantage of this web application is access from any device with Internet

Literatura

[1] ASP.NET. In Wikipedia: the free encyclopedia. [online]. 2016-10-02. [cit. 2017-05-18]. Dostupné z: https://cs.wikipedia.org/wiki/ASP.NET

[2] Model-view-controller. In Wikipedia: the free encyclopedia. [online]. 2016-07-16. [cit. 2017-05-18]. Dostupné z: https://cs.wikipedia.org/wiki/Model-view-controller

[3] HyperText Markup Language. In Wikipedia: the free encyclopedia. [online]. 2017-03-01. [cit. 2017-05-18]. Dostupné z: https://cs.wikipedia.org/wiki/HyperText_Markup_Language

[4] Kaskádové styly. In Wikipedia: the free encyclopedia. [online]. 2017-05-15. [cit. 2017-05-18]. Dostupné z: https://cs.wikipedia.org/wiki/Kask%C3%A1dov%C3%A9_styly

[5] JavaScript. In Wikipedia: the free encyclopedia. [online]. 2017-01-11. [cit. 2017-05-18]. Dostupné z: https://cs.wikipedia.org/wiki/JavaScript

[6] AJAX. In Wikipedia: the free encyclopedia. [online]. 2016-12-05. [cit. 2017-05-18]. Dostupné z: https://cs.wikipedia.org/wiki/AJAX

[7] C Sharp. In Wikipedia: the free encyclopedia. [online]. 2016-10-18. [cit. 2017-05-18]. Dostupné z: https://cs.wikipedia.org/wiki/C_Sharp

[8] HOMER, Alex; SUSSMAN, Dave; HOWARD, Rob. A First Look at ASP.NET v.2.0. Addison Wesley, 2003. ISBN 978-0321228963.

[9] POKORNÝ, Jaroslav. Databázové systémy a jejich použití v informačních systémech. 1. vyd. Praha, 313 s: Academia, 1992. ISBN 8020001778.

[10] POKORNÝ, Jaroslav. Databázové systémy a jejich použití v informačních systémech. 1. vyd. Praha, ČVUT, 1999. ISBN 80-01-01935-7.

[11] SCOTT, Michael L. Programming Language Pragmatics. San Francisco, Morgan Kuffmann Publishers, 2000. ISBN 1-55860-442-1.

A Obsah přiloženého CD/DVD

Na samotném konci textu práce je uveden stručný popis obsahu přiloženého CD/DVD, tj. jeho závazné adresářové struktury, důležitých souborů apod.

`bin/`
: Obsahuje kompletní adresářovou strukturu webové aplikace v archivu ZIP, včetně SQL databázového skriptu.

`doc/`
: Obsahuje text práce ve formátu PDF a archiv ZIP se zdrojovým textem.

`src/`
: Obsahuje adresářovou strukturu webové aplikace v archivu ZIP s SQL databázovým skriptem.

`readme.txt`
: Obsahuje instrukce pro spuštění aplikace.