Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Vývoj aplikace pro státní správu

Bakalářská práce

Autor: Tomáš Trnka
Studijní obor: Aplikovaná informatika

Vedoucí práce: prof. RNDr. PhDr. Antonín Slabý, CSc.

Hradec Králové duben 2023

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 24.4.2023 Tomáš Trnka

Poděkování:

Děkuji vedoucímu bakalářské práce prof. RNDr. PhDr. Antonínu Slabému, CSc. za metodické vedení mé bakalářské práce a za ochotu se mnou vše konzultovat.

Anotace

Webová aplikace je vytvořena pro pracovníky pracující ve státní správě, především pro personální oddělení a představené zaměstnance, kteří mají povinnost zveřejňovat své profesní životopisy. Aplikace komplexně zajišťuje proces zveřejnění životopisů. Díky této aplikaci jsou představení ředitelé a ostatní přestavení schopni sami na webové stránky své životopisy vkládat, upravovat a případně je jinak aktualizovat. Aplikace je propojená s personálním systémem nebo telefonním seznamem daného úřadu a díky tomu je schopná sama vyhodnocovat, zda je nebo není nutné u dané osoby zveřejnit životopis. Pokud aplikace vyhodnotí, že daná osoba je povinna svůj životopis zveřejnit, sama této osobě zašle notifikaci do pracovní emailové schránky.

Annotation

Application development for state administration

The web application is designed for employees working in the government administration, especially for HR departments and the employees who are obliged to publish their professional CVs. The application comprehensively handles the CV publishing process. Thanks to this application, directors and other officials are able to upload, edit and otherwise update their CVs on the website themselves. The application is linked to the personnel system or telephone directory of the office and is therefore able to assess for itself whether or not a person's CV needs to be published. If the application assesses that a person is required to publish his or her CV, it will send a notification to the person's work email inbox.

Obsah

1 Úvod

Dne 14. prosince 2015 schválila vláda ČR Akční plán boje s korupcí na rok 2016. V něm je mj. obsaženo podpůrné opatření, které má zajistit transparentnost při obsazování vedoucích pozic ve státní správě. Tímto opatřením je povinnost zveřejnit profesní životopis představených od úrovně ředitelů odborů, kteří uspěli ve výběrových řízeních, a to na webových stránkách příslušných služebních úřadů.[1]

Pracovníci státní organizace mají na starosti denně spoustu administrativních úkolů. Jedním z takových úkolů je i zveřejňování profesních životopisů. Bez užití aplikace proces zveřejňování životopisů řeší více pracovníků naráz, jedná se především o pracovníky personálního oddělení, kteří primárně hlídají seznam představených a splnění povinnosti zveřejnění životopisu. Dále představený je povinen předat svůj životopis personálnímu oddělení, které následně jeho zveřejnění řeší s oddělením informatiky. Je třeba, aby všechny tyto dotčené osoby mezi sebou spolupracovali, neboť by v opačném případě docházelo k prodlení a k opakovanému upozorňování o nesplnění dané povinnosti. V rámci snížení vytíženosti pracovníků a zároveň snaze snížit počet každodenních opakujících se administrativních činností, je vhodné pokusit se zjednodušit a zautomatizovat tento proces a vytvořit aplikaci tak, aby sloužila ve prospěch všech dotčených zaměstnanců.

Cílem této práce je vytvořit aplikaci, která bude sloužit jako funkční šablona pro oddělení informatiky jednotlivých úřadů, kteří mohou tuto aplikaci přijmout s využitím vlastní databáze zaměstnanců či ji upravit dle potřeb daného úřadu. Následně funkční aplikaci předají k užívání personálnímu oddělení a představeným.

Mezi hlavní moduly aplikace patří modul pro správu životopisů, který umožňuje jednotlivé životopisy spravovat. Při zařazení nového představeného do databáze zaměstnanců aplikace automaticky vyhodnotí nutnost zveřejnit profesní životopis a o této nutnosti zašle notifikaci přímo samotnému představenému. Pokud

---

[1] Akční plán boje s korupcí na rok 2016, Úřad vlády České republiky Ministr pro lidská práva, rovné příležitosti a legislativu a předseda Rady vlády pro koordinaci boje s korupcí, Předkladatel: ministr pro lidská práva, rovné příležitosti a legislativu Praha, prosinec 2015

tuto povinnost představený do daného časového úseku nesplní a svůj životopis do aplikace nevloží, aplikace ho automaticky upozorní na nesplnění povinnosti, a to i opakovaně. Představený se přes jednoduchý odkaz, který bude obsažen v samotné notifikaci, dostane do aplikace, kam bude sám schopen svůj životopis vložit a zveřejnit ho. Pokud dojde k přerušení pracovního výkonu a představený bude vyřazen ze seznamu pracovníků, aplikace automaticky životopis na webových stránkách úřadu smaže.

2 Návrh modelu

Pro účely vývoje aplikace bylo vytvořeno vhodné vzorové prostředí, aby bylo možné aplikaci navrhnout a následně naprogramovat.

Prvním krokem při tvorbě konkrétní aplikace by měla být schopnost popsat proces, který probíhá bez automatizace činnosti. Konkrétně v této aplikaci se jedná o několik dílčích činností různých pracovníků, které musejí pro zveřejnění životopisu provést (např. personální pracovník, představený pracovník a správce webových stránek).

Po určení těchto konkrétních činností je dále možné přistoupit k samotnému návrhu aplikace, která by měla tyto činnosti zastoupit. Měla by je zjednodušit, snížit počet dotčených osob a zrychlit celý proces zveřejnění. Výsledkem sběru těchto dat vznikne celkový rozsah aplikace.

2.1 Popis procesu bez systému

Do procesu zveřejňování profesních životopisů (v případě, že neexistuje aplikace na automatické vkládání) je zapojeno více pracovníků úřadu. Jedná se o pracovníka personálního oddělení, který má na starosti správu profesních životopisů. Ten kontroluje, zda byla splněna zákonná povinnost a životopis byl zveřejněn, případně urguje jeho zveřejnění. Dalším je samotný představený, který má povinnost vyplnit profesní životopis. Posledním je správce webových stránek, který zajišťuje vyvěšení dokumentu na portálu úřadu. K úspěšnému zveřejnění profesního životopisu je třeba, aby tyto tři osoby mezi sebou spolupracovali, což

může být v praxi složité z časových i jiných důvodů, a tím se vyvěšení životopisu pozdržuje.

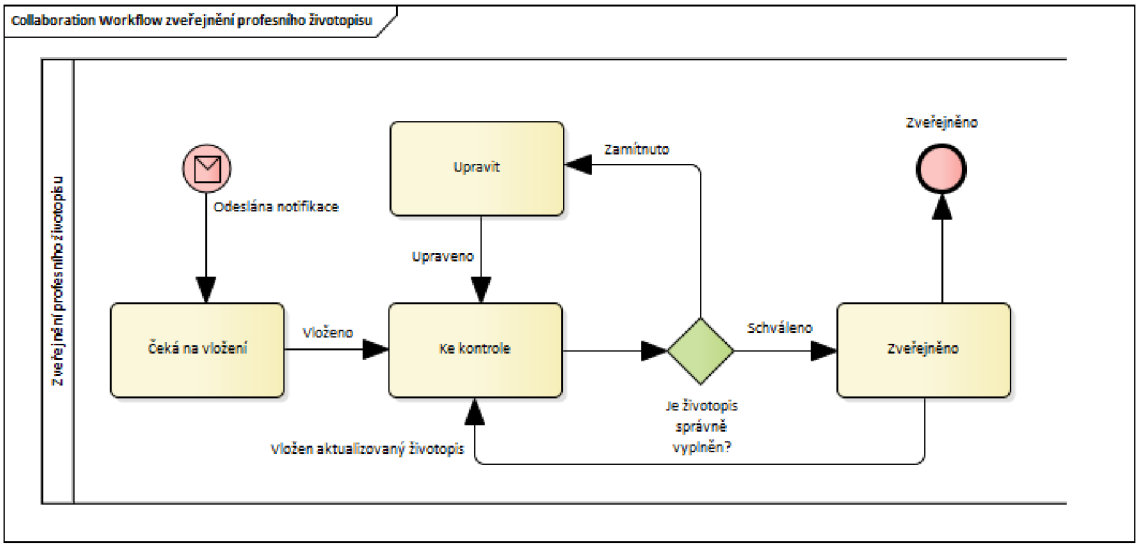
Samotný proces zveřejňování probíhá tak, že personální pracovník provede kontrolu organizační struktury. Výsledkem této kontroly je zjištěno, že existuje v rámci organizace představený pracovník, u kterého je potřeba vyplnit profesní životopis. Personální pracovník zašle žádost představenému, ve které ho žádá o zaslání profesního životopisu. Představený vyplní svůj profesní životopis a zašle ho zpět personálnímu pracovníkovi. Ten provede kontrolu přijatého dokumentu. V případě, že je v dokumentu chyba, pošle dokument zpět představenému. V případě správnosti dokumentu ho předá správci webových stránek s požadavkem k jeho zveřejnění. Správce webových stránek následně zveřejní životopis představeného na portálu organizace. Jestliže je nutné aktualizovat profesní životopis, je potřeba tento proces opakovat.

2.2 Návrh systému pro zveřejňování životopisů

Díky zavedení automatizace do procesu zveřejňování profesních životopisů se bude jejich spravování týkat menšího počtu pracovníků, a to zejména samotného představeného, který má povinnost svůj životopis zveřejnit a správce aplikace, kterým je pracovník personálního oddělení. Proces automatizovaného zveřejňování životopisů probíhá v několika následujících krocích.

Systém provede kontrolu zaměstnanců organizace z organizační struktury v pravidelných intervalech a vyhledá všechny pracovníky, kteří odpovídají parametrům pro potřebu vyplnění profesního životopisu. Těmto vybraným pracovníkům systém zašle notifikační zprávu, ve které budou požádáni o vložení životopisu. Představený vstoupí do aplikace a životopis vloží. Správci aplikace přijde notifikace s informací, že představený vložil profesní životopis do systému, následně správce vstoupí do systému a provede kontrolu životopisu. Dále ho buď schválí nebo zamítne. Pokud je třeba životopis opravit, zadá žádost o úpravu do systému a ten zašle notifikaci představenému o nutnosti úpravy. V případě, že je dokument schválen, systém zveřejní profesní životopis na webovém portálu organizace.

V případě nutnosti aktualizace informací v životopisu může představený upravit svůj životopis a schvalovací část je opakována.



Obrázek 1 – Workflow systému

2.3 Model systému

Z popisu návrhu systému je vytvořen model systému, který lépe zobrazí rozsah a náročnost aplikace. V modelu jsou aktéři, kteří mohou vykonávat sadu určitých akcí. Aktéři reprezentují jednotlivé role, které jsou nastaveny v aplikaci. Sada akcí je „Případ užití“ neboli „Use Case“, případně je lze označit jako jednotlivé funkcionality systému.

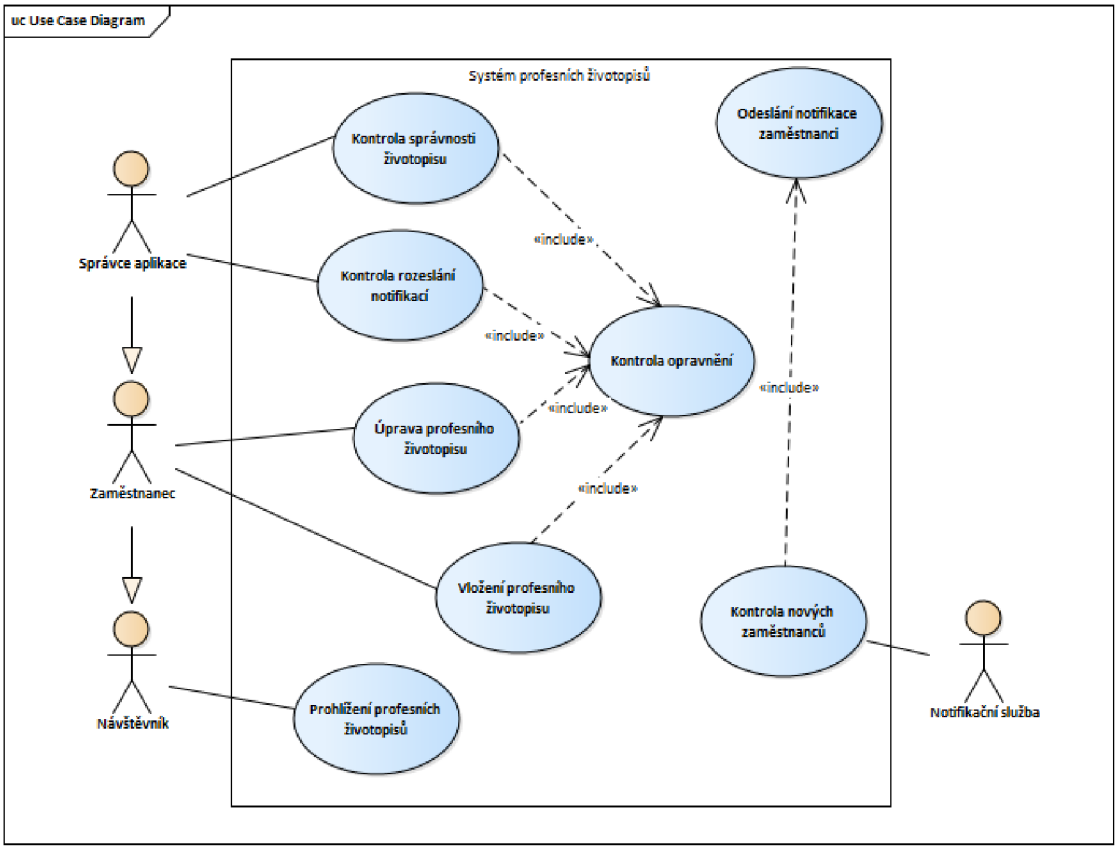
Neověřený uživatel (návštěvník webového portálu úřadu) má oprávnění pro prohlížení jednotlivých profesních životopisů. Návštěvník otevře příslušnou webovou stránku, kde systém načte seznam profesních životopisů. Návštěvník si jeden z nich vybere a systém mu zobrazí detail vybraného záznamu.

Představený má oprávnění neověřeného uživatele a jako další dostupnou funkcionalitu má vkládání profesních životopisů a jejich úpravu. Pokud představený vkládá nový dokument, tak systém zobrazí formulář pro vložení profesního životopisu. Představený vloží vyplněný dokument, následně systém uloží tento dokument do souborového systému a data o profesním životopisu do databáze.

V případě, že představený upravuje stávající záznam, je jím nejdříve vybrán záznam k úpravě. Dále se systém zachová stejně jako ve výše uvedeném postupu.

Správce aplikace má oprávnění představeného, a navíc funkce potřebné pro správu aplikace. Kontrola správnosti životopisu je akce, kdy správce požaduje seznam profesních životopisů, které jsou ve stavu ke kontrole. Systém vyhledá v databázi všechny profesní životopisy, které tomuto pokynu vyhovují a zobrazí uživateli jejich seznam. Ze seznamu správce vybere požadovaný životopis, který chce zkontrolovat a systém následně zobrazí detail záznamu. Pokud je profesní životopis z pohledu správce vyplněn správně, správce iniciuje zveřejnění profesního životopisu a systém vystaví životopis na webovém portálu úřadu. V opačném případě je představenému zaslána notifikace, ve které je požádán o doplnění záznamu. Kontrola rozesílání notifikací je akce vyvolaná správcem, kdy zadá pokyn systému a ten vyhledá seznam notifikací, které byly rozeslány představeným a zobrazí ho správci.

Notifikační služba je automaticky spouštěný proces v pravidelných intervalech, který vyhledá všechny představené, kteří mají povinnost vložit svůj profesní životopis, ale ještě tak neučinili. Systém načítá seznam představených a těmto pracovníkům zasílá notifikace. Nakonec systém uloží informace o rozeslaných notifikacích.

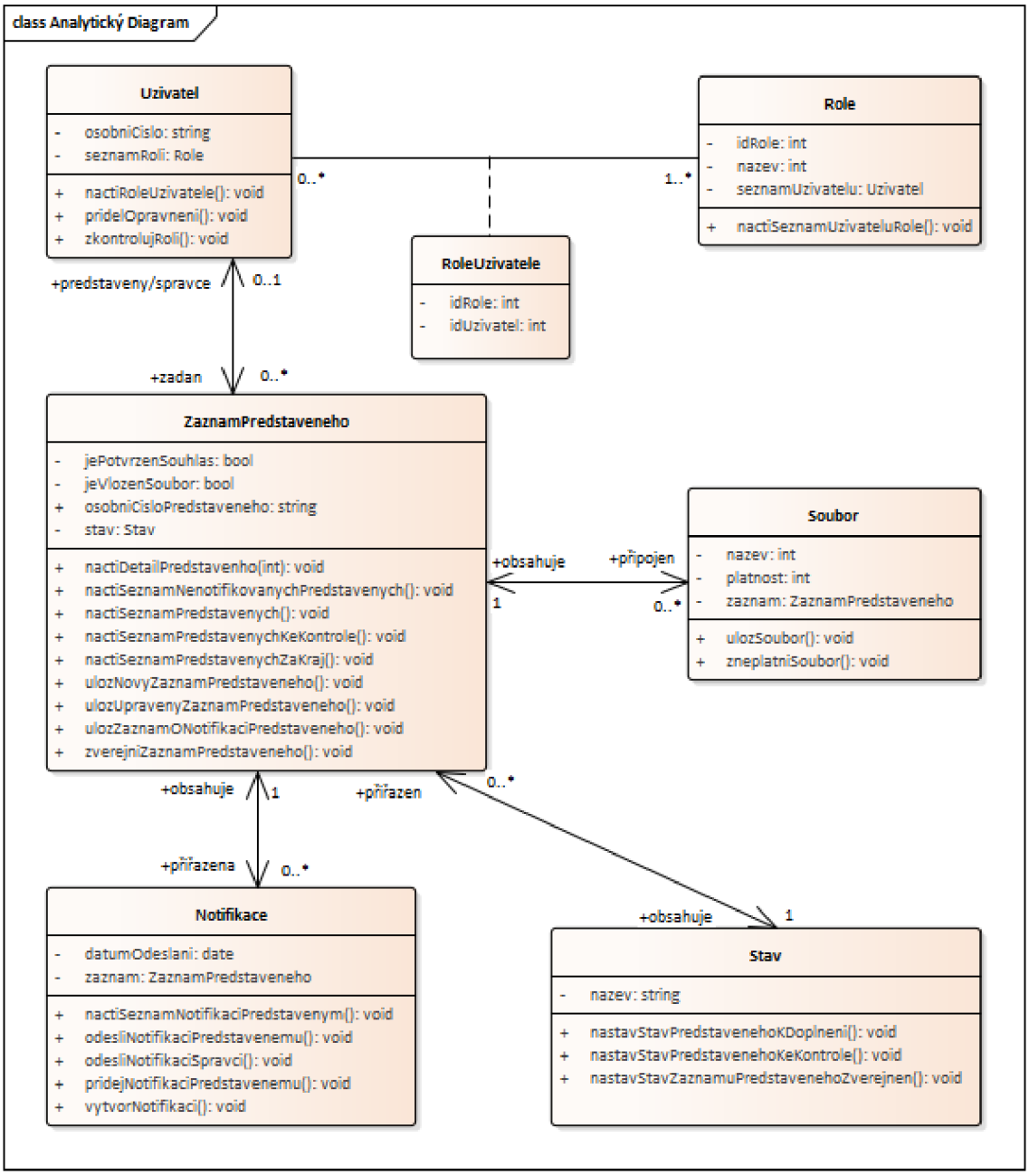


Obrázek 2 – Případy užití systému

2.4 Analytický diagram systému

Analytický diagram tříd vychází z „UseCase“ diagramu a zobrazuje základní část systému, ze které je možné v dalších krocích naprogramovat aplikační vrstvu systému a vytvořit databázi.

Návrhem modelu aplikace je určeno, že systém je rozdělen na veřejnou část dostupnou každému, kdo aplikaci spustí a dále privátní část, která je dostupná pouze autorizovaným uživatelům. Do privátní části mají přístup pouze uživatelé, kteří mají přiřazenou roli „Predstaveny“ nebo „Spravce“. Sada funkcionalit aplikace dostupných pro uživatele je určena podle toho, jakou má uživatel přidělenou roli. Do modelu je zavedena třída „Uzivatel“. Uživatelé, kteří jsou pověřeni rolemi, mají možnost spravovat záznamy představených. Třída „ZaznamPredstaveneho“ obsahuje informaci o stavu, souborech a notifikacích. Z tohoto důvodu byly do modelu zavedeny třídy „Stav“, „Notifikace“ a „Soubory“.



Obrázek 3 – Návrh analytického diagramu systému

3 Vývoj aplikace

Aplikace se skládá ze tří samostatných modulů. Tyto moduly mají společné úložiště dat, do kterého zapisují nebo z něj čerpají data. Hlavním důvodem rozdělení aplikace na více modulů je nutnost autorizace uživatelů, kteří vstupují do privátní části.

Prvním z těchto modulů je modul pro správu, kterým lze z pozice správce spravovat veškeré vložené profesní životopisy a z pozice představeného je možné do aplikace životopisy vkládat. Tento modul zastupuje privátní část aplikace z návrhu.

Druhý modul pro zobrazení zajišťuje načtení dat do formátovaného seznamu a jejich zobrazení na webovém portálu úřadu. Tento modul bude veřejně dostupný.

Posledním modulem je služba, která zajišťuje rozesílání informačních zpráv představeným a správcům aplikace. Zároveň modul kontroluje nové pracovníky, u kterých je třeba splnit povinnost vyvěsit na webový portál úřadu svůj životopis.

Modul pro správu a modul pro zobrazování dat jsou tvořeny metodou MVC. Pro zavedení projektů jsou použity šablony ASP.NET framework 4.7 a ASP.NET Core web API, které tuto metodu využívají. Pro vývoj aplikací využívá tato metoda vzor typu model, kontrolér a pohled. Modely jsou třídy, které replikují data databáze. Kontroléry zpracovávají požadavky uživatelů, načítají data z modelů a zpracované je předávají pohledům. Pohledy jsou dynamicky generované html šablony, které využívají data z kontroléru. Ke správnému směřování URL adres slouží třída „RouteConfig“, ve které je nastaveno, že aplikace bude automaticky směřovat na kontrolér s názvem „Home“ a v tomto kontroléru bude hledat metodu s názvem „Index“, pokud nebude parametr kontroléru ani parametr metody definován.[2]

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new {
                controller = "Home",
                action = "Index",
                id = UrlParameter.Optional
            }
        );
    }
}
```

Zdrojový kód 1 – Konfigurace směřování

[2] Anderson Rick, Přidání kontroleru, Microsoft Documents, 13.5.2021, https://docs.microsoft.com/cs-cz/aspnet/mvc/overview/getting-started/introduction/adding-a-controller

Při vývoji aplikace je třeba dále řešit její napojení na databázi. K tomu účelu je vhodné použít Entity Framework, který zajišťuje objektově relační mapování. Pro napojení aplikace na databázi je do konfiguračního souboru aplikace přidán připojovací řetězec, do kterého jsou vyplněny následující údaje:

- **Data source** – požaduje IP adresu serveru
- **Initial catalog** – název databáze na serveru
- **User** – uživatelské jméno pro přihlášení do databáze
- **Password** – heslo pro přihlášení do databáze

```
<connectionStrings>
    <add name="DbModel"
         connectionString="data source=IP_SERVERU;initial catalog=DATABAZE;
           user id=UZIVATELSKE_JMENO;password=HESLO;
           MultipleActiveResultSets=True; App=EntityFramework"
         providerName="System.Data.SqlClient" />
</connectionStrings>
```

Zdrojový kód 2 - Náhled na konfiguraci připojovacího řetězce – webconfig

Druhým krokem k připojení na databázi je zavedení třídy databázového kontextu, kde je zvolen připojovací řetězec „DbModel“ a napojeny všechny potřebné třídy, které představují databázové tabulky jako objekty. [3]

```
public class DbModel : DbContext
{
    public DbModel() : base("name=DbModel") {}

    public virtual DbSet<Urad> Urad { get; set; }
    public virtual DbSet<Predstaveny> Predstaveny { get; set; }
}
```

Zdrojový kód 3 - Náhled na konfiguraci databázového modelu

Poslední částí je vytvoření tříd, které obsahují stejné vlastnosti jako sloupce databázových tabulek.

[3] Práce s DbContext, Microsoft Documents, 30.11.2021, https://docs.microsoft.com/cs-cz/ef/ef6/fundamentals/working-with-dbcontext

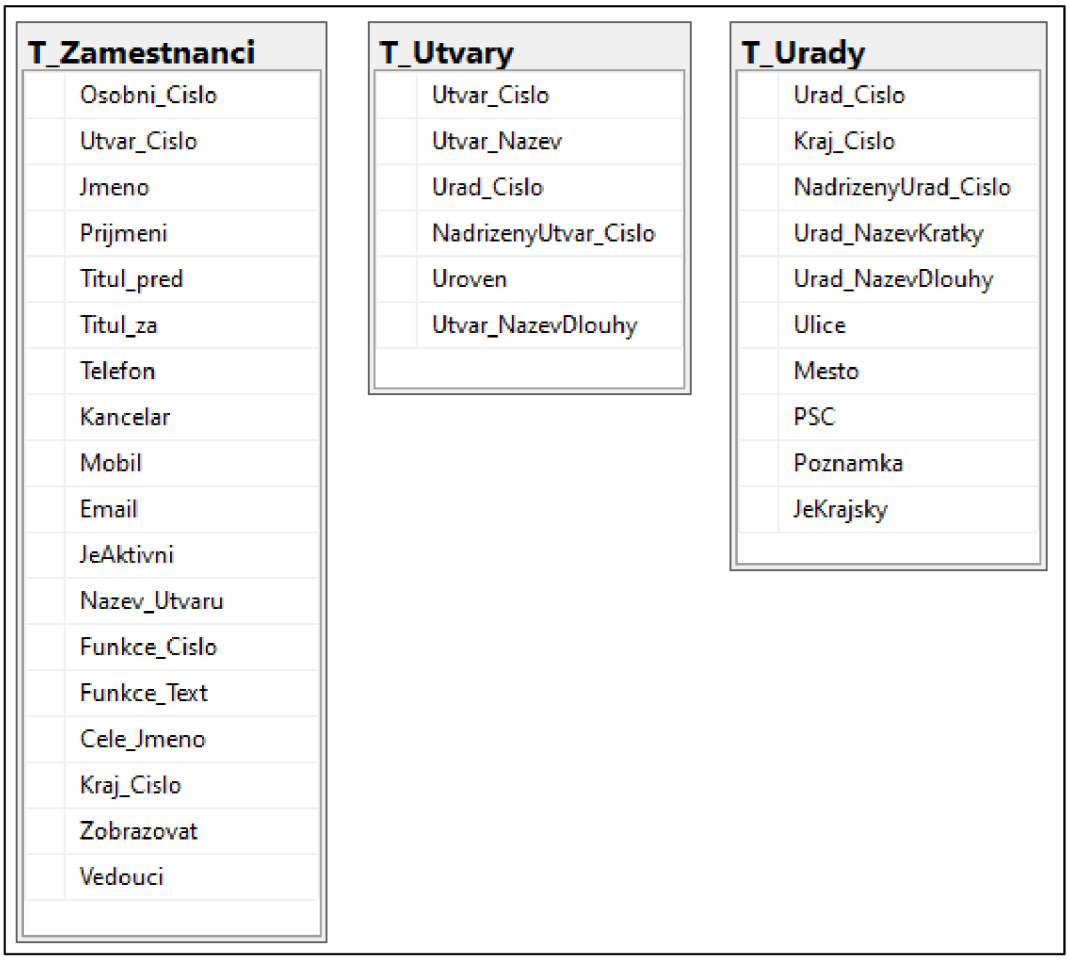
3.1 Databáze

Databáze je hlavním prostředkem k uchovávání doplňujících dat. Soubory profesních životopisů jsou uloženy v souborovém systému serveru.

Z důvodu načítání informací o představených, jejich zařazení a řazení dle organizační struktury, je pro tuto aplikaci předpoklad, že daný úřad poskytne export těchto dat z organizační struktury úřadu ve formě databázových tabulek. Tyto data jsou z organizační struktury načítána automatickými procedurami, které data následně exportují do databázových tabulek. Pro potřeby aplikace je nutné z organizační struktury úřadu získat následující data:

- Seznam zaměstnanců, který nese potřebné informace pro aplikaci o každém pracovníkovi úřadu. Na základě těchto dat lze určit, jestli daný pracovník je představený a má tedy možnost vložit svůj profesní životopis. Dále jsou zde některé osobní údaje pracovníka v podobě jména, příjmení, emailové adresy a telefonního čísla a dále čísla útvaru, na kterém je pracovník zařazen.
- Seznam, kde jsou načteny názvy a čísla útvarů, včetně informace o nadřízeném útvaru a čísla úřadu, na který je útvar zařazen.
- Seznam názvů a čísel úřadů, který dále nese informace o kraji, ve kterém je umístěn.

V databázi aplikace jsou vytvořeny tabulky “T_Zamestnanci”, “T_Utvary” a “T_Urady”, do kterých jsou automaticky každý den načítána aktualizovaná výše zmíněná data. Z důvodu složitosti každodenního exportu dat neobsahují tyto tabulky vzájemné vazby.



Obrázek 4 - Databázový diagram organizační struktury

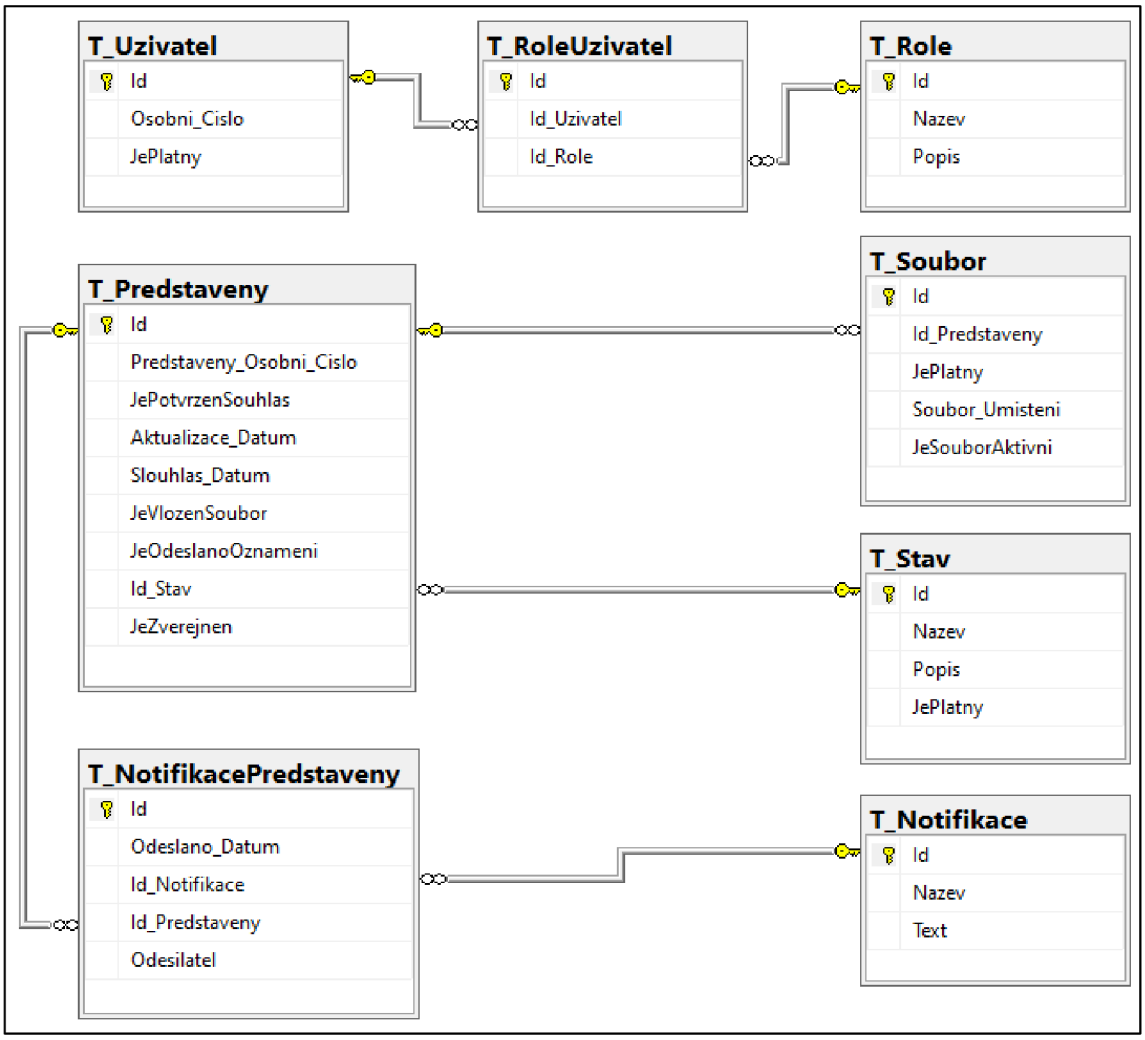
Pro privátní část aplikace je nutné uchovávat seznam uživatelů, kteří do této části mají přístup. Tito uživatelé jsou identifikováni pomocí osobního čísla, které mají přiřazeno z organizační struktury úřadu. Toto identifikační číslo uživatelů je zapsáno v tabulce “T_Uzivatel”, a zároveň je toto číslo obsaženo v tabulce organizační struktury “T_Zamestnanci”.

Při vstupu do privátní části aplikace je třeba splnit několik podmínek, bez kterých nebude přístup umožněn. Jedná se o ověření, že osobní číslo uživatele je zařazeno do tabulky “T_Uzivatel” a zároveň, že je tento záznam platný. Nakonec je třeba mít přidělený seznam rolí, ze kterého bude patrné, jaké funkce jsou pro daného uživatele dostupné. Tyto role jsou uloženy v tabulce “T_Role”, která má vztah M:N na tabulku “T_Uzivatel”. K propojení těchto tabulek slouží vazební tabulka “T_RoleUzivatel”.

Stěžejní částí databáze, ale i celé aplikace, jsou záznamy představených. K uchovávání informací jednotlivých záznamů slouží tabulky. Tabulka “T_Predstaveny”, která nese informace o potvrzeních, jako jsou např. souhlas se zveřejněním, datum vložení atp. Tato tabulka je, stejně jako záznam uživatele v tabulce “T_Uzivatel”, nepřímo spojena s představeným prostřednictvím osobního čísla. Tabulka “T_Predstaveny” má vztah 1:N s tabulkou “T_Soubor”. Tabulka “T_Soubor” uchovává všechny informace o přiložených souborech. Mezi ně patří řetězec cesty k souboru v souborovém systému, dále informace, zda je soubor platný a zda má být zveřejněn.

Vzhledem k navrženému schvalování jednotlivých záznamů správcem aplikace je k tabulce “T_Predstaveny” připojena informace o stavu, ve kterém se záznam nachází. Tato informace je načítána z tabulky “T_Stav”, která je k tabulce “T_Predstaveny” ve vztahu 1:N.

Modul aplikace, který zajišťuje rozesílání notifikací, ukládá informace o odeslané notifikaci do tabulky “T_NotifikacePredstaveny“, ve které je uloženo, komu byla notifikace poslána, kdy a o jaký typ notifikace se jedná. Typ notifikace je vybrán ze seznamu „T_Notifikace“. Tabulky „T_Notifikace“ a „T_Predstaveny“ jsou přes tabulku „T_NotifikacePredstaveny“ ve vzájemném vztahu M:N. Tedy každý záznam představeného může mít více odeslaných notifikací.



Obrázek 5 - Databázový diagram systému profesních životopisů

3.2 Modul pro správu profesních životopisů

Modul pro správu zajišťuje možnost správy dat aplikace (přednostně se jedná o zápis, úpravu a zveřejnění životopisů). Jedná se o část systému, do kterého mají přístup pouze oprávnění uživatelé. Správa zahrnuje možnost vložit životopis přes formulář, zobrazit seznam všech představených, načíst detail představeného a seznam notifikací.

Tento modul je rozdělen na dvě vrstvy (API a uživatelské rozhraní), které spolu komunikují pomocí textových souborů typu Json.

3.2.1 API

API je část modulu, která zajišťuje komunikaci klientské části s daty, která jsou uložena v databázi. Modul zajišťuje zpracování přijatého požadavku, načtení dat z databáze a předání dat zpět uživatelskému rozhraní.

Uživatel klade prostřednictvím webového rozhraní požadavky na čtení, úpravu a zápis dat do databáze. Rozhraní API zajišťuje autentizaci (ověření oprávnění) uživatele a koordinaci jednotlivých kroků. Do webového rozhraní se předávají jen data, která uživatel v daný moment potřebuje, a tím se šetří velikost přenesených dat.

Pro některé operace jsou vytvořeny v databázi pohledy do databázových tabulek, které spojují data aplikace a data organizační struktury.

Připojení API do databáze je zajištěno připojovacím řetězcem v souboru appsettings.json.

```
"ConnectionStrings": {
        "DbE": "data source=NazevDatabazovahoServeru; initial catalog=NazevDatabaze;
                user id=Uzivatel; password=Heslo;MultipleActiveResultSets=True;
                App=EntityFramework"
},
```

Zdrojový kód 4 - Náhled na konfiguraci připojovacího řetězce – appsettings

Následně je tento připojovací řetězec z konfigurace předán databázovému kontextu.

```
services.AddDbContext<DbE>(options =>
       options.UseSqlServer(Configuration.GetConnectionString("DbE")));
```

Zdrojový kód 5 - Zavedení služby pro databázový kontext

Databázový model je do API vygenerován příkazem „Scaffold-DbContext“. Tomuto příkazu jsou nastaveny parametry, které zpřesňují generování. Mezi ně patří parametr určující připojení k databázi („`name=ConnectionStrings:DbE`“), poskytovatele („`Microsoft.EntityFrameworkCore.SqlServer`“), název balíku, do kterého se má model vygenerovat („`-o Model`“), parametr, který určuje přepsání dosavadního

modelu - pokud existuje („`-force`“). Dále je nastaven název databázového kontextu („`-Context DbE`“). [4]

Pro dotazy do databáze je zavedeno rozhraní Data Access Object. Toto rozhraní definuje obecné CRUD metody pro vyhledání všech záznamů, jednoho záznamu dle identifikačního čísla, založení záznamu, úpravy záznamu a smazání záznamu.

```
public interface IDaoBase<T>
{
    Task<IList<T>> GetAll();
    Task<T> GetById(int id);
    Task<T> Create(T entity);
    Task Update(T entity);
    Task Detele(T entity);
}
```

Zdrojový kód 6 – Náhled na rozhraní data access object

Toto rozhraní implementuje třída „DaoBase“, která zavádí logiku pro jednotlivé metody. Pro jednotlivé entity jsou vytvořeny vlastní třídy, které dědí základní metody z třídy „DaoBase“ a obsahují další vlastní metody, které zajišťují dotazy do databáze. Veškeré dotazy aplikace do databáze jsou tak (odděleny) ve vlastní části.

Pro zajištění autentizace uživatelů, kteří přistupují do privátní části aplikace, slouží pomocná třída „Uzivatel“, která má dvě hlavní úlohy. Načtení osobního čísla z přihlašovacího jména, kterým jsou zaměstnanci úřadu přihlášeni do svého služebního počítače. Druhou úlohou je ověření rolí uživatele. Na základě třídy „HttpContext“ a nastavení Windows autentizace je možné získat osobní číslo, kterým se zaměstnanci přihlašují do sítě [5]. Použití Windows autentizace pro aplikaci je nastaveno v konfiguračním souboru „launchSettings.json“. Pokud je autentizace povolena, je možné použít uživatelské jméno z třídy „HttpContext“. Tato metoda

[4] Referenční Entity Framework Core nástrojů – Správce balíčků Console v Visual Studio, 14.01.2022, https://docs.microsoft.com/cs-cz/ef/core/cli/powershell#scaffold-dbcontext

[5] HttpContext User Vlastnost, System web, https://docs.microsoft.com/cs-cz/dotnet/api/system.web.httpcontext.user?view=netframework-4.8&viewFallbackFrom=net-5.0

ověření je vhodná pro uzavřené interní sítě, do kterých mají přístup pouze ověření uživatelé.[6]

```
"iisSettings": {
  "windowsAuthentication": true,
  "anonymousAuthentication": false,
},
```

Zdrojový kód 7 - Nastavení konfigurace pro Windows autentizaci

Metoda pro získání osobního čísla načte přihlašovací jméno a následně ho upraví do potřebné podoby a vrátí textový řetězec ve formátu pěti nebo šestimístného čísla.

```
if (accessor.HttpContext.User.Identity.Name != null)
{
   var userName = accessor.HttpContext.User.Identity.Name;
   //Odebrání domény z přihlašovacího jména URAD\\######
   string oscislo = (6 > userName.Length
        ? userName
        : userName.Substring(userName.Length - 6));
   //Uprava osobního čísla z formátu 0##### na #####
   return oscislo[0] == '0' ? oscislo.Substring(1, 5) : oscislo.Substring(0, 6);
}
return "";
```

Zdrojový kód 8 – Metoda pro načtení a úpravu osobního čísla

Metoda „VyplnDataZDatabaze“ zajišťuje ověření osobního čísla v databázi uživatelů a načtení rolí uživatele. Pro dotaz do databáze je vytvořena metoda „ExistujeUzivatelSOsobnimCislem“ v třídě „UzivatelDao“.

```
public async Task VyplnDataZDatabaze()
{
  if (await new UzivatelDao(_db).ExistujeUzivatelSOsobnimCislem(OsobniCislo))
  {
     ListRoli = await new
        RoleDao(_db).VratSeznamRoliZamestnanceDleOsobnihoCisla(OsobniCislo);
     Nalezeny = ListRoli.Any();
  }
  else
  {
     Nalezeny = false;
  }
}
```

Zdrojový kód 9 – Metoda pro vyplnění dat uživatele z databáze

---

[6] Scott Addie, Konfigurace ověřování systému Windows v ASP.NET Core, 13.05.2021, https://docs.microsoft.com/cs-cz/aspnet/core/security/authentication/windowsauth?view=aspnetcore-6.0&tabs=visual-studio

```
public async Task<bool> ExistujeUzivatelSOsobnimCislem(string OsobniCislo)
{
    var existuje = await _db.TUzivatels
        .Join(_db.TZamestnancis, uziv => uziv.OsobniCislo,
            zam => zam.OsobniCislo, (uziv, zam) => new { uziv, zam })
        .AnyAsync(w => w.uziv.JePlatny &&
            w.uziv.OsobniCislo == OsobniCislo &&
            w.zam.JeAktivni);
    return existuje;
}
```

Zdrojový kód 10 – Dotaz pro ověření existence uživatele

V případě, že byl uživatel nalezen v databázi, je následně nutné získat seznam jeho rolí. Z tohoto důvodu je metoda pro získání rolí v podmínce. Metoda „VratSeznamRoliZamestnanceDleOsobnihoCisla“ získá z databáze seznam rolí pro platného uživatele a z tohoto seznamu vrátí seznam identifikátorů rolí bez názvu role a popisu role, které nejsou pro tento účel potřebné.

```
public async Task<List<RoleBaseDto>> VratSeznamRoliZamestnanceDleOsobnihoCisla(string
OsobniCislo)
{
    return await _db.TRoleUzivatels.Include(t => t.IdUzivatelNavigation).Where(w =>
            w.IdUzivatelNavigation.OsobniCislo == OsobniCislo &&
            w.IdUzivatelNavigation.JePlatny)
            .Select(role => new RoleBaseDto() { IdRole = role.IdRoleNavigation.Id })
            .Distinct().ToListAsync();
}
```

Zdrojový kód 11 – Dotaz pro načtení seznamu rolí zaměstnance

Pro seznam identifikátorů je vytvořena třída „RoleBaseDto“, která definuje pouze základní vlastnost identifikátor role. Třídy Data transfer object zajišťují konzistenci (definují rozsah) přenášených dat webovému rozhraní. Díky tomu se ke klientovi přenáší jen data, o která je žádáno, navíc je přenášeno méně dat mezi serverem a klientem. Tyto třídy tvoří další vrstvu aplikace.

V rámci aplikace jsou použity role „Spravce“ a „Predstaveny“. Tyto role jsou v rámci aplikace jasně definovány výčtovým typem Enum. Identifikátory rolí musí být zavedeny shodně v aplikaci a také v databázi. Podle načtených identifikátorů rolí uživatele je následně rozhodnuto, jaká data jsou uživateli poskytnuta. Z toho důvodu

jsou role definovány na jednom místě a následně se v rámci celé aplikace lze odkazovat na tyto výčtové typy. [7]

```
public enum Role
{
    Spravce = 1,
    Predstaveny = 2
}
```

Zdrojový kód 12 – Seznam rolí

	Id	Nazev	Popis
1	1	Správce	Zajišťuje správu aplikace
2	2	Představený	Vlkádá profesní životopis

Obrázek 6 – Seznam rolí

Pro ověření, zda mezi seznamem rolí uživatele je patřičná role, jsou zavedeny ve třídě „Uzivatel“ metody „MaRole“, které požadují parametr typu enum „Role“. Následná logika metody ověřuje, jestli je daná role nebo vice rolí obsaženo v seznamu rolí uživatele.

```
public bool MaRole(Role hledany)
{
   return ListRoli.Any(x => x.IdRole == (int)hledany);
}
public bool MaRole(Role[] seznam)
{
   var prunik = Array.ConvertAll(seznam, value =>
                        (int)value).Intersect(ListRoli.Select(x => x.IdRole));
   return prunik.Any();
}
```

Zdrojový kód 13 – Metody pro ověření hledané role

Pro zajištění konzistence všech požadavků na rozhraní jsou zavedeny pomocné třídy „Kontrola“ a „DbAuthorize“. Tyto třídy obsahují sadu kontrol, které je nutné provést vždy před poskytnutím dat uživateli. Metody jsou napojeny v posloupnosti tak, aby bylo možné jejich kombinací zajistit všechny typy kontrol.

Nejzákladnější kontrola, která může proběhnout, je kontrola, jestli je uživatel nalezen mezi zaměstnanci a má přidělenou alespoň jednu roli.

---

[7] Bill Wagner, Výčtové typy (Referenční dokumentace jazyka C#), 4.3.2022, https://docs.microsoft.com/cs-cz/dotnet/csharp/language-reference/builtin-types/enum

```
private static async Task<(bool ok, string chyba, Uzivatel prihlaseny)>
                UzivatelNalezen(IHttpContextAccessor accessor, DbE db)
{
    Uzivatel prihlaseny = new Uzivatel(accessor, db);
    await prihlaseny.VyplnDataZDatabaze();
    if (!prihlaseny.Nalezeny)
    {
        return (false, "Není nalezen uživatel v DB", prihlaseny);
    }
    return (true, "", prihlaseny);
}
```

Zdrojový kód 14 – Kontrola nalezení uživatele

Metoda zavede novou instanci třídy „Uzivatel“, která získá osobní číslo uživatele a následně je zadán dotaz na ověření uživatele v databázi metodou „VyplnDataZDatabaze“. Pokud je uživatel nalezen, metoda vrátí informaci o tom, že kontrola proběhla v pořádku a instanci nalezeného uživatele. V případě, že uživatel nebyl nalezen, vrátí metoda chybový stav s popisem chyby. Nadřazená kontrola nejdříve zkontroluje, jestli je uživatel nalezen a pokud je vše v pořádku, zkontroluje, jestli má uživatel patřičnou roli. Tato metoda je vytvořena shodně i pro seznam rolí.

```
public static async Task<(bool ok, string chyba, Uzivatel prihlaseny)>
                UzivatelNalezenRole(IHttpContextAccessor accessor, DbE db, Role role)
{
    var uzivatel = await UzivatelNalezen(accessor, db);
    if (!uzivatel.ok)
    {
        return (false, uzivatel.chyba, uzivatel.prihlaseny);
    }
    if (!uzivatel.prihlaseny.MaRole(role))
    {
        return (false, "Uzivatel nemá patřičné oprávnění", uzivatel.prihlaseny);
    }

    return (true, "", uzivatel.prihlaseny);
}
```

Zdrojový kód 15 – Kontrola nalezení uživatele s rolí

Pokud se jedná o vlastní záznam, je nutné, aby přihlášený uživatel měl stejné osobní číslo, jako má vlastník záznamu. Metoda zkontroluje, jestli má uživatel patřičnou roli a jestli se osobní čísla shodují.

```
public static async Task<(bool ok, string chyba, Uzivatel prihlaseny)>
                UzivatelNalezenRoleDleOsobnihoCisla(IHttpContextAccessor accessor, DbE db,
                        Role role, string osobniCislo)
{
    var uzivatel = await UzivatelNalezenRole(accessor, db, role);
    if (!uzivatel.ok)
    {
        return (false, uzivatel.chyba, uzivatel.prihlaseny);
    }
    if (uzivatel.prihlaseny.OsobniCislo != osobniCislo)
    {
        return (false, "Osobní číslo uživatele se neshoduje", uzivatel.prihlaseny);
    }
    return (true, "", uzivatel.prihlaseny);
}
```

Zdrojový kód 16 – Kontrola nalezení uživatele s rolí a osobním číslem

Pokud uživatel požaduje konkrétní záznam, proběhne kontrola, jestli je zadáno identifikační číslo, následně je provedena kontrola, jestli požadovaný záznam existuje a jestli má uživatel patřičné oprávnění pro tento záznam.

```
internal static async Task<(bool ok, string chyba, Uzivatel prihlaseny)>
                        UzivatelNalezenRoleId(IHttpContextAccessor accessor,
                                DbE db, int id, string Nazev, Role[] role)
{
    var uzivatel = await UzivatelNalezenRole(accessor, db, role);
    if (!uzivatel.ok)
    {
        return (false, uzivatel.chyba, uzivatel.prihlaseny);
    }
    if (id == 0)
    {
        return (false, "Identifikace " + Nazev + " nezadána",
                         uzivatel.prihlaseny);
    }
    return (true, "", uzivatel.prihlaseny);
}
```

Zdrojový kód 17 – Kontrola nalezení uživatele s rolí a identifikátorem záznamu

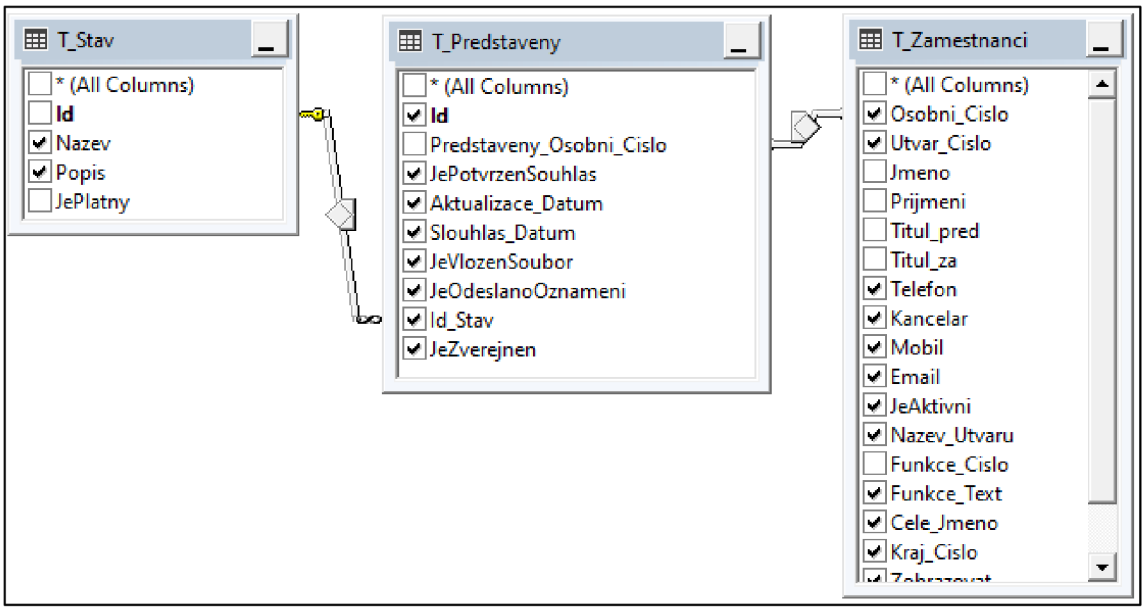
```
private static async Task<(bool ok, T table, string chyba, Uzivatel prihlaseny)>
                VnitniKontrola((bool ok, string chyba, Uzivatel prihlaseny) kontrola, DbE db,
                               int id, string nazev)
{
    if (kontrola.ok)
    {
        T table = null;
        table = await new DaoBase<T>(db).GetById(id);
        if (table != null)
        {
            if (table.MaOpravneniEditovat(kontrola.prihlaseny))
            {
                return (true, table, "ok", kontrola.prihlaseny);
            }
            return (false, null, "Uživatel nemá oprávnění upravovat záznam",
                        kontrola.prihlaseny);
        }
        return (false, null, "Záznam nenalezen", kontrola.prihlaseny);
    }

    return (false, null, "Záznam " + nazev + " nenalezen", kontrola.prihlaseny);
}
```

Zdrojový kód 18 – Kontrola existence záznamu a oprávnění uživatele pro daný záznam

Hlavní funkcí této části aplikace je poskytovat pro webové rozhraní veškerá data, která je nutné získat z databáze. Pro každou potřebnou formu zobrazení poskytuje jedinečný výstup. Výstupy z aplikace jsou rozděleny do jednotlivých kontrolérů, aby bylo možné na tyto výstupy přehledně směřovat z webového rozhraní. Pro výstupy, které se týkají představených, je zaveden kontrolér „PredstavenyController“. Pro správu uživatelů je zaveden kontrolér „UzivatelController“. V těchto kontrolérech jsou pak zavedeny akce pro jednotlivé úkony, ve kterých je vždy v první řadě nutné zkontrolovat, jestli přihlášený uživatel má právo na tuto danou akci. Kontrola je provedena ověřením uživatele v databázi a ověřením role. Pokud akce obsahuje nějaké vstupní parametry, je nutné prověřit správnost těchto dat.

Postup získání dat z databáze do webového rozhraní prostřednictvím API je zahájen zadáním požadavku z webového rozhraní na server. Například zasláním požadavku na adresu https://domena/api/Predstaveny/GetListPredstaveny, bude pomocí směřovače požadavek zaveden na kontrolér „PredstavenyController“, ve kterém je následně vyhledána akce s názvem „GetListPredstaveny“. Tato akce poskytuje webovému rozhraní seznam představených, který budou správci aplikace kontrolovat. Tento seznam bude načten do tabulky a z důvodu přehlednosti zde budou načteny jen nejdůležitější data. Pro tuto akci je zavedena kontrola uživatele,

který musí mít přidělenu roli „Spravce“. Pokud je uživatel úspěšně ověřen, je zadán dotaz do databáze. Pro tento dotaz je zavedena metoda „VratSeznamPredstavenych“ ve třídě „PredstavenyDao“, která vrací seznam představených. Pro tento dotaz byl vytvořen v databázi pohled, který spojuje tabulky „T_Predstaveny“, „T_Stav“ a „T_Zamestnanci“. Z tabulky „T_Stav je dodán název stavu a z tabulky „T_Zamestnanci“ všechny informace o zaměstnanci.



Obrázek 7 – Náhled na pohled V_Predstaveni

Jako šablona pro seznam představených je zavedena třída „PredstavenyDto“, která obsahuje nejdůležitější data z výstupu databáze. Tento proces je podobný u všech ostatních akcí.

```
[HttpGet]
[Route("GetListPredstaveny")]
public async Task<ActionResult<IEnumerable<PredstavenyDto>>> GetListPredstaveny()
{
    var kontrola = await Kontrola.UzivatelNalezenRole(_accessor, _db, Role.Spravce);
    if (!kontrola.ok)
    {
        return BadRequest(new { kontrola.chyba });
    }
    var predstaveni = await new PredstavenyDao(_db).VratSeznamPredstavenych();
    return Ok(predstaveni);
}
```

Zdrojový kód 19 – Výstupní bod pro seznam představených

Dále je potřeba vytvořit výstup pro zobrazení podrobnějších informací o jednom záznamu představeného na stránce detailu. Tento výstup bude sloužit pro

představeného, který bude nahlížet na vlastní záznam a dále pro správce, který bude na detail záznamu přistupovat ze seznamu představených. Pro každý typ oprávnění bude proces pro získání záznamů odlišný. Z toho důvodu jsou zavedeny dva výstupní body. V obou případech je nejdříve provedena kontrola, jestli je uživatel nalezen. V případě, že je volán výstupní bod pro vlastní záznam, musí uživatel mít roli „Predstaveny“ a následně je záznam vyhledán dle osobního čísla představeného. Pokud záznam požaduje správce, je provedena kontrola, jestli uživatel má roli „Spravce“ a jestli je vyplněn parametr identifikátoru, dle kterého je záznam vyhledán. Pokud tedy kontrola proběhne v pořádku, je zadán požadavek do databáze pro načtení požadovaných dat. Společně s informacemi záznamu je načtena i informace o stavu, ve kterém se záznam nachází. V případě, že existují ještě další data, např. seznam souborů nebo seznam notifikací, načtou se tyto seznamy společně s informacemi o představeném. Tyto data jsou následně předána zpět do webového rozhraní.

```
[Route("GetPredstaveny/{id}")]
[HttpGet("{id}")]
public async Task<ActionResult<PredstavenyDetailDto>> GetUzivatel(int id)
{
    var kontrola = await Kontrola.UzivatelNalezenRoleId(_accessor, _db, id, "představeného",
                                                        new[] { Role.Spravce });
    if (!kontrola.ok)
    {
        return BadRequest(new { kontrola.chyba });
    }
    var predstaveny = new PredstavenyDao(_db).VratPredstavenehoDleId(id);
    return Ok(predstaveny);
}
```

Zdrojový kód 20 – Výstupní bod pro jeden záznam představeného

Pro vyplnění profesního životopisu a předání záznamu ke kontrole správci aplikace je zaveden výstupní bod, který požaduje vstupní parametr identifikátor představeného a objekt se zaslanými informacemi z formuláře webového klienta. Následně proběhne kontrola role a zároveň se zkontroluje oprávněnost tohoto kroku. Dále systém uloží přiložený soubor do souborového systému. Předá záznam ke kontrole a odešle notifikaci správcům aplikace.

V případě, že správce schválí záznam představeného, další výchozí bod provede kontrolu oprávnění a záznam nastaví do stavu zveřejněno.

Pokud je nutné přiložený soubor opravit, je dalším výchozím bodem záznam vrácen představenému nastavením stavu „K opravě“, soubor je nastaven jako neaktivní a představenému je odeslána notifikace, že je nutné profesní životopis upravit.

Jako další sada výchozích bodů je sada pro uživatele. Výchozí bod pro zobrazení seznamu správců zkontroluje roli „Spravce“ přihlášeného uživatele a vrátí seznam správců.

Poté je nutné získat seznam stránek, na které má uživatel oprávnění. Výchozí bod zkontroluje, jaké role má uživatel a dle seznamu rolí přidělí seznam stránek, na které má uživatel oprávnění. Pokud má uživatel roli „Predstaveny“, získá přístup na stránku s vlastními záznamy. Pokud má uživatel roli „Spravce“, získá přístup na stránky seznam představených a seznam správců. Pokud má uživatel alespoň jednu z uvedených rolí, získá přístup na stránku s pokyny.

3.2.2 Uživatelské rozhraní

Uživatelské rozhraní je důležitou součástí každé aplikace. Slouží k propojení mezi uživatelem a aplikací. Je to prostředník, díky kterému může uživatel pracovat s daty, vkládat nová data a mít přehled o tom, co se v aplikaci děje. Je důležité, aby bylo uživatelské rozhraní přehledné a intuitivní. V případě této aplikace je pro vývoj uživatelského rozhraní použita knihovna React, která využívá skriptovací jazyk JavaScript. Je to aplikace s dynamickým obsahem, což znamená, že se obsah mění v závislosti na akcích a vstupech uživatele. To umožňuje uživateli mít přístup k aktuálním datům a informacím bez nutnosti stálého obnovování stránky.

Tato webová aplikace obsahuje několik stránek, z nichž každá má svou specifickou roli. Jednou z nich je stránka určená k zobrazení návodu. Je určena pro uživatele, kteří chtějí vložit nový záznam do systému a potřebují instrukce, jak toho dosáhnout.

Další stránka zobrazuje uživateli podrobné informace o jeho vlastním záznamu. V této stránce najde uživatel přehled všech vlastních údajů, včetně vložených profesních životopisů. Kromě toho zde může vidět seznam notifikací, které mu byly odeslány a informaci o stavu jeho záznamu. Tato stránka tedy

poskytuje uživateli komplexní přehled o všech údajích spojených s jeho osobou v rámci aplikace.

V aplikaci je také stránka, která zobrazí seznam představených v přehledné tabulce. V této tabulce jsou základní informace o představeném a dále tlačítko, díky kterému je možné zobrazit detail představeného. Detail představeného je načten ve stejné formě jako stránka pro vlastní záznam představeného. Stránka je přístupná pouze pro uživatele s oprávněním správce, kteří mohou pomocí této stránky spravovat záznamy představených.

V neposlední řadě je součástí stránka, pomocí které lze zobrazit seznam správců. Stránka obsahuje tabulku, kde jsou zobrazeny záznamy o správcích. Stránka je dostupná pouze pro uživatele s oprávněním správce.

Základní struktura React aplikace zahrnuje hlavní soubory „index.html“, index.tsx a app.tsx. Soubor index.html obsahuje základní HTML kostru aplikace a v těle dokumentu se nachází element s identifikátorem "root".

```
<body>
  <noscript>Pro spuštění aplikace je nutné povolit javascript.</noscript>
  <div id="root"></div>
</body>
```

Zdrojový kód 21 – Náhled na obsah souboru index.html

Soubor „index.tsx“ obsahuje funkci „render“ z knihovny „ReactDom“, která zajišťuje vykreslení komponent do souboru „index.html“. Tato funkce přijímá parametr vstupní komponenty a parametr elementu, do kterého má být vstupní komponenta vykreslena.

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root'),
);
```

Zdrojový kód 22 – Náhled na obsah souboru index.tsx

Jako hlavní komponenta je použita „App.tsx“, která slouží jako výchozí bod aplikace. V této komponentě je zavedeno směřování aplikace, jednotlivé stránky a hlavní obal všech stránek „MasterPage.tsx“.

Jako hlavní komponenta je použita „BrowserRoute“ z knihovny „React-Router-Dom“, pro zajištění a nastavení směřování. Díky ní je možné nadefinovat strukturu aplikace tím, že pro jednotlivé komponenty jsou přiřazeny URL adresy, na kterých se budou komponenty následně zobrazovat, tzn. knihovna bude na komponenty směřovat. To usnadňuje uživatelům navigaci po aplikaci pomocí odkazů a adresního řádku prohlížeče, aniž by bylo nutné načítat celou stránku. Knihovna navíc poskytuje funkce pro programové přesměrování a další funkce pro správu historie aplikace. [8]

```
<BrowserRouter>
  <Routes>
    <Route path='/' element={<Masterpage />}>
      <Route index element={<UvodniStranka />} />
      <Route path='uzivatel'>
        <Route index element={<SeznamUzivatel />} />
        <Route path=':idDetailUzivatel' element={<DetailUzivatel />} />
      </Route>
      <Route path='predstaveny'>
        <Route index element={<SeznamPredstaveny />} />
        <Route
          path=':idDetailPredstaveny'
          element={<DetailPredstaveny />}
        />
      </Route>
      <Route path='vlastni-zaznam' element={<VlastniZaznam />} />

      <Route
        path='*'
        element={<ChybovaStranka text='Stránka nenalezena' barva='error' />}
      />
    </Route>
  </Routes>
</BrowserRouter>
```

Zdrojový kód 23 – Nastavení routování v souboru App.tsx

Obalovací komponenta „MasterPage.tsx“ obsahuje poskytovatele globálních funkcí, které jsou využívány napříč aplikací. Patří mezi ně „ThemeProvider“ pro nastavení jednotného grafického vzhledu, „SnackbarProvider“ pro zobrazení notifikačních zpráv, „HlavniNabidka“ pro navigaci v aplikaci a komponenta „Outlet“, do které router nasměřuje obsah stránek.

[8] Introduction. React Router [online]. remix: remix, 2022 [cit. 2022-03-31]. Dostupné z: https://reactrouter.com/docs/en/v6/getting-started/tutorial

```
<ThemeProvider theme={theme}>
  <SnackbarProvider ref={notistackRef} maxSnack={3} action={Action}>
    <CssBaseline />
    <HlavniNabidka />
    <Outlet />
  </SnackbarProvider>
</ThemeProvider>
```

Zdrojový kód 24 – Obsah komponenty MasterPage.tsx

Pro spojení aplikace s API je zavedena funkce „getUrl“, která umožňuje snadné a efektivní spojení s různými koncovými body API. Tato funkce přijímá jako parametr řetězec výchozího bodu, na který má být zaslán požadavek. Tato funkce pak tento řetězec obohatí prefixem URL adresy, na které se API nachází. Tento prefix se dynamicky mění dle prostředí, ve kterém je aplikace spuštěna.

```
export const getUrl = (page: string) => {
  return process.env.NODE_ENV === 'production'
    ? 'http://ds7000ap4701.fs.mfcr.cz/sykoc-statistiky-vyvoj/api/' + page
    : 'https://localhost:44341/api/' + page;
};
```

Zdrojový kód 25 – Náhled na metodu getUrl

Pro načítání dat z API do aplikace (respektive práci s http požadavky) je použita knihovna „Axios“. Tato knihovna zajišťuje posílání požadavků na API a serverové aplikace. Poskytuje potřebné metody pro odesílání požadavků. Metody typu GET pro získání dat, POST a PUT pro odeslání dat a DELETE pro smazání dat. Axios také umožňuje nastavit hlavičky požadavků, což zajišťuje nastavení a úpravu požadavků. Knihovna automaticky transformuje data v odpovědi na požadavek do JSON formátu, což usnadňuje práci s daty v aplikaci. [9]

```
const dataUzivatelResp = await axios.get<IUzivatel>(
  getUrl(`uzivatel/GetUzivatel/${paramsRoute.idDetailUzivatel}`),
  {
    headers: hlavicka,
  },
);
setDataUzivatel(dataUzivatelResp.data);
```

Zdrojový kód 26 – Metoda pro načtení dat z koncového bodu API

[9] Axios: Promise based HTTP client for the browser and node.js [online]. [cit. 2023-01-18]. Dostupné z: https://axios-http.com/

Pro přehledné zobrazení seznamů v aplikaci je zavedena knihovna „AgGrid“. Tato knihovna slouží ke správě dat a poskytuje vysokou úroveň funkcionality, včetně možnosti filtrování, řazení, rozdělení na stránky, animací, řazení sloupců i řádků. Knihovna může pracovat s velkým množstvím dat a poskytuje rychlý a efektivní výkon. Umožňuje snadnou integraci s ostatními knihovnami pro React.[10]

Po spuštění aplikace bude na úvodní obrazovce načtena stránka „UvodniStranka“, která bude obsahovat pokyny, jak správně vyplnit a vložit profesní životopis do aplikace. „Route“ směřovače pro úvodní stránku obsahuje parametr „index“. Tento parametr zajišťuje, aby byla „UvodniStranka“ zobrazena vždy v základním tvaru URL, respektive po prvním načtení aplikace. Dále je zavedena „Route“ pro zobrazení formuláře, díky kterému je možné vložit profesní životopis, „Route“ pro zobrazení seznamu představených a seznamu uživatelů. Pokud se zadaná cesta URL nebude shodovat s žádným tvarem parametrů „path“ jednotlivých „Route“, aplikace zvolí poslední cestu „path='*'“ a vykreslí chybovou stránku.

Na stránce pro zobrazení vlastního záznamu, stejně jako na stránce pro zobrazení detailu představeného, je zavedena komponenta „Detail“. Tato komponenta přijímá dva parametry. Prvním z těchto parametrů lze nastavit koncový bod API, kde bude ověřeno oprávnění uživatele a případně budou načítána data. Jako druhý parametr je předána informace, jestli se komponenta má nastavit pro potřeby vlastníka záznamu nebo z pohledu správce. Při načítání komponenty „Detail“ je spuštěna funkce, která pomocí knihovny „Axios“ načte do komponenty data z koncového bodu API. Pro načtení vlastního záznamu představeného jsou data načítány z koncového bodu „predstaveny/GetVlastniZaznam“, pro načtení detailu záznamu pro správce je záznam načten z koncového bodu „predstaveny/GetPredstaveny/id“, kde „id“ představuje proměnou s identifikátorem záznamu. Automatické spuštění načítání zajišťuje funkce „useEffect“, která je součástí knihovny React, a pokud je zavedena, spouští se vždy

[10] Ag-grid [online]. AG Grid, 2022 [cit. 2023-02-19]. Dostupné z: https://ag-grid.com/javascript-data-grid/grid-features/

při prvním načtení dané komponenty. Této funkci lze ještě nastavit parametr pro další spuštění.[11]

Jako další funkcionalita v komponentě „Detail“ je možnost vložení profesního životopisu. Po kliknutí na tlačítko určené pro vložení životopisu se zobrazí okno pro vložení souboru a pole pro potvrzení souhlasu o správnosti a aktuálnosti dat. Potvrzením dojde k předání do stavu „ke kontrole“. Odesláním dat na API je požadavek zpracován (uloží záznam do databáze a soubor do souborového systému). Správce aplikace pak musí zkontrolovat záznam (postoupit zveřejněním nebo vrácením k opravě kliknutím na požadované tlačítko).

Další funkcionalita je odeslání notifikace, která je zpřístupněna jen pro správce. Notifikace odesílá upozornění představenému, že je třeba vložit nový životopis.

Dalším typem stránek jsou seznamy. Tyto stránky zajišťují načtení dat do přehledné tabulky „AgGrid“. Stránky „SeznamPredstaveny“ a „SeznamUzivatel“ obsahují funkci pro načtení dat pomocí „Axios“ z koncových bodů API. Dále obsahují objekt, který definuje nastavení jednotlivých sloupců tabulky. Načtená data a objekt definující sloupce, jsou pak společně předány do komponenty „AgGrid“. Jedním z definovaných sloupců pro seznam představených je komponenta, která vytváří tlačítko pro zobrazení detailu záznamu.

3.3 Modul pro zobrazení profesních životopisů

Modul pro zobrazení profesních životopisů slouží k zobrazení životopisů na webovém portálu úřadu. Na tomto portálu je zavedena sekce „Představení a vedoucí zaměstnanci“, ve které je seznam odkazující na stránky jednotlivých krajů. Na stránce každého kraje je dále seznam jednotlivých úřadů (krajský úřad a všechny jeho podřízená pracoviště). Při zvolení konkrétního úřadu se zobrazí seznam představených na daném úřadu.

---

[11] React: Using the Effect Hook [online]. Meta Platforms, 2023 [cit. 2023-02-19]. Dostupné z: https://reactjs.org/docs/hooks-effect.html

Sekce „Představení a vedoucí zaměstnanci“ a seznam stránek pro jednotlivé kraje jsou vytvořeny správcem webového portálu úřadu. Pro tyto stránky je použita šablona, která má funkci pro zobrazení obsahu jiné webové stránky na základě zadané URL s parametrem označujícím daný kraj. Modul pro zobrazení zajišťuje načítání dat z databáze životopisů pro daný kraj a generuje seznam do html souboru. Tento soubor má takovou strukturu html, která odpovídá vzhledu daného webového portálu.

Pro zveřejnění dat na webovém portálu úřadu jsou potřebná jen některá data. Konkrétně se jedná o tyto údaje představeného:

- jméno,
- příjmení,
- tituly,
- název pozice,
- e-mail,
- telefonní číslo.

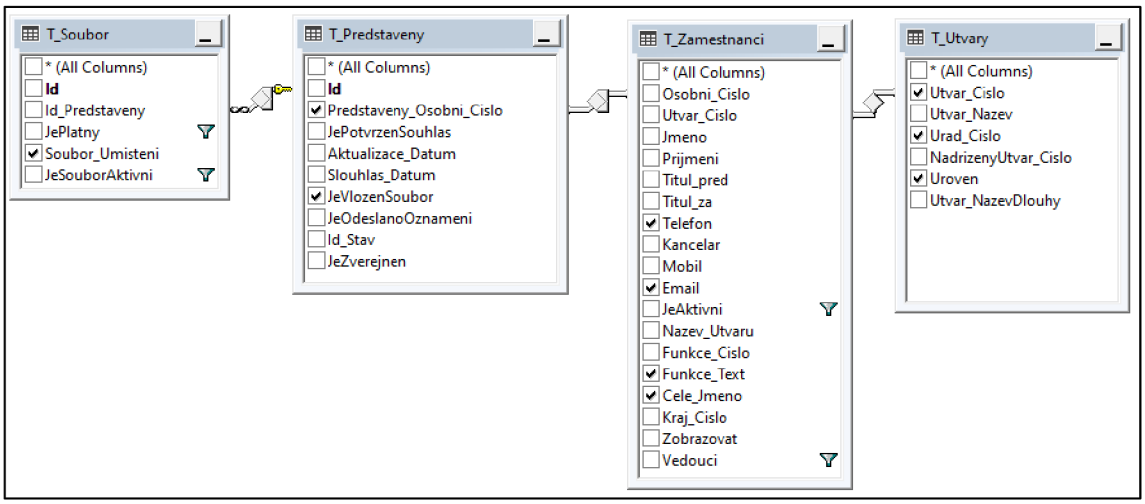
Dále je nutné načíst další technická data, která aplikaci pomáhají ke správnému zařazení představeného. Jedná se o:

- číslo kraje,
- číslo úřadu,
- číslo útvaru – podle něj lze v dané organizační struktuře řadit jednotlivé pracovníky v seznamu daného úřadu od nejvýše zařazených útvarů,
- úroveň zařazení pozice – pro rozlišení, zda se jedná o ředitele nebo vedoucího pracovníka (dle obsahu Akčního plánu je nutné zveřejnit profesní životopis pouze od pozice ředitele odboru a výše),
- vložení souboru – je potřeba mít informaci, zda byl u daného pracovníka soubor vložen či nikoliv,
- osobní číslo – určuje název složky souborového systému, ve které má pracovník vložené své profesní životopisy,
- název souboru – určuje, jaký soubor se ze složky představeného má otevřít.

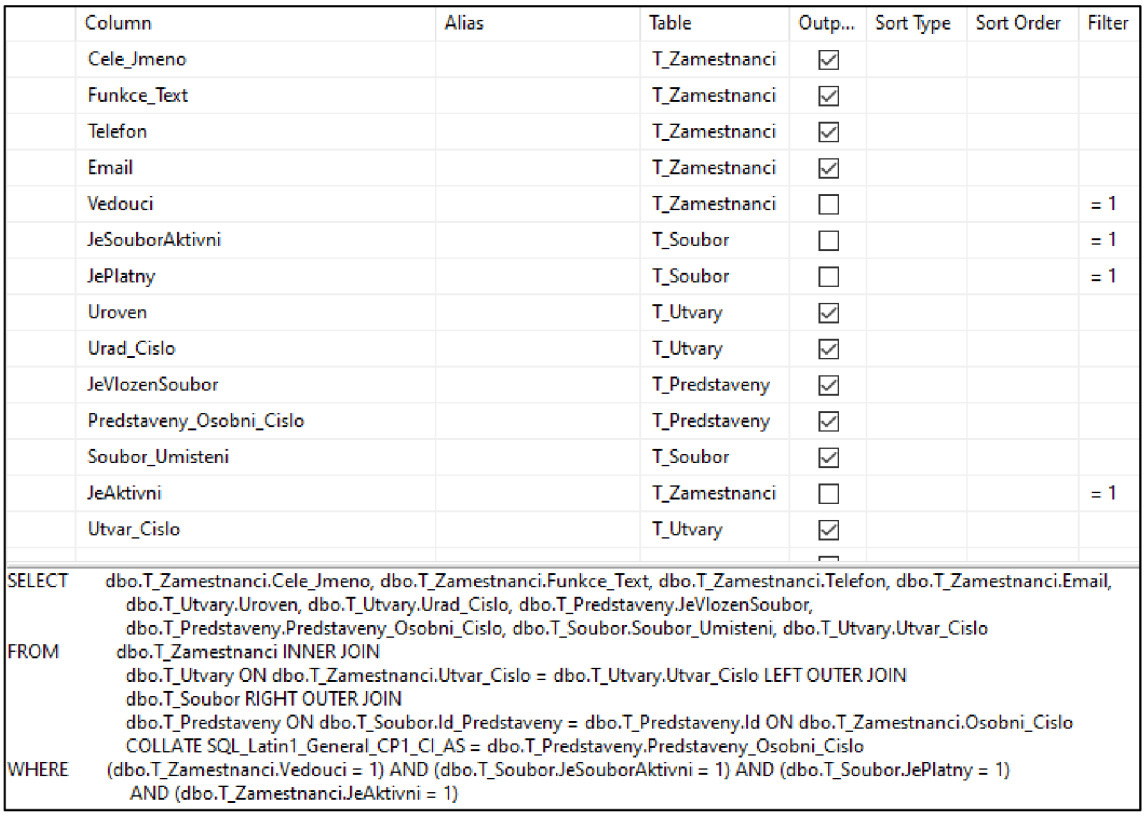
Dále jsou potřeba některé bližší údaje o úřadu:

- číslo úřadu,
- číslo nadřízeného úřadu,
- název úřadu,
- určení typu úřadu – zda se jedná o krajský úřad (určuje, jestli má být daný úřad po otevření stránky rozbalen).

Z toho důvodu byly vytvořeny pohledy „V_UredniciWeb“ a „V_UradyWeb“ v databázi. Pohled „V_UredniciWeb“ načítá přes pole osobního čísla z tabulky „T_Predstaveny“ záznam z tabulky „T_Zamestnanci“ a informace o souboru úředníka z tabulky „T_Soubor“.



Obrázek 8 – Náhled na diagram pohledu V_UredniciWeb

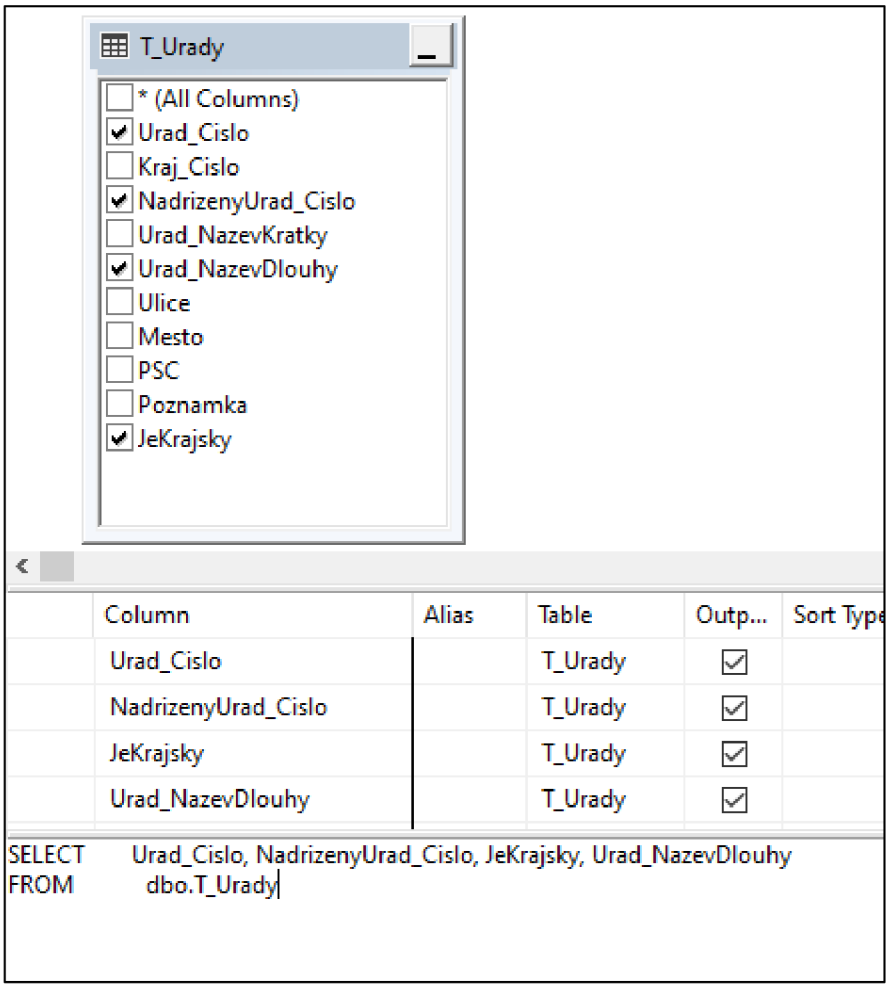


	Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter
	Cele_Jmeno		T_Zamestnanci	☑			
	Funkce_Text		T_Zamestnanci	☑			
	Telefon		T_Zamestnanci	☑			
	Email		T_Zamestnanci	☑			
	Vedouci		T_Zamestnanci	☐			= 1
	JeSouborAktivni		T_Soubor	☐			= 1
	JePlatny		T_Soubor	☐			= 1
	Uroven		T_Utvary	☑			
	Urad_Cislo		T_Utvary	☑			
	JeVlozenSoubor		T_Predstaveny	☑			
	Predstaveny_Osobni_Cislo		T_Predstaveny	☑			
	Soubor_Umisteni		T_Soubor	☑			
	JeAktivni		T_Zamestnanci	☐			= 1
	Utvar_Cislo		T_Utvary	☑			

```
SELECT      dbo.T_Zamestnanci.Cele_Jmeno, dbo.T_Zamestnanci.Funkce_Text, dbo.T_Zamestnanci.Telefon, dbo.T_Zamestnanci.Email,
                dbo.T_Utvary.Uroven, dbo.T_Utvary.Urad_Cislo, dbo.T_Predstaveny.JeVlozenSoubor,
                dbo.T_Predstaveny.Predstaveny_Osobni_Cislo, dbo.T_Soubor.Soubor_Umisteni, dbo.T_Utvary.Utvar_Cislo
FROM        dbo.T_Zamestnanci INNER JOIN
                dbo.T_Utvary ON dbo.T_Zamestnanci.Utvar_Cislo = dbo.T_Utvary.Utvar_Cislo LEFT OUTER JOIN
                dbo.T_Soubor RIGHT OUTER JOIN
                dbo.T_Predstaveny ON dbo.T_Soubor.Id_Predstaveny = dbo.T_Predstaveny.Id ON dbo.T_Zamestnanci.Osobni_Cislo
                COLLATE SQL_Latin1_General_CP1_CI_AS = dbo.T_Predstaveny.Predstaveny_Osobni_Cislo
WHERE       (dbo.T_Zamestnanci.Vedouci = 1) AND (dbo.T_Soubor.JeSouborAktivni = 1) AND (dbo.T_Soubor.JePlatny = 1)
                AND (dbo.T_Zamestnanci.JeAktivni = 1)
```

Obrázek 9 – Náhled dotaz pro pohled V_UredniciWeb

Pohled „V_UradyWeb“ je vytvořen z tabulky „T_Urady“ proto, aby pro veřejnou část nebyly dostupné nepotřebné informace.



Obrázek 10 – Náhled na pohled V_UradyWeb

Pro vytvoření modulu je zvolena šablona projektu ASP.NET framework 4.7. Do projektu aplikace je zaveden Entity Framework, který zajišťuje objektově relační mapování. Pro napojení aplikace na databázi je do konfiguračního souboru přidán připojovací řetězec s IP adresou databázového serveru, názvem databáze, uživatelským jménem a heslem. Do databázového kontextu je zaveden připojovací řetězec „DbModel“ a jednotlivé třídy „Urad“ a „Predstaveny“, které jsou namapovány na pohledy „V_UradyWeb“ a „V_PredstaveniWeb“. Třídy obsahují stejné vlastnosti jako sloupce databázových pohledů.[12]

[12] Práce s DbContext, Microsoft Documents, 30.11.2021, https://docs.microsoft.com/cs-cz/ef/ef6/fundamentals/working-with-dbcontext

```
public class DbModel : DbContext
{
    public DbModel()
        : base("name=DbModel")
    {
    }

    public virtual DbSet<Urad> Urad { get; set; }
    public virtual DbSet<Predstaveny> Predstaveny { get; set; }
}
```

Zdrojový kód 27 – Náhled na DbModel

Ve třídě „Urad“ je navíc vlastnost „NazevUradu“, která pomocí metody „KontrolaPrazdneHodnoty“ ověří hodnotu z databázového sloupce „Urad_NazevDlouhy“. Vlastnost „NazevUradu“ je využita v pohledu pro webový portál úřadu.

```
[Table("V_UradyWeb")]
public class Urad
{
    [Key]
    public int Urad_Cislo { get; set; }
    public int NadrizenyUrad_Cislo { get; set; }
    public string Urad_NazevDlouhy { get; set; }
    public bool JeKrajsky { get; set; }

    public ICollection<Predstaveny> Predstaveny { get; set; }

    [NotMapped]
    public string NazevUradu => Urad_NazevDlouhy.KontrolaPrazdneHodnoty();
}
```

Zdrojový kód 28 – Třída Urad namapovaná na pohled V_UradyWeb

Podobným způsobem je vytvořena i třída „Predstaveny“, která obsahuje navíc kontrolu, jestli je daný představený ředitelem. Sloupec „urovne“ přebíraný z organizační struktury nese pro každé oddělení informaci o zařazení, kde nejnižší úroveň je rovna hodnotě „00001“ a nejvyšší hodnota je „11111“. Jak už bylo zmíněno, tato hodnota bude následně rozhodovat, jestli bude pro daného pracovníka zobrazen profesní životopis na webovém portálu úřadu. Dalším polem, které je nutné před použitím upravit, je pole „Telefon“.

```
[NotMapped]
public bool JeReditel => Uroven.KontrolaPrazdneHodnoty() != "00011";
[NotMapped]
public string Telefon =>
TelefonDb.KontrolaPrazdneHodnoty().TelefonniCisloSPredvolbou(Utvar_Cislo);
```

Zdrojový kód 29 – Náhled na funkce třídy „Predstaveny“

Pole telefonního čísla je prověřeno metodou „TelefonniCisloSPredvolbou“, která zajišťuje, že bude výstup mít vždy stejnou hodnotu.

Ve zvláštním případě se může stát, že v databázi v tabulce úřadů mohou být telefonní čísla vedená pouze jako čtyřmístná, s ohledem na používané interní linky. V takové situaci se doplní prefix, aby byla čísla jednotná. Případně je nutné odebrat předvolbu „+420“ nebo doplnit oddělovací mezery.

```
public static string TelefonniCisloSPredvolbou(this string telefon, int cislo)
{
    if (!(telefon is null))
    {
        if (cislo.ToString().StartsWith("7") && telefon.Length == 4)
        {
            telefon = "12345" + telefon;
        }
        if (telefon.StartsWith("+420"))
        {
            telefon = telefon.Substring(4);
        }
        if (telefon.Length == 9)
        {
            telefon = telefon.Substring(0, 3) + " " + telefon.Substring(3, 3) + " " +
                        telefon.Substring(6, 3);
        }
        return telefon;
    }

    return "";
}
```

Zdrojový kód 30 – Metoda pro doplnění telefonního čísla

Pro obě zmíněné třídy je nutné zavést kontrolní funkci, která pro zobrazení na web zajišťuje, že pokud z databáze bude načten sloupec s hodnotou „NULL“ nebo jiná prázdná hodnota, nastaví se vlastnost dané instance na hodnotu prázdného řetězce, který nezpůsobí neočekávaný pád aplikace. Tato funkce je použita na všechny vlastnosti, které se používají ve výpisu.

```
public static string KontrolaPrazdneHodnoty(this string s)
{
    return !s.IsEmpty() || !(s is null) ? s : "";
}
```

Zdrojový kód 31 – Metoda pro kontrolu prázdné hodnoty

Pro připojení k databázi je zavedena třída „DaoBase“, která vytváří novou instanci „DbModelu“ v konstruktoru a implementuje rozhraní „Data Access Object“ s obecnými CRUD metodami.

```
public class DaoBase<T> : IDaoBase<T> where T : class
{
    protected DbModel db;

    public DaoBase()
    {
        db = new DbModel();
    }
    public IList<T> GetAll()
    {
        return db.Set<T>().ToList();
    }

    public T GetById(int id)
    {
        return db.Set<T>().Find(id);
    }
}
```

Zdrojový kód 32 – Náhled na třídu DaoBase

Třídy „PredstavenyDao“ a „UradDao“ zajišťují dotazy do databáze. Tyto třídy implementují již zmíněnou třídu „DaoBase“. Výstup z aplikace poskytuje seznam úřadů za daný kraj a každý úřad obsahuje seznam představených za daný úřad. Do třídy „UradDao“ je zavedena metoda, která dle čísla kraje vrátí seznam úřadů odpovídajícího kraje. Pro tento případ je v organizační struktuře sloupec kraj označen jako „NadrizenyUrad_Cislo“.

```
public List<Urad> VratSeznamUraduZaKraj(int kraj)
{
    return db.Urad.Where(urad => urad.NadrizenyUrad_Cislo == kraj).OrderBy(o =>
                                o.Urad_Cislo).ToList();
}
```

Zdrojový kód 33 – Metoda pro načtení úřadů daného kraje

Pro tento modul je použita pouze jedna stránka. Je tedy vytvořen kontrolér „PredstaveniController“, ve kterém je zavedena akce Kraj. Tato akce požaduje jako parametr ID, které bude určovat číslo kraje. Na tento kontrolér se uživatel dostane, pokud bude zvolena URL webové aplikace a následně „/Predstaveni/Kraj/[čísloKraje]“. Obecně se v jednotlivých akcích kontroléru nachází logika pro načtení dat do objektu, který je následně předán do pohledu „View“. Požadavek na tento kontrolér je vytáhnout data z databáze tak, aby v pohledu mohl být vypsán seznam úřadů pro daný kraj a pro každý úřad byl vypsán seznam představených. Jak již bylo zmíněno, organizační struktura je přebírána z personálního systému a databázové tabulky, tedy neobsahují relace. Z tohoto důvodu nelze jednoduše načíst seznam všech úřadů jako jeden list objektů, kde

každý úřad bude obsahovat kolekci představených požadovaného úřadu. V tomto případě do pohledu předáváme seznam úřadů a seznam představených za daný kraj. Načítání tak nebude probíhat jedním dotazem do databáze. Do pohledu tedy nepředáváme jeden objekt, ale proměnnou řazené kolekce členů, kde první člen bude seznam úřadů a druhý člen bude seznam představených.

```
public class PredstaveniController : Controller
{
    public ActionResult Kraj(int id)
    {
        if (id == 0)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }

        var Urady = new UradDao().VratSeznamUraduZaKraj(id);
        var Predstaveni = new PredstavenyDao().VratSeznamPredstavenychZaKraj(id);

        return View((Urady, Predstaveni));
    }
}
```

Zdrojový kód 34 – Kontrolér Predstaveny s akcí Kraj

Vzhledem k tomu, že výstup této aplikace bude začleněn do struktury stránky webového portálu úřadu, je pohled tvořen jako vnořená komponenta. Do pohledu je zaveden "@model" typu proměnná řazené kolekce. [13]

```
@model (
        List<CVproWeb2021.Models.Urad> urady,
        List<CVproWeb2021.Models.Predstaveny> predstaveni
)
```

Zdrojový kód 35 – Náhled na model s proměnou řazené kolekce

V šabloně je zaveden html blok, který bude vložen do obsahu stránky webového portálu úřadu.

V tomto objektu následuje cyklus, který ze seznamu „urady" vypisuje opakovaně blok html kódu. Tento blok na webovém portálu úřadu odpovídá nadpisu úřadu, který lze kliknutím rozbalit a zobrazit tak další podrobné informace.

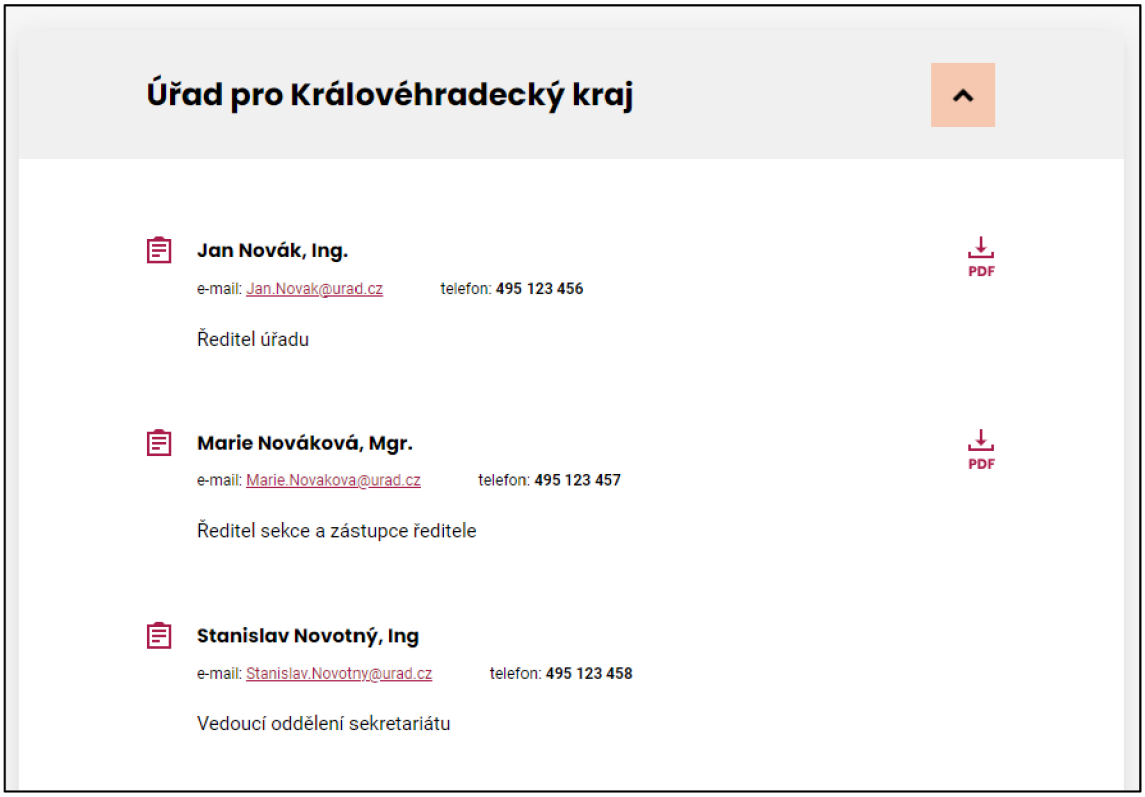
---

[13] Anderson Rick, Přístup k datům modelu z kontroleru, Microsoft Documents, 13.05.2021, https://docs.microsoft.com/cs-cz/aspnet/mvc/overview/older-versions/getting-started-with-aspnet-mvc4/accessing-your-models-data-from-a-controller#strongly-typed-models-and-the-model-keyword

```
<div class="b-accordion-lg u-mb-md">
    @foreach (var urad in Model.urady)
    {
        <div class="b-accordion-lg__item js-accordion__item"
                data-open="@(urad.JeKrajsky ? "true" : "false")">
            <h2 class="b-accordion-lg__title js-accordion-title">
                <button class="b-accordion-lg__toggle"
                        aria-expanded="@(urad.JeKrajsky ? "true" : "false")">
                    <span>
                        <span class="b-accordion-lg__title-text h3">@(urad.NazevUradu)</span>
                    </span>
```

Zdrojový kód 36 – Náhled na blok pro vypsání seznamu úřadů

Pokud se jedná o krajský úřad, je nastaven blok nadpisu takovým způsobem, aby byl po načtení stránky rozbalen. V takovém případě vlastnost „JeUradKrajsky“ nese hodnotu „true“, a tím je do parametrů „data-open“ a „aria-expanded“ nastavena textová hodnota „true“. Z tohoto důvodu je u každého nadpisu ověřeno, jestli se jedná o krajský úřad.



Obrázek 11 – Náhled zobrazení rozbaleného úřadu se seznamem představených

Ostatní úřady v seznamu tohoto kraje mají hodnotu „false“ ve vlastnosti „JeUradKrajsky“ a po načtení stránky na webovém portálu úřadu budou zobrazeny jako sbalené.

Obrázek 12 – Náhled zobrazení sbalených úřadů

Tímto cyklem jsou vypsány všechny úřady z proměnné „urady“. V rámci tohoto cyklu následuje blok, který obsahuje jednotlivé představené pro daný úřad. Jedná se o blok, který je zobrazen po klinutí na nadpis úřadu. U tohoto bloku (stejně jako u nadpisu úřadu) je ověřeno, jestli má být při načtení zobrazený nebo skrytý. Uvnitř tohoto bloku je cyklus, který vypisuje seznam představených pro daný úřad.

```
<div class="b-accordion-lg__content b-box__inner--lg" @(urad.JeKrajsky ? "" : "hidden")>
  @foreach (var predstaveny in Model.predstaveni.Where(w => w.Urad_Cislo == urad.Urad_Cislo))
    {
        <div class="b-teaser u-mb-md u-mb-lg">…</div>
    }
</div>
```

Zdrojový kód 37 – Náhled na blok pro vypsání jednotlivých představených

Tento cyklus vypisuje informace o představeném.

```
<div class="b-teaser__header link-mask">
    <h3 class="b-teaser__title h5 js-matchHeight">
        <div class="link-mask__link v-teaser__link">@(predstaveny.CeleJmeno)</div>
    </h3>
    <p class="b-teaser__meta">
        <span> e-mail: <a href="mailto: @(predstaveny.Email)">@(predstaveny.Email)</a></span>
        <span class="">telefon: <strong>@(predstaveny.Telefon)</strong></span>
    </p>
</div>
```

Zdrojový kód 38 – Náhled na blok pro vypsání informací o představeném

Dále je v bloku tohoto cyklu vložena podmínka, která rozhoduje, jestli bude zobrazen odkaz na PDF soubor. Povinnost zveřejnit profesní životopis se vztahuje pouze na ředitele. Pokud je tedy představený vedoucí pracovník, jsou zobrazeny

pouze jeho údaje, ale jeho profesní životopis není na webovém portálu úřadu zveřejněn.

```
@if (predstaveny.JeVlozenSoubor == true && predstaveny.JeReditel)
{
    <div class="b-teaser__extra">
        <div class="group group--nowrap group--md">
            <p class="btn__wrapper">
                <a
href="https://www.urad.cz/ProfesniZivotopisy/Data/@(predstaveny.OsobniCislo)/@(predstaveny.Na
zevSouboru).pdf"
                class="btn btn--ghost btn--icon-only"
                title="@(predstaveny.CeleJmeno)">
                    <span class="btn__inner">…span>
                </a>
            </p>
        </div>
    </div>
}
```

Zdrojový kód 39 – Náhled na blok pro zobrazení odkazu na soubor

3.4 Služba pro rozesílání notifikací

Modul pro rozesílání notifikací zajišťuje automatickou kontrolu všech zaměstnanců v pravidelných intervalech. Po spuštění služba načte všechny zaměstnance z databáze, kteří mají povinnost vložit profesní životopis a nejsou ještě zavedeni v aplikaci. K tomuto účelu slouží pohled „V_ZamestananciNovy“, který tuto podmínku obsahuje na úrovni databáze.

Služba pro každého zaměstnance z tohoto seznamu musí založit nový záznam v tabulce „T_Uzivatel“. Každému nově vytvořenému uživateli je nastaveno osobní číslo, které je jedinečným identifikátorem zaměstnance. Dále je nastavena informace, že uživatel je platný a následně je do jeho seznamu rolí vložena role Představený. Tím je zajištěno uživatelské oprávnění pro vstup do modulu pro správu profesních životopisů.

Dalším krokem služby je nastavení záznamu představeného. Pro každého zaměstnance ze seznamu je vytvořen záznam. Tomu je nastaveno osobní číslo zaměstnance, ostatní parametry jsou nastaveny do výchozího stavu, tzn. stav čekající na vložení (označován hodnotou 1), záznam nezveřejněn, soubor nevložen, souhlas nepotvrzen, oznámení odesláno. Součástí tohoto kroku je i vytvoření nové notifikace, která je k tomuto záznamu připojena. Tato notifikace oznamuje uživateli, že je nutné vložit profesní životopis.

Služba také zajišťuje kontrolu neaktivních uživatelů. K tomuto účelu je vytvořen pohled „V_ZamestnanciNeaktivni“, který vrátí všechny zaměstnance, kteří mají vytvořen účet v aplikaci a zároveň jsou již neaktivní. Každému z těchto uživatelů je odebrána role „Predstaveny“, účet uživatele je nastaven jako neplatný a dále všechny jeho soubory v záznamu představeného jsou nastaveny jako neplatné, zároveň záznam představeného je nastaven jako nezveřejněn, soubor nevložen, souhlas nepotvrzen a oznámení neodesláno.

Posledním krokem služby je kontrola stavu profesního životopisu. V případě, že životopis nebyl vložen do 14 dnů od zaslání první notifikace, služba zašle další notifikaci. Notifikaci bude opakovaně zasílat v pravidelných intervalech do doby, než dojde k vložení životopisu.

```
internal async Task<List<PredstavenyEmailDto>>
NactiSeznamNotifikaciNevyplnenychPredstavenych()
{
    var datumMax = DateTime.Now.AddDays(-14);
    var seznamNevyplnenych = await _db.T_NotifikacePredstaveny
        .Where(w => w.Id_Notifikace == 1 && w.Odeslano_Datum < datumMax
                    && w.T_Predstaveny.Id_Stav == 1
                    && w.T_Predstaveny.JeOdeslanoOznameni
                    && !w.T_Predstaveny.JeVlozenSoubor)
        .Join(_db.T_Zamestnanci, pred => pred.T_Predstaveny.Predstaveny_Osobni_Cislo,
               zam => zam.Osobni_Cislo, (all, zam) => new PredstavenyEmailDto(){
               Email = zam.Email, Id = all.Id_Predstaveny,
               OsobniCislo = zam.Osobni_Cislo})
        .ToListAsync();
    return seznamNevyplnenych;
}
```

Zdrojový kód 40 – Metoda pro načtení představených, kteří nevložili životopis

4 Použité technologie

Nejrozsáhlejší část aplikace byla tvořena v programovacím jazyce C#, a to především pro své široké spektrum možností a využitelnosti. Jednotlivé moduly zajištují připojení do databáze a dále poskytnutí dat klientské části aplikace. Také zajišťuje autentizaci uživatele a na základě přidělených rolí uděluje oprávnění ke čtení nebo zápisu dat. Aplikace vytvořená v C# zajišťuje serverové části modulů.

Pro konzistentní a bezpečné ukládání dat v aplikaci je použita relační databáze MSSQL.

Javascript, zvláště pak s využitím knihovny React.js, zajišťuje dynamickou interakci mezi aplikací a uživatelem (dynamicky upravuje webové stránky).

React rozšiřuje funkce Javascriptu o možnost vytvářet komponenty, díky kterým je možné zefektivnit a zrychlit vývoj uživatelského rozhraní.

Kombinace těchto programovacích jazyků umožňuje efektivní vytvoření komplexního systému.

4.1 C#

Programovací jazyk C# byl publikován v roce 2002 firmou Microsoft. Původním záměrem bylo využití pouze k programování pro Windows, ale později bylo jeho užití rozšířeno i pro jiné platformy, například Linux, macOS, Android a další.[14]

V minulosti probíhalo programování ve strojovém kódu, ale bylo to velice nepřehledné a namáhavé. Téměř nikdo tímto způsobem dnes již neprogramuje, místo toho se využívají tzv. vyšší programovací jazyky, mezi které patří např. Basic, Ada, C, C++ a C#.

Program, který je tvořený ve vyšším programovacím jazyce, je textový soubor, ve kterém je obsažen popis řešené úlohy, který je vyjádřený pomocí výrazů zapsaných podobně jako v matematice a vybraných anglických slov. Vytvořit program ve vyšším programovacím jazyce je daleko snazší než napsat obdobný program ve strojovém kódu. Stinnou stránkou je fakt, že program ve vyšším programovacím jazyce nejde na počítači spustit přímo, protože počítač mu nerozumí. Proto se tento program musí buď interpretovat nebo přeložit do strojového kódu. V obou případech je třeba mít další program, který toto sám zajistí.

Zápis programu ve vyšším programovacím jazyce, který je obsažen v textovém souboru, se jinak nazývá zdrojový kód nebo také zdrojový program. Ke kompilaci (překlad do strojového kódu) slouží program, kterému se říká překladač nebo také kompilátor.

---

[14] ROBINSON, Simon. C#: programujeme profesionálně. Brno: Computer Press, 2003. Programmer to programmer. ISBN 80-251-0085-5

Kompilací programu vznikne soubor, který obsahuje strojový kód. Ten již lze na cílovém počítači spustit. Mezi programovací jazyky, které jsou typicky kompilované, patří např. Pascal, C nebo C++.

Jazyk C# patří mezi jazyky čistě objektové.

V C# se pracuje pouze s dynamickými objekty (jsou vytvářené pomocí operátoru new). Úklid vytvořených objektových souborů (automatické odstranění nepoužívaných objektů) zajišťuje prostředí .NET, které obsahuje automatickou správu paměti, tzv. garbage collector.

Prostředí .NET je k dispozici pod různými operačními systémy, ale možnosti jeho využití se na různých platformách zatím liší, i když je zřejmá tendence, která má vést ke sjednocení. Aplikace vyvíjené pro Windows mají v současné době nejširší možnosti.

Prostředí .NET verze 4.7 je standardní součástí Windows 10, skládá se z několika částí.

Jednou z nich je knihovna tříd (Framework Class Library). Ta pokrývá např. vstupní a výstupní operace pro třídy sloužící pro ukládání různých druhů dat. S ohledem na to jazyk C# prakticky nepotřebuje své vlastní knihovny.

Nad touto knihovnou jsou ještě další knihovny, např. pro tvorbu databázových aplikací, pro tvorbu grafického uživatelského rozhraní, alternativní nástroje pro tvorbu grafického rozhraní a knihovny pro webové služby. Dále také nástroje pro tvorbu systémů, pro oběh dokumentů, vnořený dotazovací jazyk zvaný LINQ apod. Každá nová verze .NET přináší nové nástroje.

Prostředí .NET Framework umožňuje práci s metadaty. Metadata jsou dodatečné informace, které umožňují získávat z přeloženého programu podrobné informace o třídách, které obsahuje a o jejich metodách. Tomuto procesu se říká reflexe.

V programovacím jazyce C# jsou rozlišeny dvě základní skupiny datových typů. Do první skupiny spadají tzv. hodnotové typy, což jsou kupříkladu znaky, čísla, logické hodnoty a struktury. Do druhé skupiny patří referenční typy, což jsou třídy.[15]

---

[15] VIRIUS, Miroslav. Programování v C#: od základů k profesionálnímu použití. Praha: Grada Publishing, 2021. Knihovna programátora (Grada). ISBN 978-80-271-1216-6

4.2 JavaScript – React

React byl vytvořen vývojáři Facebooku kolem roku 2010, důvodem pro jeho vytvoření byly problémy s údržbou kódu, především při vytváření jednostránkových aplikací, kdy vznikal obrovský problém s udržením systematičnosti a přehlednosti ve struktuře projektu. Čím více bylo v týmu vývojářů, tím exponenciálně rostla nepřehlednost v kódu.

V roce 2010 vývojáři zavedli XHP do PHP. XHP je rozšíření pro PHP, rozšiřuje syntaxi, a tím usnadňuje čtení kódu PHP. Klíčovou vlastností je pojem složených komponent, což umožňuje rozdělit rozhraní do nezávislých, ale snadno složitelných jednotek funkčnosti.

Projekt FaxJs byl projektem vytvořeným v roce 2011, který vycházel z XHP. Mezi klíčové charakteristiky FaxJs patří automatická aktualizace pohledu, kdykoliv se změní jejich stav, tj. reaktivní rozhraní. [16]

React je JavaScriptová knihovna, která má pomoci uživatelům a vývojářům vytvářet ideální prostředí. Když vývojáři tvoří uživatelské rozhraní pro web, desktop, smartphone nebo pro VR, chtějí, aby jejich stránka nebo aplikace zobrazovala různé údaje, které se v průběhu času mění, jako např. informace o uživateli, filtrované seznamy produktu, vizualizace dat nebo podrobnosti o zákaznících. Uživatel bude vybírat filtry a datové sady, vybírat zákazníky k zobrazení, vyplňovat formuláře anebo prozkoumávat prostor VR. React tyto situace usnadňuje vytvářením komponent uživatelského rozhraní, které lze skládat a opakovaně používat a které reagují na změny v datech a v interakcích uživatelů. [17]

---

[16] ZAMMETTI, Frank. Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker. Berlin, Germany: Apress (Verlag), 2020. ISBN 978-1-4842-5737-1.
[17] LARSEN, John. React Hooks in Action: With suspense and concurrent mode. Manning Publications Co., 2021. ISBN 9781617297632.

React je obecně považován za zobrazovací vrstvu – pokud je na aplikaci nahlíženo jako na množinu různých vrstev, kdy Javascript s knihovou React umí zajistit zobrazovací část. React ovlivňuje, co uživatel vidí.[18]

Se vzestupem Reactu a podobných knihoven je vidět posun ve způsobu, jakým vývojáři píší uživatelské rozhraní. Tyto knihovny poskytují prostředky pro správu stavu uživatelského rozhraní na úrovni komponent. To uživatelům umožňuje, aby aplikace běžely bezproblémově a zároveň poskytovaly dobrou vývojářskou zkušenost. S nástroji jako je Electron (pro vytváření desktopových aplikací) a React Native (pro multiplatformní nativní mobilní aplikace) mohou nyní vývojáři využít tyto vzory ve všech svých aplikacích.[19]

React provádí vykreslování pomocí virtuální implementace DOM, čímž se odlišuje od ostatních knihoven webového uživatelského rozhraní, které zpracovávají aktualizace stránek nákladnými manipulacemi přímo v DOM prohlížeče.[20]

4.3 Microsoft SQL databáze

Hlavní myšlenka konceptu relační databáze je založená na uchovávání dat v tabulkách. V těch jsou data uspořádána do jednotlivých řádků a sloupců. Pro některé tabulky jsou údaje v některých sloupcích společné, slouží tak k vyjádření vzájemných relací (vztahů) mezi datovými tabulkami. Tato koncepce tvoří základ databázových systémů používaných v dnešní době.

Firma IBM vyvinula první verzi dotazovacího jazyka SQL pro práci s relačními databázemi. Mezi první databázové systémy, které užívají jazyk SQL, je Oracle. Další systémy, které tyto databáze využívají, jsou Access, SQL Server, MySQL, SQLite, a jiné.

---

[18] BODUCH, Adam a Roy DERKS. React and React Native: A complete hands-on guide to modern web and mobile development with React.js, 3rd Edition. 3. vyd. Birmingham, England: Packt Publishing Limited (Verlag), 2020. ISBN 978-1-83921-114-0.
[19] SCOTT, Adam. JavaScript everywhere: Building cross-platform applications with GraphQL, react, react native, and electron. Sebastopol, CA: O'Reilly Media, 2020. ISBN 9781492046981.
[20] HOQUE, Shama. Full-Stack React Projects: Modern web development using React 16, Node, Express, and MongoDB. Birmingham, UK: Packt Publishing, 2018. ISBN 9781788835534.

Počátek vývoje jazyka SQL začal uveřejněním koncepce relační databáze (E.F. Codd 1970), kdy výzkumná laboratoř IBM vyvíjela prototyp relační databáze. Brzy se ukázalo, že je třeba vytvořit také jazyk, který bude schopen s relačními databázemi pracovat. Tím vznikl dotazovací jazyk zvaný SEQUEL, později nazývaný SQL.

Roku 1977 firma s názvem Relational Software představila na trhu databázový systém, který se nazývá Oracle. Tento systém již jazyk SQL užíval. Vzhledem k tomu, že byl jazyk SQL používán i u dalších databázových systémů pracujících pod operačním systémem DOS, objevila se snaha o jeho standardizaci.

První standardní verze jazyka SQL byla vytvořena v roce 1982 pracovníky v instituci ANSI (American National Standards Instituce). Během let byl tento standard několikrát přepracován a doplněn. V dnešní době je jazyk SQL využíván v řadě databázových programů, nicméně téměř nikdo nevyužívá standardní verzi v plné šíři.

Dobře navržená databáze je základem každého informačního systému. Očekává se od ní stabilita, spolehlivost, rychlost a bezpečnost uložených dat. Pro společnost je databáze důležitá, neboť chrání cenná data. Relační databáze nabízí více softwarových společností, např. Microsoft, Oracle, IBM atd. Databáze mohou být volně dostupné nebo komerčně distribuované. Každá má různě specifikované hardwarové nároky, které se týkají procesorové kapacity, paměti a diskového prostoru.

Databáze užívané v dnešní době tvoří základ pro stabilní a kvalitní přístup k údajům. Obsahují bezpečné technologie, které umožňují přístup k údajům systému odkudkoliv ve světe.

S vývojem databázových technologií začaly nastupovat nové trendy. Patří mezi ně přechod na architekturu klient-server a také trend decentralizace informačních technologií, která byla vynucena globalizací ekonomiky a dynamickými změnami v řízení a struktuře firem. Databázový server je základním pilířem třívrstvé architektury klient-server-databáze. Tento server je možné

realizovat na samotných uzlech nebo na stejném hardwarovém serveru jako middlewarová vrstva. [21]

5 Závěr

V rámci této bakalářské práce byla úspěšně navrhnuta a vytvořena aplikace, jejímž cílem byla automatizace procesu vkládání profesních životopisů na webový portál úřadu. Jednalo se zejména o zjednodušení procesních kroků, které bylo možné nahradit automatizací a zajistit tak lepší plynulost celého procesu.

V první řadě byla provedena analýza tohoto procesu bez systému, aby následný navrhovaný systém co nejlépe odpovídal potřebám všech pracovníků. Jednalo se zejména o stanovení daných úkolů, které má aplikace zajišťovat. Následně byl vytvořen model systému a analytický diagram, které vycházely z předchozích návrhů.

Aplikace obsahuje základní funkce, jejichž využití bude společné pro všechny úřady státní správy. Mezi tyto základní funkce je možné zařadit ukládání a zneplatnění neaktuálních souborů a především vedení procesu pomocí stavů. V aplikaci je také funkce, která zajišťuje odesílání notifikací o nutnosti vložení životopisu. Tato funkce je naprogramována a připravena pro spuštění, ale pro její funkčnost je nutná implementace konfigurace emailového serveru daného úřadu. S ohledem na absenci emailového serveru nebylo možné tuto funkcionalitu otestovat.

S přihlédnutím na subjektivní potřeby daných úřadů je možné do aplikace implementovat další funkcionality, které by dále rozšiřovaly daný systém a aplikace by byla uživatelsky přívětivější. Jedním z možných rozšíření je administrace pro správce – možnost pro vložení nového správce, přidělení oprávnění správci a také možnost pro zrušení role správce. Další by mohla být možnost, aby správce aplikace mohl vkládat a upravovat soubor, který by fungoval jako šablona pro vytvoření profesního životopisu. Aplikace by také mohla být rozšířená o možnost upravovat

---

[21] LACKO, Ľuboslav. 1001 tipů a triků pro SQL. Brno: Computer Press, 2011. ISBN 978-80-251-3010-0.

text zasílané notifikace správcem. Služba pro automatické rozesílání notifikací by mohla dále obsahovat funkci, která by ověřovala, zda vložené soubory představených jsou správně zařazeny v souborovém systému. V případě nenalezení souboru by byl představený požádán notifikací o nahrání nového souboru a jeho záznam byl nastaven do stavu 1 „Čeká na vložení“.

Je mnoho možností, jak tuto základní aplikaci rozšířit, ale tato aplikace již sama o sobě splňuje stanovené požadavky a cíle. Aplikace umožňuje pracovníkům provádět úkony jednoduše a efektivně, díky čemuž je celý proces vkládání životopisů na webové stránky uživatelsky přívětivý. Aplikace byla vyvinuta pomocí použití moderních technologií tak, aby byla snadno použitelná a intuitivní. Testování aplikace ukázalo, že splňuje funkčnost a klíčové požadavky.

6 Seznam použité literatury

1. Ag-grid [online]. AG Grid, 2022 [cit. 2023-02-19]. Dostupné z: https://ag-grid.com/javascript-data-grid/grid-features/
2. Akční plán boje s korupcí na rok 2016, Úřad vlády České republiky Ministr pro lidská práva, rovné příležitosti a legislativu a předseda Rady vlády pro koordinaci boje s korupcí, Předkladatel: ministr pro lidská práva, rovné příležitosti a legislativu Praha, prosinec 2015
3. Anderson Rick, Přidání kontroleru, Microsoft Documents, 13.5.2021, https://docs.microsoft.com/cs-cz/aspnet/mvc/overview/getting-started/introduction/adding-a-controller
4. Anderson Rick, Přístup k datům modelu z kontroleru, Microsoft Documents, 13.5.2021, https://docs.microsoft.com/cs-cz/aspnet/mvc/overview/older-versions/getting-started-with-aspnet-mvc4/accessing-your-models-data-from-a-controller#strongly-typed-models-and-the-model-keyword
5. Axios: Promise based HTTP client for the browser and node.js [online]. [cit. 2023-01-18]. Dostupné z: https://axios-http.com/

6. Bill Wagner, Výčtové typy (Referenční dokumentace jazyka C#), 4.3.2022, https://docs.microsoft.com/cs-cz/dotnet/csharp/language-reference/builtin-types/enum
7. BODUCH, Adam a Roy DERKS. React and React Native: A complete hands-on guide to modern web and mobile development with React.js, 3rd Edition. 3. vyd. Birmingham, England: Packt Publishing Limited (Verlag), 2020. ISBN 978-1-83921-114-0.
8. HOQUE, Shama. Full-Stack React Projects: Modern web development using React 16, Node, Express, and MongoDB. Birmingham, UK: Packt Publishing, 2018. ISBN 9781788835534.
9. HttpContext User Vlastnost, System web, https://docs.microsoft.com/cs-cz/dotnet/api/system.web.httpcontext.user?view=netframework-4.8&viewFallbackFrom=net-5.0
10. Introduction. React Router [online]. remix: remix, 2022 [cit. 2022-03-31]. Dostupné z: https://reactrouter.com/docs/en/v6/getting-started/tutorial
11. LACKO, Ľuboslav. 1001 tipů a triků pro SQL. Brno: Computer Press, 2011. ISBN 978-80-251-3010-0.
12. LARSEN, John. React Hooks in Action: With suspense and concurrent mode. Manning Publications Co., 2021. ISBN 9781617297632.
13. Práce s DbContext, Microsoft Documents, 30.11.2021, https://docs.microsoft.com/cs-cz/ef/ef6/fundamentals/working-with-dbcontext
14. React: Using the Effect Hook [online]. Meta Platforms, 2023 [cit. 2023-02-19]. Dostupné z: https://reactjs.org/docs/hooks-effect.html
15. Referenční Entity Framework Core nástrojů – Správce balíčků Console v Visual Studio, 14.1.2022, https://docs.microsoft.com/cs-cz/ef/core/cli/powershell#scaffold-dbcontext
16. ROBINSON, Simon. C#: programujeme profesionálně. Brno: Computer Press, 2003. Programmer to programmer. ISBN 80-251-0085-5.
17. SCOTT, Adam. JavaScript everywhere: Building cross-platform applications with GraphQL, react, react native, and electron. Sebastopol, CA: O'Reilly Media, 2020. ISBN 9781492046981.

18. Scott Addie, Konfigurace ověřování systému Windows v ASP.NET Core, 13.5.2021, https://docs.microsoft.com/cs-cz/aspnet/core/security/authentication/windowsauth?view=aspnetcore-6.0&tabs=visual-studio
19. VIRIUS, Miroslav. Programování v C#: od základů k profesionálnímu použití. Praha: Grada Publishing, 2021. Knihovna programátora (Grada). ISBN 978-80-271-1216-6
20. ZAMMETTI, Frank. Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker. Berlin, Germany: Apress (Verlag), 2020. ISBN 978-1-4842-5737-1.

7 Seznam obrázků

8 Seznam zdrojových kódů

9 Přílohy

1) Příloha č.1 - Usnesení vlády
2) Příloha č.2 - Akční plán

USNESENÍ
VLÁDY ČESKÉ REPUBLIKY

ze dne 14. prosince 2015 č. 1033

o Akčním plánu boje s korupcí na rok 2016

Vláda

I. schvaluje Akční plán boje s korupcí na rok 2016, obsažený v části III materiálu čj. 1537/15 (dále jen „Akční plán");

II. ukládá

1. členům vlády a vedoucím ostatních ústředních správních úřadů neuvedeným v článku III tohoto usnesení

a) plnit opatření uvedená v Akčním plánu,

b) do 31. ledna 2017 zaslat ministru pro lidská práva, rovné příležitosti a legislativu a předsedovi Rady vlády pro koordinaci boje s korupcí podrobnou zprávu o stavu a způsobu splnění úkolů obsažených v Akčním plánu a spadajících do jejich gesce,

2. ministru pro lidská práva, rovné příležitosti a legislativu a předsedovi Rady vlády pro koordinaci boje s korupcí předložit vládě

a) do 30. listopadu 2016 návrh Akčního plánu boje s korupcí na rok 2017,

b) do 31. března 2017 zhodnocení plnění opatření uvedených v Akčním plánu,

3. ministru vnitra zajistit zveřejnění tohoto usnesení ve Věstníku vlády pro orgány krajů a orgány obcí;

III. doporučuje prezidentovi Nejvyššího kontrolního úřadu, řediteli Generální inspekce bezpečnostních sborů a předsedkyni Úřadu pro ochranu osobních údajů postupovat podle bodu II/1 tohoto usnesení.

Provedou:
členové vlády, vedoucí ostatních
ústředních správních úřadů

Na vědomí:
prezident Nejvyššího kontrolního úřadu,
ředitel Generální inspekce bezpečnostních sborů

Mgr. Bohuslav Sobotka, v. r.
předseda vlády

Úřad vlády České republiky
Ministr pro lidská práva, rovné příležitosti a legislativu
a předseda Rady vlády pro koordinaci boje s korupcí

Akční plán boje s korupcí na rok 2016

Předkladatel: ministr pro lidská práva, rovné příležitosti a legislativu

Praha, prosinec 2015

1. Výkonná a nezávislá exekutiva

Dne 1. ledna 2015 nabyl účinnosti zákon č. 234/2014 Sb., o státní službě, jehož základními cíli jsou depolitizace (transparentní výběrová řízení, trvání služebního poměru bez ohledu na politické změny, rigidnější proces schvalování systemizace služebních míst), stabilizace (systém kariérního růstu, mechanismy pro motivaci státních zaměstnanců setrvat ve služebním poměru) a profesionalizace (úřednická zkouška a služební hodnocení, systém vzdělávání státních úředníků). Současně byla přijata většina prováděcích předpisů nezbytných k efektivní aplikaci zákona o státní službě.

V roce 2016 bude realizována většina výběrových řízení na pozice představených a dále budou státními zaměstnanci skládány úřednické zkoušky v předepsaných oborech služby. Úspěšná implementace zákona o státní službě je klíčovým závazkem vlády. Momentálně jsou nastaveny účinné institucionální mechanismy v působnosti Sekce pro státní službu Ministerstva vnitra a státních tajemníků na jednotlivých služebních úřadech, které vytvářejí reálné předpoklady pro to, aby jednotlivá ustanovení zákona o státní službě a jeho prováděcích předpisů byla aplikována způsobem zajišťujícím kvalitní fungování státní správy. Podpůrným opatřením, které výrazně napomůže prosazování prvků transparentnosti při obsazování vedoucích pozic ve státní službě, je **zveřejnění profesních životopisů představených** od úrovně ředitelů odboru, kteří uspějí ve výběrových řízeních v průběhu roku 2016, na webu příslušných služebních úřadů (vyjma zpravodajských služeb). Bez souhlasu dotčeného představeného budou zveřejněny profesní životopisy, pokud se ve smyslu § 5 odst. 2 písm. f) zákona č. 101/2000 Sb., o ochraně osobních údajů jedná o *„osobní údaje o veřejně činné osobě, funkcionáři či zaměstnanci veřejné správy, které vypovídají o jeho veřejné anebo úřední činnosti, o jeho funkčním nebo pracovním zařazení"*. Ostatní údaje mohou být zveřejněny pouze se souhlasem dotčeného představeného. Jedná se o opatření, kterým dojde k dovršení procesu transparentního výběru představených ve státní správě, tj. státních zaměstnanců, kteří se budou na dlouhou dobu spolupodílet na klíčových rozhodnutích ovlivňujících životy občanů České republiky.

V souvislosti s účinností a implementací zákona o státní službě je žádoucí, aby byla důkladně srovnána práva a povinnosti státních zaměstnanců a úředníků územních samosprávných celků. Ministerstvo vnitra v roce 2016 **zahájí přípravu nelegislativního materiálu (analýzy), který se bude věnovat srovnání jednotlivých aspektů činnosti, právům a povinnostem státních zaměstnanců a úředníků územních samosprávných celků**.

V roce 2015 byl v souladu s Plánem legislativních prací vlády na rok 2015 a Akčním plánem boje s korupcí na rok 2015 rozeslán do mezirezortního připomínkového řízení návrh zákona o státním zastupitelství. Z hlediska stanovených parametrů dotčený návrh většinově splňoval požadavky obsažené v Akčním plánu boje s korupcí na rok 2015. Nejvíce diskutovaným opatřením je zřízení **Speciálního státního zastupitelství** (dále jen „SSZ"), zejména s ohledem na atrakci/delegaci jednotlivých případů, možnosti opakovaného funkčního období vedoucího SSZ a absenci dohledu nad SSZ ze strany Nejvyššího státního zastupitelství. Předkladateli návrhu zákona o státním zastupitelství se nepodařilo zajistit jeho schválení vládou ve stanoveném termínu a není zřejmé, zdali se tak stane do konce roku 2015. Přetrvávají však minimální požadavky na jeho podobu obsažené v Akčním plánu boje s korupcí na rok 2015.

Předložení návrhu zákona o státním zastupitelství vyplývá jak z programových a protikorupčních dokumentů vlády, tak i z mezinárodních závazků ČR (Evropská komise, GRECO), proto tomuto úkolu musí být ze strany Ministerstva spravedlnosti v roce 2016 věnována maximální pozornost. Druhou fázi, v případě schválení zákona o státním zastupitelství Parlamentem, představuje proces zřízení SSZ, zrušení obou vrchních státních zastupitelství včetně přeložení vrchních státních zástupců na jiná (nikoliv okresní) státní zastupitelství a vhodné zakotvení výběrových řízení na pozice vedoucích státních zástupců. Aniž by tento Akční plán stanovoval detailnější parametry jednotlivých organizačně-technických procesů, jedná se o mechanismy, kterým bude důkladnou pozornost v příslušném okamžiku věnovat i Rada.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Tomáš Trnka**
Studium:	I1600497
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Vývoj aplikace pro státní správu**
Název bakalářské práce AJ:	Application development for state administration

Cíl, metody, literatura, předpoklady:

Cílem práce je navrhnout funkční aplikaci určenou pro státní správu dle strategie vlády ČR a její koncepce pro boj s korupcí. Aplikace by měla automatizovat proces vkládání profesních životopisů a také upozorňovat na nutnost jejich vložení na webový portál daného úřadu.

ROBINSON, Simon. C#: programujeme profesionálně. Brno: Computer Press, 2003. Programmer to programmer. ISBN 80-251-0085-5
VIRIUS, Miroslav. Programování v C#: od základů k profesionálnímu použití. Praha: Grada Publishing, 2021. Knihovna programátora (Grada). ISBN 978-80-271-1216-6

Zadávající pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	prof. RNDr. PhDr. Antonín Slabý, CSc.
Datum zadání závěrečné práce:	26.1.2021