Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Migrace Access Forms do webového rozhraní

Marek Pilař

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Marek Pilař

Informatika

Název práce

Migrace Access Forms do webového rozhraní

Název anglicky

Migrating Access Forms to the web interface

Cíle práce

Cílem práce je migrace aplikace Microsoft Access forms do webového prostředí. Stávající Access forms aplikace má data uložená na Microsoft SQL serveru a Access forms je využíván pouze jako front end pro zadávání nových dat.
Dílčím cílem této práce je analyzovat stávající řešení (datová struktura, aplikační logika, use case aplikace, řízení oprávnění atd.).

Metodika

Metodika práce je založena na studiu odborných zdrojů. Dále bude navrženo nové řešení, které by mělo mít webové rozhraní. Pro nové řešení bude pomocí analýzy vybrána technologii pro tvorbu frontendu a backendu. Dále bude rozhodnuto, zda bude služba provozována lokálně či bude fungovat za pomocí cloudu. V rámci zpracování bude analyzována bezpečnost navržené aplikace a vyhodnocena případná rizik, na základě kterých bude vytvořeno bezpečnostní omezení.

Doporučený rozsah práce

35-45s.

Klíčová slova

Microsoft Access, migrace, web, databáze, cloud

Doporučené zdroje informací

FREEMAN, Adam a SpringerLink (online služba). Pro ASP.NET Core MVC 2 [online].7th. Berkeley, CA: Apress, 2017. ISBN 1484231503;9781484231500;148423149X;9781484231494

JAPIKSE, Philip et al. Building Web Applications with Visual Studio 2017: Using .NET Core and Modern JavaScript Frameworks [online].Berkeley, CA: Apress, 2017. ISBN 9781484224786;1484224787;1484224779;9781484224779

MORGADO, Flavio a SpringerLink (online služba). Introducing Microsoft Access using macro programming techniques: an introduction to desktop database development by example [online].New York, NY: Apress, 2021. ISBN 9781484265550;1484265556

RAMOS APOLINARIO, Vinicius a SpringerLink (online služba). Windows Containers for IT Pros: transitioning existing applications to containers for on-premises, cloud, or hybrid [online].New York: Apress, 2021. ISBN 1484266862;9781484266861

STRAUSS, Dirk a SpringerLink (online služba). Creating ASP.NET Core web applications: proven approaches to application design and development [online].New York: Apress, 2021. ISBN 1484268288;9781484268285

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Martin Havránek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 14. 7. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 27. 10. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 15. 03. 2023

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Migrace Access Forms do webového rozhraní" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15. 3. 2023 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Martinu Havránkovi, Ph. D. za odborné vedení bakalářské práce, věnovaný čas a cenné rady. Také bych chtěl poděkovat kolegům ze společnosti Trask solutions a.s. Zejména Bc. Tomáši Kratochvílovi a Ing. Martinu Brůžkovi za umožnění psát práci na uvedenou tématiku a věnovaný čas potřebný pro konzultaci této problematiky. Poděkování také patří mé rodině a nejbližším za to, že mi byli vždy oporou jak během psaní bakalářské práce, tak i během celého studia.

Migrace Access Forms do webového rozhraní

Abstrakt

Hlavním cílem této bakalářské práce je migrace aplikace vytvořené za pomocí Microsoft Access Forms do webové aplikace.

Teoretická část se věnuje seznámení s technologiemi, které využívá původní aplikace. V teoretické části se čtenáři seznámí s dalšími technologiemi, které jsou využity k dosažení cíle této bakalářské práce. Jedno z hlavních témat je technologie WebAssembly.

Praktická část této práce se zabývá podrobnou analýzou původní aplikace. Následně se věnuje sběru požadavků na novou webovou aplikaci. Na základě těchto kroků jsou vybrány vhodné technologie a tyto technologie jsou následně implementovány do nové aplikace. Výsledkem je následně nová webová aplikace, která obsahuje data z původní aplikace a umožňuje provádět stejné operace jako původní aplikace, a však s vyšším zabezpečením a komfortem.

Hlavním přínosem práce je, že může sloužit jako metodický postup pro migraci obdobné aplikace. Dále je součástí práce výběr vhodných technologií, potřebných pro implementaci. Výběr těchto technologií je odůvodněn a zároveň je i zmíněno, na jaký typ webové aplikace jej lze použít.

Klíčová slova: Microsoft Access, Access formulář, migrace, webová aplikace, databáze, .NET, C#, WebAssembly, Blazor, SPA

Migrating Access Forms to the web interface

Abstract

The main goal of this bachelor's thesis is to migrate an application created using Microsoft Access Forms into a web application.

The theoretical part focuses on introducing the technologies used in the original application. Furthermore, the theoretical part of this thesis presents many other technologies that are used to achieve the goal of this bachelor's thesis. One of the main topics is the WebAssembly technology.

The practical part of this thesis deals with a detailed analysis of the original application. Subsequently, it collects requirements for the new web application. Based on these steps, suitable technologies are selected. These technologies are then implemented into the new application. The result is a new web application that contains data from the original application and allows performing the same operations as the original application, but with higher security and comfort.

The main benefit of this work is that it can serve as a methodical procedure for migrating a similar application. In addition, the work includes the selection of suitable technologies necessary for implementation. The selection of these technologies is justified, and it is also stated for which type of web application it can be used.

Keywords: Microsoft Access, Access form, migration, web application, database, .NET, C#, WebAssembly, Blazor, SPA

Obsah

1 Úvod

Každá firma potřebuje evidovat své informace v nějakém informačním systému. Pokud se jedná o začínající firmu, tak se zpravidla jedná o jednoduché a cenově dostupné řešení, které odpovídá požadavkům daného podniku. S postupným růstem podniku se většina firem dostane do bodu, kdy aktuální řešení již není dostačující a je potřeba staré systémy upgradovat nebo integrovat do nových systémů. Případně vytvořit řešení upravené a reflektující specifické potřeby firmy.

Dost podstatnou roli hraje i fakt, kdy daná firma začala podnikat, jelikož řešení a technologie, které byly k dispozici před více než 20 lety, jsou nyní výrazně překonány. V dnešní době to mají nové startupy o poznání jednodušší. Například, když chce startup nové CRM řešení, případně jiný komplexní systém, tak v dnešní době na trhu působí mnoho firem, které tato řešení poskytují za přijatelné peníze. Tyto softwary bývají obvykle lehce škálovatelné a jejich licenční modely se často mohou odvíjet od počtu uživatelů nebo potřebného výkonu. To je pro nové startupy velice přínosné v tom, že nemusí platit obrovské částky jako firmy, které začínaly před 20 lety a musely platit za nějaký on-premise systém, který v té době stál mnohem více než řešení, které se dá dnes na trhu sehnat.

Z tohoto důvodu se tehdejší začínající firmy s omezeným kapitálem a s omezenými technologiemi uchylovaly k tomu, že často využívaly technologie, které měly k dispozici a pokoušely se z nich vytvořit systém, který jim bude v tehdejších podmínkách vyhovovat. Jako příklad lze uvést evidenci informací o zaměstnancích. Nejjednodušší a nejlevnější způsob, jak s těmito informacemi pracovat, byly kancelářské balíčky typu Microsoft Excel a Microsoft Access. Dnes se to může zdát někdy až úsměvné, jelikož v dnešní době se dají sehnat online služby, které jsou pro práci s těmito informacemi vytvořené a obvykle jsou i levnější než zakoupení kancelářského balíčku Microsoft office. Proto, starší firmy musí hledat řešení, co udělat se svými zastaralými systémy a tomuto problému se věnuje tato bakalářská práce.

2 Cíl práce a metodika

2.1 Vymezení tématu práce a důvod výběru tématu

Problematice napsané v úvodu se budu věnovat na konkrétním příkladu. Jako příklad jsem vybral jeden zastaralý systém firmy Trask Solutions a. s., který byl vytvořen pomocí kancelářského balíčku Microsoft Access a slouží oddělení HR pro evidenci údajů o zaměstnancích.

Hlavní motivací pro výběr tématu byla má praktická zkušenost v dané firmě. Zjistil jsem, že současné řešení je potřeba nahradit novějším, lepším a jsem rád, že jsem měl příležitost se zabývat tímto problémem z praxe, který je přínosný pro mě jako autora práce tak i pro společnost Trask Solutions a. s.

2.2 Cíl práce

Cílem práce je migrace aplikace Microsoft Access forms do webového prostředí. Stávající Access forms aplikace má data uložená na Microsoft SQL serveru a Access forms je využíván pouze jako front end pro zadávání nových dat. Tudíž výstupem této práce by měla být nová webová aplikace, která bude používat stejný datový model (s možnými úpravami, které nesměji mít za následek ztrátu funkcionality či části dat). Výsledná aplikace by měla plně nahradit aplikaci v MS Access.

Dílčími cíli této práce jsou:

- Analýza stávajícího řešení:
    - datová struktura
    - aplikační logika
    - řízení oprávnění
- Analýza požadavků firmy
- Vytvoření dokumentace

2.3 Metodika

Metodický postup je rozdělen do čtyř kategorií:

1. Analýza aktuálního řešení: V této fázi jsou analyzovány současné metody nebo technologie, které jsou využívány v rámci původní aplikace. Cílem je identifikovat slabiny a nedostatky stávajícího řešení, které je třeba vylepšit nebo nahradit.
2. Analýza požadavku firmy: V této fázi jsou zjištěny požadavky a potřeby firmy, které využívají daného řešení. Cílem je získat co nejvíce informací o tom, jakými způsoby firma využívá současné řešení, a co by potřebovala, aby byla schopna lépe splnit své cíle.
3. Studie odborných zdrojů: V této fázi jsou prozkoumány odborné zdroje. Cílem je získat přehled o nejnovějších trendech a technologiích v oblasti vývoje webových aplikací a zjistit, jaké možnosti jsou k dispozici pro vylepšení řešení daného problému.
4. Výběr vhodných technologií: Na základě analýzy stávajícího řešení, požadavků firmy a studie odborných zdrojů, jsou vybírány nejvhodnější technologie, které by mohly být použity k vylepšení nebo nahrazení stávajícího řešení. Cílem je zvolit takové technologie, které by nejlépe odpovídaly požadavkům firmy a byly by efektivní a udržitelné v dlouhodobém horizontu.
5. Ověření pomocí implementace: V této fázi jsou vybrané technologie implementovány a otestovány v reálném prostředí. Cílem je zjistit, zda vybrané technologie opravdu fungují a splňují požadavky firmy. Pokud ne, tak přizpůsobit řešení, aby bylo co nejefektivnější a neúčinnější.

3 Teoretická východiska

Tato bakalářská práce se v teoretické části zaměřuje na seznámení se s technologiemi používanými v původní aplikaci. V rámci literární rešerše budu hledat alternativní technologie, které by mohly nahradit původní řešení. Poté budu v praktické části na základě poznatků získaných o nových technologiích vybírat technologický strom pro implementaci nové aplikace.

3.1 Relační databáze

Princip relačních databází spočívá v uložení dat v logickém smyslu. Pod pojmem relace chápeme jako základní stavební prvek tabulky. Tabulky mají mezi sebou definované vazby 1:1, 1: n anebo m: n. Řádky v tabulkách představují jednotlivé záznamy a sloupce reprezentují jednotlivé atributy. Každý atribut má definován datový typ, kterému musí jednotlivé záznamy odpovídat. Díky tomu jsou relační databáze konzistentní, strukturované a také je možné data jednoduše třídit a provádět nad nimi výpočty. K tomu využíváme dotazovací jazyk SQL (Morgado, 2021).

3.2 Microsoft Access

Microsoft Access je program s uživatelským rozhraním určeným ke správě lokální nebo vzdálené relační databáze. Tento software je součástí kancelářského balíčku Microsoft office a je vyvíjen od roku 1992 firmou Microsoft (Microsoft Access, 2022). Access je takzvaně single file interface program, tzn. každá jednotlivá instance tohoto programu (jednotlivé okno) dokáže obsluhovat pouze jednu lokální databázi. Pro obsluhu další databáze je potřeba otevřít další okno. Lokální databázový soubor má příponu *.MDB (verze 2003 a starší). Od verze 2007 (a novější) mají tyto soubory příponu *.ACCDB. Tento program je podporován pouze pro operační systém Windows a je dostupný po instalaci tohoto software (Morgado, 2021).

3.3 Microsoft Access forms

Access forms jsou součástí kancelářského softwaru Microsoft Access. Jak již napovídá název, tento balíček umožňuje využívat již vytvořené uživatelsky přívětivé komponenty. Z těchto komponentů lze vytvořit formuláře pro vytváření uživatelsky přívětivých a intuitivních

aplikací pro práci s daty. Formuláře umožňují uživatelům vytvářet, číst, upravovat a mazat záznamy v databázových tabulkách (Morgado, 2021).

Microsoft Access forms obsahují různé komponenty, jako jsou textová pole, výběrová pole, tlačítka, tabulky a další. Tyto prvky lze uspořádat do různých formátů, jako jsou základní formuláře, subformuláře, dialogová okna atd. Formuláře mohou být také personalizovány pomocí VBA (Visual Basic for Applications) kódu pro vytvoření pokročilých funkcí, jako jsou automatické úkoly, validace dat a další (Morgado, 2021).

Formuláře v Microsoft Access jsou důležitým nástrojem pro zjednodušení práce s daty a zlepšení produktivity. Tyto formuláře umožňují vývojářům vytvářet uživatelsky přívětivé aplikace pro řízení dat, bez nutnosti hlubšího znalosti databázových systémů. To významně urychluje proces vývoje aplikací a snižuje tím náklady na vývoj a údržbu (Morgado, 2021).

Tyto formuláře nejsou striktně omezeny pouze pro komunikaci s lokální databází MS Access, ale lze je i použít pro práci s daty na vzdálených relačních databázových serverech typu Microsoft SQL případně MYSQL (Morgado, 2021).

3.4 Access macro programming

Access macro programing se používá k automatizaci databázových procesů a přidání různých funkcionalit. Pod databázovými procesy si můžeme představit jakékoli procesy spojené s tabulkami, formuláři či sestavami. Například se jedná o otevírání a zavírání formulářů, spouštění sestav, případně validaci vstupů, nebo jejich transformaci před odesláním (Morgado, 2021).

3.5 Microsoft SQL server

Microsoft SQL Server je databázový management systém (RDBMS) od společnosti Microsoft. Z pravidla se jedná o relační databáze, která umožňují ukládat, spravovat a získávat data z různých aplikací a služeb. SQL Server nabízí široké spektrum funkcí pro správu dat, včetně bezpečnosti, kopírování, zálohování a obnovy, analýzy a řízení dat (Radivojević a kolektiv, 2019).

Je využíván v široké škále aplikací, včetně podnikových informačních systémů, e-commerce, webových aplikací, cloudových služeb a dalších. Tyto aplikace využívají funkce SQL Serveru pro správu velkých objemů dat a zajištění vysoké dostupnosti a výkonu (Radivojević a kolektiv, 2019).

SQL Server nabízí také široké možnosti integrace s jinými nástroji a technologiemi, včetně Microsoft Excel, SharePoint, Power BI a dalších. Tyto integrace umožňují vývojářům využívat pokročilé funkce SQL Serveru pro vývoj vlastních aplikací a řešení (Radivojević a kolektiv, 2019).

Je nabízen ve více edicích včetně Express, Standard, Enterprise a Developer, které umožňují uživatelům vybrat edici, která nejlépe vyhovuje jejich potřebám a rozpočtu. To znamená, že SQL Server je dostupný jak pro malé a střední podniky, tak pro velké podniky s náročnými potřebami (Radivojević a kolektiv, 2019).

3.6 WebAssambly

Standart WebAssembly byl představen v roce 2015. Mezi zakladatele patří Microsoft, Apple, Google. Jedná se o poměrně přelomovou technologii, jelikož umožňuje spouštět binární kód přímo v prohlížeči, bez žádných dalších instalací. Tuto technologii podporuje většina moderních prohlížečů, jako je Opera, Safari, Google Chrome, Microsoft Edge. Web assambly lze spustit i na mobilních zařízeních. Velkou výhodou je, že díky této technologii lze spouštět knihovny i v jiných programovacích jazycích, než je Java Script. Velký posun to je i pro herní průmysl, kde lze přenést celý kód hry na klienta a následně celu hru spustit. Dále umožňuje tuto hru hrát i v off-line režimu, a proto není potřeba být připojen k internetu (WebAssembly, 2023).

3.7 .NET a .NET CORE

První verze .NET Core frameworku byla představena 12. 11. 2014. První verze vyšla až po několika letech, a to 27. 6. 2016. Tato verze byla pro celý svět .NET přelomová, jelikož na rozdíl od předchozího .NET Frameworku se stal multiplatformní. Pro kompilování případně spuštění již není vyžadován operační systém Microsoft Windows (Price, 2021).

Mezi další velkou výhodu patří to, že je open source a je tvořen z části komunitně. Zdrojový kód lez nalézt na .NET Platform · GitHub (2023). Na GITu lze dále nalézt veškeré issues, případně je možné přidat svojí vlastní issue nebo rovnou přidat nějaký pull request s navrhovanou změnou ve zdrojovém kód (Price, 2021).

Od verzí .NET 5 a novější dále přibyla podpora ARM procesorů, které mají rostoucí trend používání a dá se předpokládat, že tento trend bude zrychlovat. Výhoda ARM procesorů je, že jsou úspornější a nepotřebují tolik chlazení (Price, 2021).

Dříve se používalo označení .NET CORE (verze 1,2,3) a od verze 5 se nazývá pouze .NET a přípona CORE odpadá. Při pohledu na seznam verzí lze zaregistrovat, že neexistuje

verze 4, a to je z důvodu, že Microsoft tuto verzi přeskočil, aby nedocházelo k záměně s .NET Frameworkem, který má aktuální a zároveň nejspíš poslední verzi 4.8 (Price, 2021).

3.8 ASP .NET

ASP.NET je framework pro vývoj webových aplikací, který je vyvinut společností Microsoft. ASP.NET umožňuje vývojářům vytvářet webové aplikace s vysokou úrovní funkčnosti, bezpečnosti a výkonu (Price, 2021).

Je založen na .NET frameworku a nabízí široké spektrum funkcí a nástrojů pro vývoj webových aplikací včetně komponent, knihoven a nástrojů pro tvorbu uživatelského rozhraní. ASP.NET umožňuje vývojářům vytvářet webové aplikace v mnoha jazycích jako na příklad C# a Visual Basic (Price, 2021).

V rámci frameworku lze nalézt řadu bezpečnostních funkcí, jako jsou autentifikace a autorizace, šifrování dat a ochrana proti útokům. Tyto funkce zajišťují, že webové aplikace vytvořené v ASP.NET jsou bezpečné a zabezpečené (Price, 2021).

3.9 Entity Framework (EF)

Jedná se o jednu z nejpoužívanějších knihoven v rámci .NET. Vývoj této knihovny zajišťuje Microsoft. Entity Framework má již poměrně dlouhou historii a byl již ve verzích starého .NET Frameworku. Jedná se o objektově-relační mapovací (ORM) framework pro .NET, který umožňuje vývojářům pracovat s databázovými informacemi jako s objekty a udržovat oddělenou vrstvu mezi databází a aplikací. Tato knihovna se dá jednoduše přidat pomocí správce Nuget balíčků (Price, 2021).

EF podporuje řadu databázových serverů, včetně Microsoft SQL Server, Oracle, MySQL a dalších. Nevětší kompatibilitu má však pouze s Microsoft SQL. U ostatních serverů mohou některé funkce chybět (Price, 2021).

Obvykle jednotlivá třída reprezentuje nějakou tabulku nebo view, které je uloženo v databázi. Zároveň zde lze využít dědičnost případně vazby mezi třídami / tabulkami. EF umožňuje dva způsoby, jak postupovat:

1. **Database first**

Nejdříve se vytvoří datový model v databázi a následně se pomocí nástroje Scaffold-DbContext do zdrojového kódu aplikace vygenerují entity, které reprezentují databázový model. To

znamená, že ze všech vybraných tabulek se vygenerují třídy, zároveň jsou zohledněny vazby mezi tabulkami, případně další datová integrita jako primární klíče a nenulové hodnoty.

2. **Model first**

   Model je vytvořen za pomocí tříd. Následně je vygenerován databázový kód, který se při zahájení migrace spustí. Tzn. třídy se převedou na tabulky, případně další databázové objekty (Price, 2021).

Výhodou je, že není zapotřebí znát syntaxi SQL, protože EF kód v aplikaci převádí na SQL příkazy. Z tohoto důvodu stačí znát pouze syntaxi .NET a znát metody v EF. Zároveň používání ORM zlepšuje bezpečnost vůči SQL Injection, jelikož veškeré textové hodnoty jsou zadány jako parametr a v případě, že se jedná o select, vám umožní vrátit pouze konkrétní dotazovanou třídu (Price, 2021).

3.10 Restfull API

REST (Representational State Transfer) **API (**Application Programming Interface**)** je standardní způsob, jakým aplikace vzájemně komunikují přes síť. Tyto aplikace mohou být webové aplikace, mobilní aplikace, desktopové aplikace atd. REST API se skládá ze dvou hlavních částí: klienta a serveru. Klient posílá HTTP požadavky na server, který následně odpovídá HTTP odpovědí.

REST API se používá k výměně dat mezi aplikacemi v reálném čase. Tyto data mohou být kdekoli uložena, například v databázi, na disku nebo v cloudu. REST API poskytuje standardizovaný způsob, jak tyto data získat, uložit, aktualizovat a odstranit. Tyto operace se provádějí pomocí HTTP metod, jako je GET, POST, PUT a DELETE.

Tabulka 1 - HTTP metody (zdroj: autor)

HTTP metoda	Příklad endpointu	Action	Akce
POST	/employees	**C**reate	Vytvoření nového záznamu
GET	/employees/*{ID}*	**R**ead	Čtení dat
PUT	/employees/**{ID}**	**U**pdate	Úprava záznamu (přepisuje původní)
PATCH	/employees/**{ID}**	Partial **U**pdate	Úprava záznamu (nepřepisuje původní)
DELETE	/employees/**{ID}**	**D**elete	Odstranění záznamu

GET se používá k získání dat z API, POST se používá k odeslání dat na API, PUT se používá k aktualizaci dat v API a DELETE se používá k odstranění dat z API. Tyto metody se záměrně nazývají standardními HTTP metodami, aby se zajistilo, že jsou použitelné na všech webových stránkách, které podporují HTTP.

Výhodou REST API je, že jeho rozhraní je jednoduché a srozumitelné, takže je snadno použitelné pro vývojáře všech úrovní. REST API také umožňuje vývojářům využívat stávající infrastrukturu a nástroje, jako jsou HTTP servery, proxy servery a firewally. REST API je také flexibilní, a může být použito k řešení mnoha různých typů úloh, jako je například synchronizace dat mezi aplikacemi, získávání dat ze sociálních sítí nebo integrace s jinými aplikacemi.

3.11 Java Script

JavaScript je programovací jazyk, který se používá především k vývoji dynamických webových aplikací. Jedná se o interpretovaný jazyk, což znamená, že kód se spouští přímo v prohlížeči bez nutnosti kompilace. Používá ke zlepšení interaktivity a uživatelského rozhraní na webových stránkách, jako jsou například animace, interaktivní formuláře, ajaxové volání atd (Japikse, 2017).

Dalším využitím JS je jako samostatný jazyk pro vývoj serverových aplikací pomocí technologií Node.js. Tento jazyk má silnou komunitu vývojářů a je dostupný jako open-source software. K dispozici je také mnoho knihoven a frameworků, které mohou vývojářům usnadnit a urychlit vývoj aplikací (Japikse, 2017).

JavaScript je jazyk s mnoha vlastnostmi, jako je podpora objektů, funkcí, událostí, asynchronnosti a dalších. Je to jazyk s vysokou úrovní, což znamená, že má jednoduchou syntaxi a vývojářům umožňuje vytvářet aplikace rychle a efektivně. JavaScript se stal jedním z

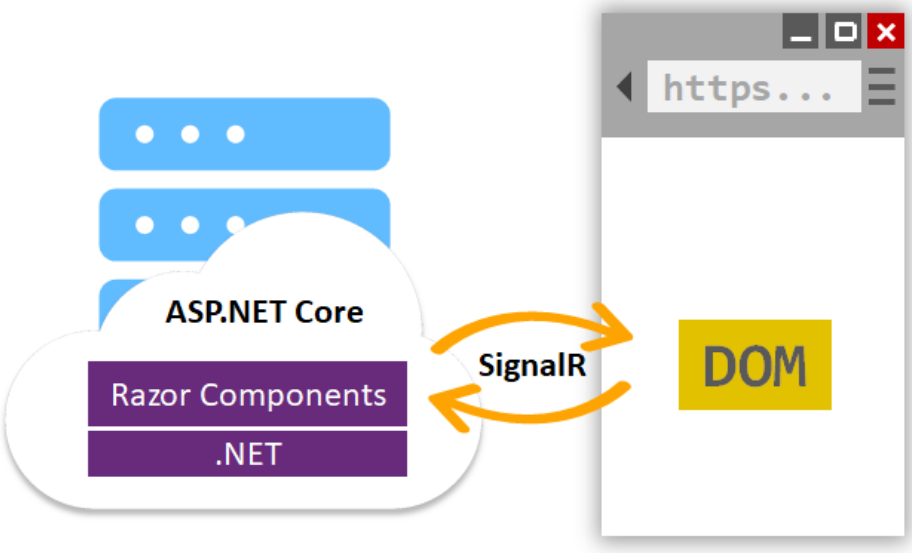
nejpopulárnějších jazyků pro vývoj webových aplikací a stále se rozvíjí, aby uspokojil potřeby vývojářů a nových technologií (Japikse, 2017).

3.12 .NET Blazor

Na úvodní stránce .NET Blazoru je poměrně výstižný titulek: „Use the power of .NET and C# to build full stack web apps without writing a line of JavaScript“ (Microsoft Corporation, 2022). Blazor je nejnovější webových framework v ASP .NET. Hlavní změnou oproti klasickému .NET MVC nebo .NET Razor Pages je využívání jednotlivých komponent. Tyto komponenty se mohou skládat z vícero dalších komponent tzn. že každá komponenta může mít nějakou rodičovskou komponentu nebo může mít několik potomků. Veškeré tyto komponenty si mohou předávat nějaký parametr. Má podobnou struktura jako klasické html s tím rozdílem, že HTML je v těchto komponentách rozšířeno o funkcionality z .NETu. Blazor má dvě základní dělení, a to Blazor Server a Blazor WebAssembly (Microsoft Corporation, 2022).

3.12.1 Blazor Server

Serverový model uchovává veškerou logiku aplikace na serveru, a tím pádem je to poměrně výhodné z hlediska bezpečnosti, jelikož potencionální útočník nevidí žádnou aplikační logiku na straně uživatele. Funguje to tak, že při otevření stránky se vytvoří tzv. SignalR tunel, který funguje na bázi websocetu. Tímto tunelem jde následně veškerá komunikace se serverem, a zároveň se přes tento tunel odesílá veškerá uživatelská interakce. Když server dostane přes tento tunel nějaká data o uživatelské interakci (například stisknutí tlačítka), tak vykoná nějaký definovaný kód a vrátí uživatele přerendrovanou stránku, která se mu zobrazí. Pokusím se tento způsob komunikace uvést na konkrétním příkladu. Máme jednoduchou webovou stránku, kde je vidět nějaká label s číslem 0 a tlačítko s nápisem Click me. Při kliknutí na tlačítko se odešle tento event na server. Server tento požadavek vyhodnotí tak, že proměnou s číslem, který zobrazuje již zmíněny label zvýší o 1. Server následně zasílá přerendrovaný HTML kód uživateli, kde se v labelu změnila hodnota 0 na hodnotu 1. Tento kód se uživateli na konec zobrazí (Microsoft Corporation, 2022).



Obrázek 1 - Schéma komunikace Blazor Server (zdroj: Microsoft Corporation)

Výhody

- Možnost využívání výhod Razor Component a knihoven .NET.
- Celý kód aplikace lze napsat pouze v .NETu a není vyžadována znalost Java Scriptu.
- Uživatel není schopen zobrazit / zneužít aplikační logiku na front endu, která bývá v případě Java Scriptu vidět.
- Validace probíhá vždy na straně serveru, zároveň tím odpadá i nutnost validace na straně klienta.
- Lze spouštět i JavaScriptový kód který je na straně uživatele.

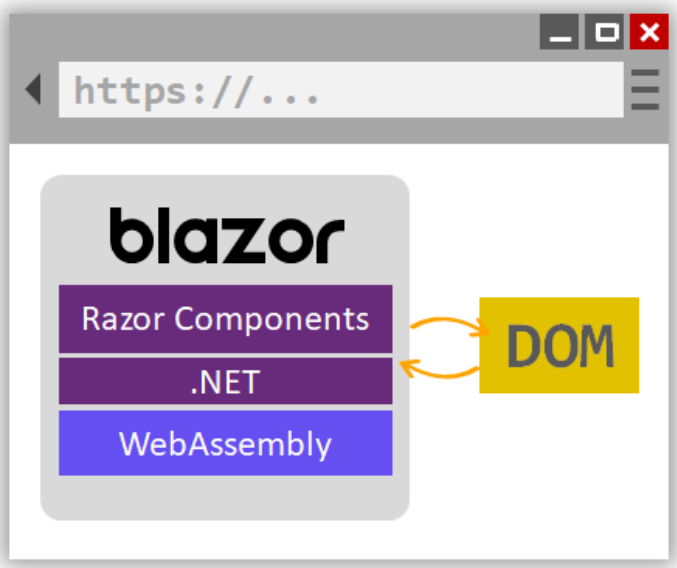
Nevýhody

- Větší zátěž pro server
- V případě horšího připojení k internetu dochází k přerušování tunelu a celé aplikace.

3.12.2 Blazor WebAssembly

Tento způsob rozděluje logiku do serverové části a do klientské části. Na klientské části funguje podobně jako Java Script pouze s tím rozdílem, že front end je napsaný v .NETu místo v JavaScriptu. Front endová část se skládá z *.ddl knihoven, které jsou převedeny na binární kód a jsou následně zpuštěny díky wasm (WebAssembly). Wasm je také poměrně dost nová technologie a je popsána v sekci níže. Proto, kdybychom pomocí WebAssembly chtěli udělat podobnou aplikaci, jakou jsem použil jako příklad u Blazer Serveru, tak by ta aplikace fungovala následovně. Klient se připojí na server -> stáhne si zdrojový kód pro frontend -> kód se za pomocí wasm spustí -> číselná proměnná je nyní uložena u uživatele v počítači nikoliv

na serveru -> uživatel klikne na button -> vykoná se lokálně uložený kód a lokálně uloženou proměnou zvýší o 1 -> následně dojde u uživatel na změnu HTML DOMu a uživatel vidí hodnotu zvýšenou o 1. Princip je obdobný jako u JavaScriptu (Microsoft Corporation, 2022).



Obrázek 2 - Schéma komunikace Blazor WASM (zdroj: Microsoft Corporation)

Výhody

- Možnost využívání výhod Razor Component a jazyku .NET.
- Celý kód aplikace lze napsat pouze v .NETu a není vyžadována znalost Java Scriptu.
- Lze spouštět i JavaScriptový kód.
- Větší výkon a větší množství knihoven (lze použít většinu knihoven pro .NET a spustit jí v prohlížeči).
- Aplikace lze spustit i v off-line režimu.

Nevýhody

- Jelikož se stahují knihovny a spouští se binární kód, tak načtení stránky chvilku trvá. V případě, že se jedná o první přístup na stránku tak cca 4 sec. Opakovaný přístup okolo 1 sec, nicméně záleží na rozsahu aplikace a na počtu knihoven. Při načítání se obvykle zobrazuje nějaká statická HTML stránka s hláškou, že se aplikace načítá.

3.13 Kontejnerizace – Docker

Kontejnerizace je technologie, která umožňuje spouštění aplikací v izolovaných prostředích, nazývaných kontejnery. Nejpoužívanější technologií pro kontejnerizaci je Docker (Ramos Apolinario, 2021).

Docker je platforma pro spouštění, nasazování a správu aplikací ve formě kontejnerů. Kontejnery jsou jednotky softwarového provozu, které obsahují aplikaci a její závislosti, a mohou být spuštěny v jakékoli infrastruktuře, která podporuje Docker. Tyto kontejnery jsou izolované od okolního systému a zajišťují, že aplikace běží na všech systémech stejně. To umožňuje vývojářům a správcům IT nasadit a spravovat aplikace s větší flexibilitou a efektivitou (Ramos Apolinario, 2021).

Umožňuje také snadné sdílení a distribuci aplikací prostřednictvím Docker Hubu, což je centrální místo pro ukládání a stahování kontejnerů. Tyto kontejnery mohou být snadno použity v jiných projektech nebo mohou být použity jako základ pro vývoj a testování nových aplikací. V docker hubu se dá nalézt například kontejner pro spuštění LMS moodle. V rámci kontejneru je webový server nginx, PHP a MySQL. Veškeré tyto technologie jsou nakonfigurované tak, že při spuštění kontejneru se zobrazí stránka s funkčním LMS moodle (Ramos Apolinario, 2021).

Kromě toho Docker umožňuje spravovat zdroje jako jsou CPU, paměť a síť konkrétního kontejneru. Tato možnost umožňuje optimalizaci výkonu aplikace a zajištění, že každý kontejner má dostatek zdrojů pro správný chod. Zároveň je díky tomu kontejner jednodušeji škálovatelný (Ramos Apolinario, 2021).

Docker je funkční na většině operačních systémech jako je Windows, Linux a Mac OS. Aplikace může běžet jak v linuxovém, tak i windowsovém prostředí. Jediné úskalí je, že na operačním systému Windows lze hostovat kontejnery jak v prostředí Linux, tak v prostředí Windows (za předpokladu že procesor umí virtualizaci a je povolen Hyper-V). Bohužel v případě, že je Docker nainstalován na linuxu, tak aplikaci lze spustit pouze v prostředí Linux (Ramos Apolinario, 2021).

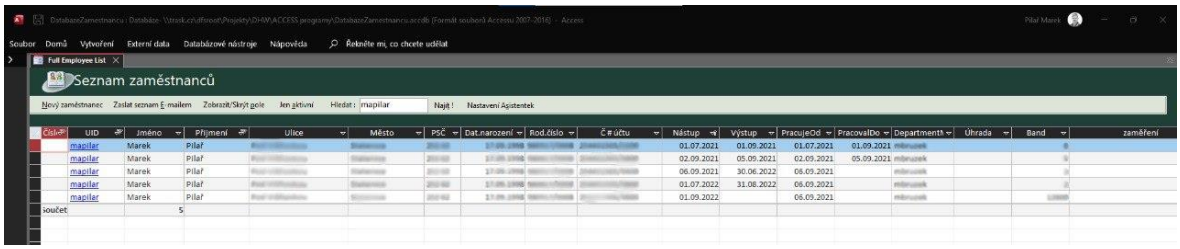
Celkově Docker pomáhá vývojářům a správcům IT snižovat náklady, zrychlovat nasazování aplikací, zvyšovat produktivitu a zlepšovat kvalitu aplikací, což je důvod, proč se stává stále populárnějším nástrojem pro nasazování a správu aplikací (Ramos Apolinario, 2021).

4 Vlastní práce

4.1 Analýza systému

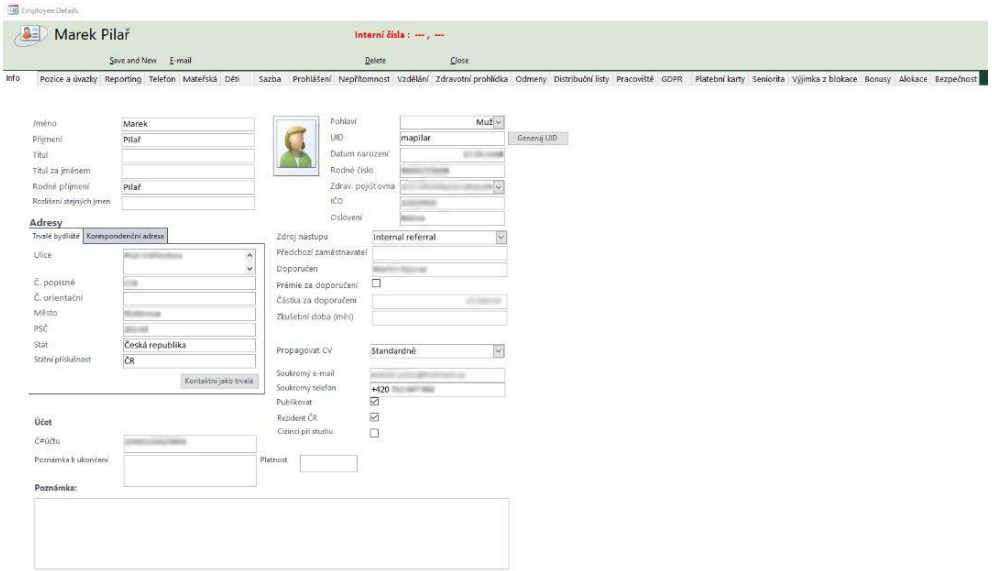
4.1.1 Představení aplikace

Aplikace slouží pro základní správu údajích o zaměstnancích a externích pracovnících firmy Trask Solutions. Jedná se o jeden z nejdůležitějších interních systému. Databáze obsahuje informace o více než 3 000 lidech, kteří historicky pro firmu pracovali nebo aktuálně pracují. Aplikace se skládá ze dvou hlavních formulářů. První je výběr konkrétní osoby ze seznamu zaměstnanců.



Obrázek 3 - Ukázka vyhledávání osob (zdroj: autor)

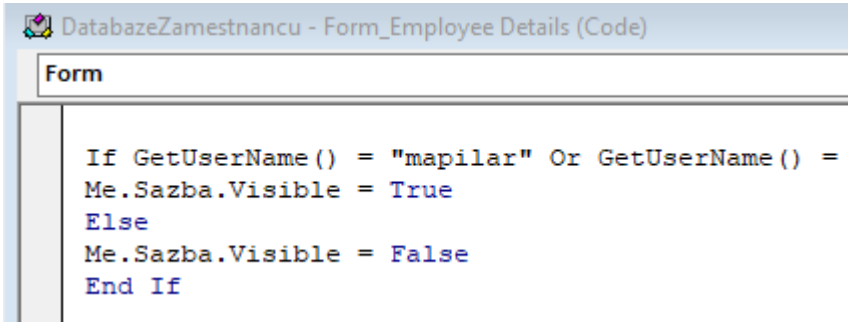
Na druhý formulář se lze prokliknout kliknutím na UID v předchozím obrázku. Druhý formulář již obsahuje podrobnější informace o zaměstnanci.



Obrázek 4 - Detailní přehled o osobě (zdroj: autor)

4.1.2 Současný stav aplikace

Systém pro HR agendu má tato firma z historických důvodů vytvořen za pomocí Access Forms. Ten slouží pouze jako front-end pro práci s daty na vzdáleném Microsoft SQL Serveru, tudíž nevyužívá základní vlastnost Microsoft Access (lokální databázový soubor, který obsahuje uložená data). Vzhled Access Forms je uložen jako soubor na sdíleném síťovém disku, který je umístěn na lokálním serverovém úložišti. Oprávnění s přístupem na tento soubor je řízený AD skupinou. Jedná se o první stupeň zabezpečení, kdy k tomuto souboru má přístup pouze určitá skupina lidí. Bohužel, tento typ zabezpečení se dá jednoduše obejít tím, že lze tento soubor přemístit do jiné složky, případně nahrát na přenosný USB disk nebo odeslat emailem. Dalším stupněm zabezpečení je pevně naspaný kód v tomto souboru, který obsahuj výčet lidí nebo skupin, který mají přístup k určitým sekcím. Tento způsob zabezpečení je velice nešťastným řešením, jelikož kód lze jednoduše nalézt, přepsat a udělit si přistup i do ostatních sekcí / modulů. Poslední způsob zabezpečení je až na databázové vrstvě. Na databázové vrstvě se nastavují oprávnění na čtení případně zápis informací. Pokud se někdo dostane přes první dvě vrstvy zabezpečení, tak tuto poslední vrstvu zabezpečení by již neměl mít možnost obejít. Nevýhodou je, že uživatelé musí mít přímý přístup k této databázi, což není z bezpečnostního hlediska úplně ideální řešení.



Obrázek 5 - Příklad kódu, který určuje přístup do jednotlivých sekcí (zdroj: autor)

4.1.3 Nedostatky aktuálního řešení

Některé nedostatky v zabezpečení jsem již popsal v předchozí sekci, nicméně nedostatků zůstává více. V této kapitole si tyto nedostatky shrneme.

- Zabezpečení
    - Z větší části se jedná o neúčinné a neefektivní typy zabezpečení.
    - Složitější proces udělování oprávnění (je potřeba udělit v několika vrstvách).
    - SQL Server je dostupný z celé firemní sítě.
- Výkon a ovladatelnost

- Při práci z domova je potřeba využívat VPN. Při komunikaci se serverem za použití VPN ale dochází k delší odezvě a Access se často zasekává a padá.
- Nelze využívat na mobilních telefonech a tabletech.
- Lze používat pouze na OS Windows.
- Velká limitace UI a UX. Lze používat pouze prvky / komponenty které Microsoft Access nabízí.

4.1.4 Požadavky na nový systém

Při tvorbě konceptu nového systému je zapotřebí zjistit veškeré možné požadavky od všech dotčených osob a skupin, kterých se týká užívání případně chod nového systému. Tyto požadavky následně vyhodnotím a pokusím se je zapracovat do nové aplikace. Od určitých skupin lidí se mi povedlo sehnat tyto požadavky.

1. HR a veškeří uživatelé tohoto systému
    - Zlepšení výkonosti při používání z domova. HR tento systém často využívá z Home office, tudíž by jim to ušetřilo spoustu času a nervů.
    - Veškerou HR agendu dát do jednoho software. V současnosti je něco jako webová stránka a něco jako jednotlivé Access programy.
    - Podporu mobilních telefonů a tabletů (případně i jiné OS).
    - Udělat aplikace v podobném UX, aby se uživatelé nemusely přeškolovat, respektive zvykat na úplně jiné UX a UI.
2. Interní IT
    - Podrobně popsat potřebnou infrastrukturu nové aplikace (kde aplikace funguje, co je potřeba zálohovat, způsob nastavování oprávnění).
    - V ideálním případě využít kontejnerizaci a aplikací mít jako škálovatelný Docker kontejner.
3. Interní reporting
    - Předat úpravy v datovém model.
    - Vytvořit servisní účet pro práci s daty na databázovém serveru.
    - V rámci modifikace datového modelu sjednotit konvenci pojmenovávání objektů a atributů.
4. Managment firmy
    - Stanovit předpokládaný čas potřebný na dodání nové aplikace.
    - Zjistit náklady na chod aplikace.

- Zjistit, zda bude za potřebí zakoupit nějaké licence, případně další náklady spojené s realizací aplikace.
- Vývoj pouze v technologiích, které firma využívá tudíž v .NET.

4.2 Výběr vhodných technologií

V rámci zadání od společnosti bylo definováno, že aplikace musí být buď v jazyce Java nebo c#. Při výběru vývojového jazyka jsem vycházel ze skutečnosti, že původní řešení je uděláno na technologiích od firmy Microsoft, proto bude jednodušší novou aplikaci vytvářet v jazyce C#, potažmo v .NET. Mezi hlavní výhody patří především:

- jednodušší přepis maker a funkcí ve visual basicu.
- Lepší kompatibilita s databázovou vrstvou (Microsoft SQL Server) pomocí entity framework.
- Podpora Windows autentifikace.

Dále je za potřebí vybrat architekturu pro nový projekt. Za pomocí .NET lze vytvořit několik typů aplikací jako např:

- Server only řešení:
  - ASP .NET MVC nebo Razor
  - ASP .NET Blazor (server based)
- REST API + SPA (Single-page application)
  - Rest API (ASP .NET) + SPA (java scriptový framework typu Angular nebo React)
  - Rest API (ASP .NET) + SPA (.NET Blazor za pomocí webassembly WASM)

Mezi SPA a klasickou server based aplikací je řada rozdílů.

1. Rozdíly v architektuře a komunikaci
   - SPA aplikace funguje tak, že si uživatel stáhne celou stránku a funkcionality pouze jednou a následně se dynamicky vykresluje obsah stránky bez opakovaného načítaná stránky. Výměna dat funguje díky REST API, kdy aplikace asynchronně předává požadavky serveru. Výhodou tedy je, že grafické rozhraní se generuje na straně uživatele, server odpovídá pouze na požadavky klienta. To snižuje datový provoz a zátěž serveru.

- Model MVC se skládá ze tří samostatných částí: model, pohled a kontroler. Uživatel si vždy stahuje kód stránky, na které se aktuálně nachází. Tato stránka vždy vzniká na základě požadavků od klienta, který odešle na server a kontroler ho začne zpracovávat. Kontroler následně sestaví datový model a tento datový model dosadí do pohledu. Následně vznikne webová stránka, která se odešle uživateli. Nevýhodou tedy je že uživatel dostává při interakci vždy nový kód stránky (opakovaný reload celé stránky). Výhodou je, že logika aplikace je obvykle oddělena od pohledů, tudíž pohledy lze poměrně jednoduše měnit, aniž by bylo potřeba zasahovat do logiky aplikace.
- .NET Blazor (server based) je hybrid mezi SPA a MVC. Aplikace se chová jako SPA, ale na pozadí pomocí SingnalR (websocet) komunikuje neustále se serverem. Každá interakce, kterou uživatel udělá, se odesílá na server, kde dojde k přegenerování pohledu a následné změny odesílá zpět klientovy. U tohoto principu je nevýhoda, že je aplikace závislá na dobré konektivitě k internetu / serveru. Při špatné konektivitě může docházet ke zpomalení interakce mezi uživatelem a serverem, dokonce může dojít k přetržení websocketu, což má za následek restart aplikace, díky čemuž může uživatel přijít o rozpracovaná data.

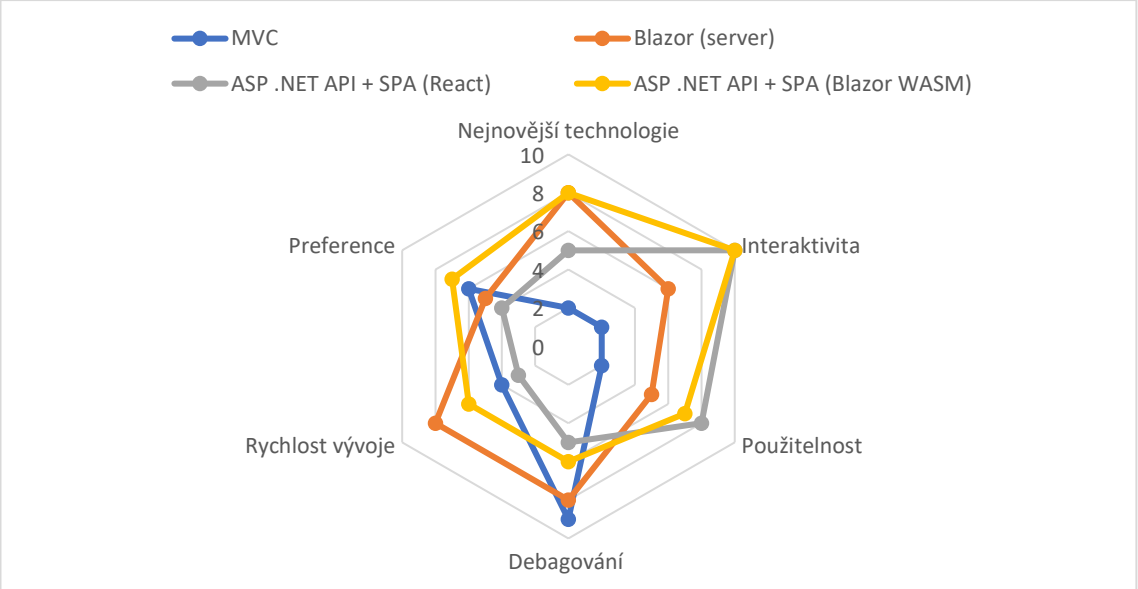
2. Interaktivita
    - SPA typu React nabízí vyšší míru interaktivity a uživatelského komfortu díky rychlejšímu a plynulému zobrazování dat bez nutnosti obnovování celé stránky.
    - MVC vyžaduje při každé interakci obnovu celé stránky
3. Vývoj a debuggování
    - MVC může být jednodušší na vývoj a debuggování, protože každá část aplikace (model, pohled, kontroler) má svou vlastní roli a je snadné identifikovat problémy, zatímco SPA typu React může být komplexnější a náročnější na vývoj a debuggování, zejména když se jedná o složité interaktivní aplikace.
5. Použití
    - Obecně platí, že SPA typu React se hodí pro interaktivní a rychle reagující aplikace, zatímco Classic MVC se hodí pro jednodušší a méně interaktivní aplikace.

Informace z porovnávání jednotlivých aspektů jsem ohodnotil na číselné škále a lze je interpretovat tabulkou níže.

Tabulka 2 - Ohodnocení jednotlivých aspektů u daných technologií (zdroj: autor)

	Aktuálnost technologie	Interaktivita	Použitelnost	Debuggování	Rychlost vývoje	Osobní Preference
MVC	2	2	2	9	4	6
Blazor (server)	8	6	5	8	8	5
ASP .NET API + SPA (React)	5	10	8	5	3	4
ASP .NET API + SPA (Blazor WASM)	8	10	7	6	6	7

Na základě této tabulky lze vytvořit grafické zobrazení těchto aspektů viz. níže.



Graf 1 - Grafické znázornění výběru vhodné technologie (zdroj: autor)

Vhodná architektura byla vybrána zhodnocením jednotlivých aspektů dané technologie a následným posouzením vhodnosti použití na tento konkrétní problém. V následujících odstavcích je zároveň vysvětleno, z jakého důvodu danou technologii nelze v tomto případě použít.

1. MVC

   Původní aplikace byly primárně určeny na zadávání a editování dat. Při těchto úkonech je v rámci aplikace prováděno mnoho interakcí. Architektura typu MVC je zároveň poměrně zastaralá a většina nových aplikací je tvořena pomocí přístupu SPA. Z těchto

poznatků můžeme usoudit, že aplikaci typu MVC není vhodná k řešení tohoto problému.

2. Blazor (server based)

   U tohoto přístupu je velká nevýhoda, že při přerušení spojení se aplikace restartuje a neuložená data se ztratí. To je v případě této aplikace dost nepraktické, a mohlo by to uživatelům značně komplikovat práci

3. SPA (Blazor WASM nebo java scriptový framework)

   Zde je zásadní rozdíl v tom, zda aplikaci psát celou jako monolit v jazyce C#, a to za pomocí webassembly anebo zda mít front-endovou aplikaci psanou za pomocí JavaScript / TypeScript.

Pro implementaci jsem si vybral Blazor WASM a to na základě těchto bodů

- Novější technologie než Angular / React.
- Funguje díky webassembly, což je technologie, která má do budoucna velký potenciál rozvoje. Tudíž je vhodná pro budoucí rozvoj aplikace.
- Má mnoho knihoven s komponenty, které jsou dostupné i pro React / Angular.
- Server side a Client side se píše ve stejném jazyku.
- Rychlejší vývoj, protože stačí znát pouze jeden programovací jazyk.
- Při programování se dají využít funkcionality, které má pouze .NET například SignalR. Což je limitace pro java script, který tyto funkcionality nemá.

4.3 Back end

4.3.1 Autentifikace

Autentifikace je řešena za pomocí Azure Active Directory (AAD). Jelikož je aplikace rozdělena na dvě části (server a klient), tak je potřeba vyřešit autentifikaci na obou stranách aplikace. Tudíž je za potřebí zaregistrovat v AAD dvě nové aplikace.

První zaregistrovanou aplikaci, která má název *HRM – Server* používá serverová část. V této registraci je potřeba provést několik úkonů, které jsou vyžadovány pro zamýšlený chod aplikace:

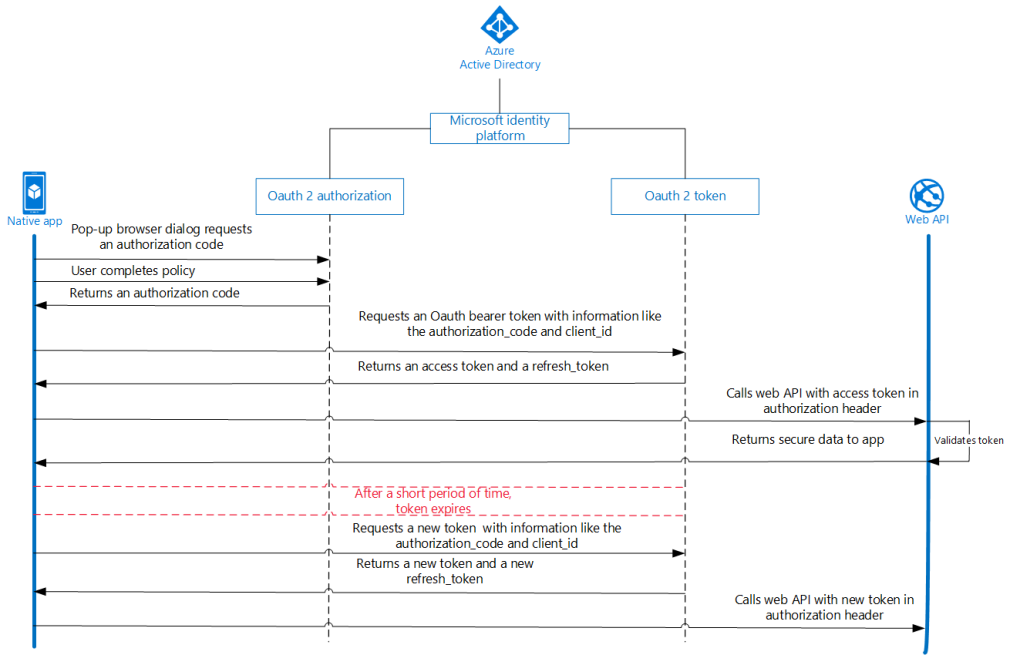
- Nastavení rolí

Na základě těchto rolí je řízen přístup do jednotlivých sekcí v aplikaci. Tudíž mezi výčtu rolí je např. Admin, Basic profil read, Advanced profile read, Mobile Admin

- Vypublikovat nový API scope
  Díky tomu se lze z klientské aplikace připojit na nově vytvořené rest API. Tento API scope lze pak přidat i do dalších aplikací, které díky tomu mohou mít přístup k této aplikaci.
- Přidělit oprávnění na další API
  Serverová část může využívat i další REST API např. Microsoft Graph. V této API lze využívat například endpoint
  GroupMember.Read.All, díky kterému lze zjistit v jakých AAD skupinách se daný uživatel nachází. Nebo lze použít User.Read.All, díky němuž lze zjistit Managera nebo oddělení určitého zaměstnance

Registrace klientské aplikace potřebuje taktéž nějaké specifické nastavení

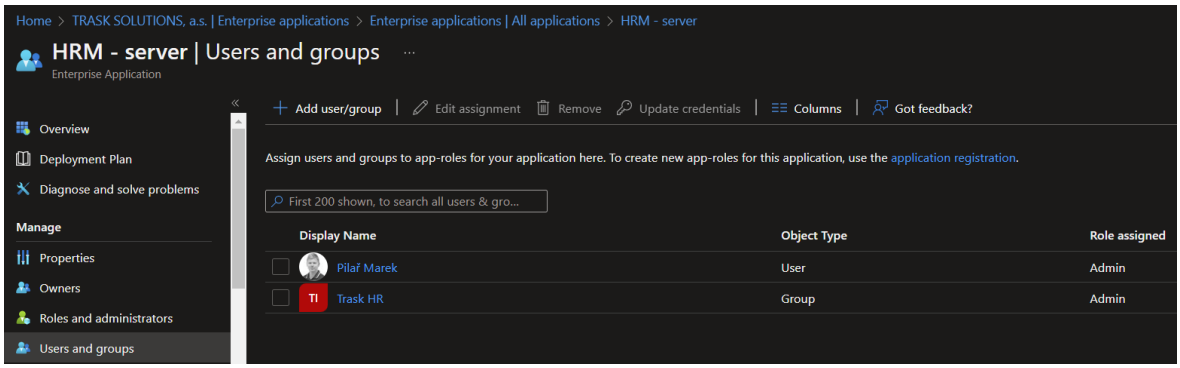
- Nastavení callback URL
  Při přihlašování je uživatel z aplikace přesměrován na přihlašovací stránku Microsoftu. Po vyplnění validních přihlašovacích údajů je uživatel přihlášen a přesměrován zpět do aplikace na tzv. call back URL. V mém případě je to https://hrm.trask.cz/authentication/login-callback. Na této adrese se předá autorizační kód a s tímto autorizačním kódem se následně žádá o tzv Bearer token.
- Přidělit oprávnění na další API
- Přidělení oprávnění k využívání serverové rest API (krok výše: Vypublikovat nový API scope). Případně dalších API jako třeba Microsoft Graph.



Obrázek 6 – Diagram znázorňující proces autentifikace (zdroj:)

4.3.2 Autorizace

Autorizace je prováděna na základě rolí, které byly nastaveny v Azure AD. Ke každé této roli lze přiřadit jednotlivé uživatele nebo skupinu uživatelů. Taktéž pomocí Azure AD v sekci Enterprise



Obrázek 7 - Ukázka oprávnění (zdroj: autor)

Aplikace je rozdělena na několik sekcí. Každá tato sekce, má dvě role např PersonalInformation.Read a PersonalInformation.Write. Tyto role se nastavují pomocí Azure AD viz předchozí obrázek číslo 7. Přehled plánovaného nastavení oprávnění do jednotlivých sekcí ukazuje tabulka níže.

Tabulka 3- Přehled oprávnění do jednotlivých sekcí (zdroj: autor)

Section	Group of users				
	HR	Mzdy	Compliance	Car	Phone
Basic informations	R	R/W	R	R	R
Personal informations	R	R/W			
Contract	R	R/W			
Vacation	R	R/W			
Maternity	R	R/W			
Education	R	R/W			
Rate		R/W			
Security	R/W	R/W	R/W		
Car assignment	R	R		R/W	
Phone assignment	R	R			R/W

4.3.3 Audit změn (log)

Implementace této vlastnosti je o dost jednodušší, když je v aplikaci aktivně využíván entity framework. Díky tomu se dá poměrně jednoduše vytvořit databázový audit. První krok je pomocí dědičnosti přidat funkcionalitu pro auditování, kdy se do určené tabulky budou zaznamenávat veškeré změny ve všech entitách. Tyto změny je vhodné uchovávat ve formátu JSON. Tabulky / třída pro evidenci auditu má takovouto strukturu (Murugan, 2022).

```
public class Audit
{
    public int Id { get; set; }
    public string UserId { get; set; }
    public string Type { get; set; }
    public string TableName { get; set; }
    public DateTime DateTime { get; set; }
    public string OldValues { get; set; }
    public string NewValues { get; set; }
    public string AffectedColumns { get; set; }
    public string PrimaryKey { get; set; }
}
```

Obrázek 8 - Ukázka třídy / tabulky obsahující audit (zdroj: Murugan, 2022)

Syntaxe pro volání změn v modelu je následující:

```
var oldEmployee = await _contextHRM. Employee.FindAsync(id);
_contextHRM.Entry(oldEmployee).CurrentValues.SetValues(newEmployee);
var result = await _contextHRM.SaveChangesAsync();
```

Obrázek 9 - Příklad volání změn v DB včetně logu (zdroj: Murugan, 2022)

Tato metoda porovná hodnoty v databází s hodnotami, které obdrželo REST API a rozdílné hodnoty v databázi upraví a zároveň se o této úpravě vytvoří záznam v auditové tabulce (Murugan, 2022).

4.3.4 Databázová vrstva

Veškerá data, která aplikace využívá, jsou uložena na Microsoft SQL serveru. Aktuálně datový model obsahuje 91 tabulek. Tabulky lze rozřadit do dvou základních skupin a to

- Číselníky (References)
  číselníky mají v původním datovém modelu před názvem napsáno ref (refTypUvazku). Což je dobrý způsob, jak je rozlišit od běžných tabulek s daty.

  Tyto tabulky jsou obvykle statické, a jejich datový obsah se moc často nemění, ale je dobré mít nějakou administrátorskou část na práci s těmito číselníky. Případně lze tyto číselníky řídit centrálně pomocí nějakého RDM systému.

  Tyto číselníky jsou obvykle využívány v dropdown menu.

- Ostatní datové tabulky

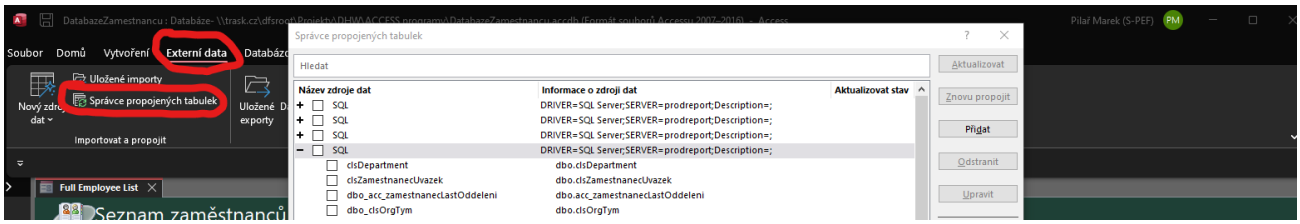
Při analýze bylo zjištěno několik zásadních nedostatků

- Špatný primární klíč
  Většina tabulek je napojena na osobu. Primární klíč OsobaID je číselný datový typ. Problémem je, že některé tabulky využívají jako cizí klíč neboli referenční hodnotu k této tabulce hodnotu sloupce Login. Jedná se taktéž o unikání hodnotu v tabulce osoba. Bohužel díky tomu nelze login uživatele měnit a když dojde ke změně příjmení z pravidla u žen po uzavření sňatku, tak tento login nemůže být změněn, jelikož je v některých tabulkách využíván jako cizí klíč. Tento problém vznikl historickým vývojem, kdy si tým data warehouse usnadňoval práci, jelikož pro potřeby reportingu se využívá jako primární klíč login.

  Je potřeba, aby všechny tabulky, které mají cizí klíč tabulky Osoba, využívali sloupec OsobaID nikoliv Login.

- Nevyužívané tabulky

  Tabulky, které jsou nevyužívané většinou, bývají označeny jako tmp. Soupis tabulek, které využívá původní Microrost Access lze získat přímo z aplikace za pomocí správce propojených tabulek.



Obrázek 10 – Přehled použitých tabelek (zdroj: autor)

Tabulky, které se v tomto soupisu nenachází, nejsou zapotřebí pro chod starého systému, proto je možné je odstranit, nicméně pro jistotu je potřeba vytvořit zálohu pro případ, kdyby byla potřeba některá data obnovit.

Tabulky, které se v tomto soupisu nenachází, nejsou zapotřebí pro chod původního systému, se mohou odstranit. Pro jistotu je potřeba vytvořit zálohy pro případ, kdyby byla nutnost některá data obnovit.

- Neexistující konvence pro pojmenovávání

  Dalším problémem je, že každá tabulka se jmenuje úplně jinak. Některé jsou v jednotném čísle, některé zase v množném. Některé jsou anglicky a některé jsou česky.

  Nové nastavení jmenné konvence:

  - Používat podstatná jména v množném čísle (Osoby, Školy atd.)
  - Před název tabulky, která slouží jako číselník dát prefix ref (např refTypUvazku)
  - Staré tabulky nechat v českém jazyce, ale nové tabulky a případné úpravy dělat už pouze v anglickém jazyce z důvodu ulehčení práce při nastavení mapování na REST API
  - Primární klíč musí obsahovat název tabulky + ID například OsobaID
  - Slova v názvu rozlišovat velkým písmenem např. TypUvazku
  - Nepoužívat diakritiku

- Některé tabulky / sloupce využívají diakritiku například sloupec: Příjmení

Tyto tabulky / sloupce je potřeba co nejdříve přejmenovat, jelikož je nepřípustné, aby diakritiku obsahovali. EF a další části .NET mývají problém s diakritikou a může to vyústit v neošetřené chyby v aplikaci.

Veškeré změny v datovém modelu je potřeba podrobně zapsat a zdokumentovat. Následně je předat dalším dotčeným týmům například tým DWH, který si z této produkční databáze sbírá data.

4.4 Front-End

Front-endová část aplikace je implementována za pomocí .NET Blazor (WASM). Jedná se o nový způsob řešení SPA aplikací, které fungují díky technologii WebAssembly. WebAssembly je k dispozici na většině aktuálních prohlížečů. Podle webové stránky Can WebAssembly | Can I use (2023) tato technologie pokrývá 96,16 % veškerých uživatelů, tudíž jí můžeme brát jako vhodnou pro tyto účely.

Při implementaci je využívána knihovna s komponenty s názvem Radzen Blazor. Jedná se o Open Source knihovnu, jejíž zdrojový kód je k nalezení na Radzenhq · GitHub (2023) , a které použití je zároveň zdarma. Radzen je založen na Material design a FluentUI. Celkově tato knihovna ušetří mnoho času. Její výhodou je, že lze z těchto jednoduchých komponentů tvořit složitější komponenty jako jsou třeba celé formuláře.

Autorizace a autentifikace je implementována za pomocí Azure AD viz kapitoly výše. U každé sekce a jednotlivých akcí je definován seznam rolí, které jsou oprávněny tyto akce provádět.

Každá sekce je implementována jako jednotlivá service, které komunikují se serverovým REST API. Příklad service, které jsou v aplikaci vytvořeny jsou: employee, education, work place.

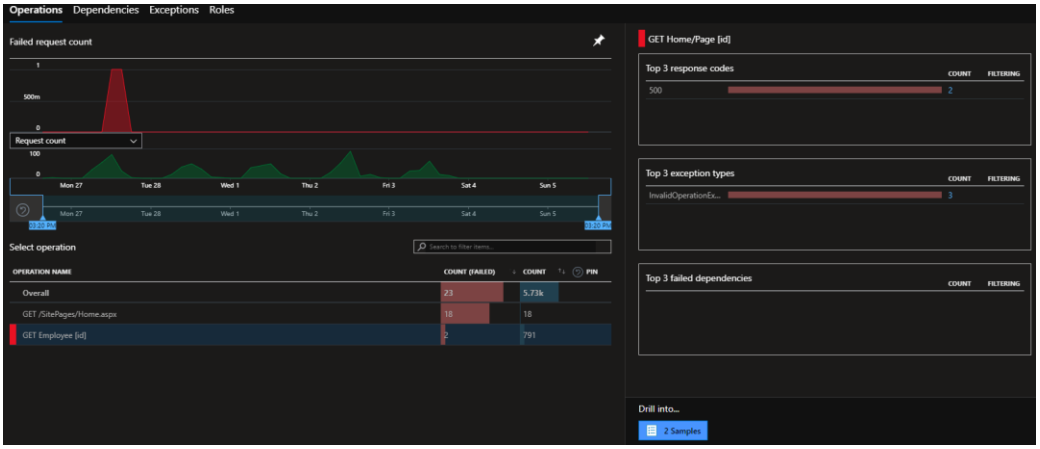
Výhoda Blazoru je ta, že lze využívat komponenty jak z knihoven, tak si tvořit vlastní komponenty. Tyto komponenty lze skládat, a tak se jedna komponenta může skládat z více komponent. Jako příklad bych uvedl přihlašovací formulář, který se skládá obvykle z komponenty pro heslo, uživatelské jméno a tlačítka pro odeslání. Celá tato komponenta může být součástí jiné komponenty nebo jakékoliv stránky. Zároveň si komponenty umí předávat parametry.

```
<div class="row">
    <div class="col-md-12">
        <RadzenDataGrid @ref="grid" PageSize="10"
            AllowFiltering="true"
            FilterMode="FilterMode.SimpleWithMenu"
            FilterCaseSensitivity="@FilterCaseSensitivity.CaseInsensitive"
            AllowPaging="true"
            AllowSorting="true"
            Data="@employees" TItem="EmployeeDTO" >
            <Columns>
                <RadzenDataGridColumn TItem="EmployeeDTO" Property="Login" Title="Login">
                </RadzenDataGridColumn>
                <RadzenDataGridColumn TItem="EmployeeDTO" Property="Name" Title="Name">
                </RadzenDataGridColumn>
                <RadzenDataGridColumn TItem="EmployeeDTO" Property="Surname" Title="Surname">
                </RadzenDataGridColumn>
                <RadzenDataGridColumn TItem="EmployeeDTO" Filterable="false" Sortable="false" TextAlign="TextAlign.Center">
                    <Template Context="employee">
                        <RadzenButton ButtonStyle="ButtonStyle.Warning" Icon="phone_enabled" Size="ButtonSize.Medium"
                            Shade="Shade.Lighter" Variant="Variant.Flat" Click=@(args => RedirectClick(args, employee)) />
                    </Template>
                </RadzenDataGridColumn>
            </Columns>
        </RadzenDataGrid>
    </div>
</div>
```

Obrázek 11– Ukázka využití komponent (zdroj: autor)

4.5 Monitoring

Monitoring aplikace je dosažen za pomocí služby Azure Application Insights. Tato aplikace sbírá data ze všech operací, které jsou v aplikaci vykonány. Díky této aplikaci je přehledně vidět její vytížení a doba odezvy na jednotlivé požadavky, počet jednotlivých požadavků. Zároveň lze v této aplikaci nalézt veškeré neošetřené chyby a díky snapshotu aplikace v daný okamžik lze tuto chybu debugovat s daty, které tuto chybu způsobili.



Obrázek 12 – Ukázka zachycených chyb (zdroj: autor)

4.6 Dokumentace

4.6.1 Verzování

Zdrojový kód celé aplikace je verzovaný v GIT repositáři na Azure DevOps. Repositář je rozdělený na dvě hlavní větve master a dev. Součástí Azure DevOps je také wiki, která zachycuje dokumentaci této aplikace. Dále tato služba umožňuje evidovat jednotlivé úkoly

(bug, feature, review, documentation). Tyto úkoly lze přenést z Application Insights tzn. pokud přibydou telemetrická data s nějakou chybou, tak se tato chyba dá přenést jako úkol k zapracování včetně telemetrických dat, které tu chybu popisují. Následně lze chybu opravit a commit, který tuto chybu opravuje lze k tomu úkolu přiřadit. Vznikne celé workflow pro řešení chyb, které lze následně jednoduše dohledat.

4.6.2 Nasazování

Aplikace může být nasazena jako docker kontejner nebo jako spustitelná stránka na webovém serveru IIS. V současnosti stránka běží na serveru IIS s operačním systémem Windows server 2022. Pro chod aplikace je za potřebí .NET 6 a lze stáhnout celé SDK nebo stačí pouze Hosting Bundle.

Současně je nasazování aplikace plně automatizováno. Proto když přijde schválený pull request do větve master, tak se spustí automatický deployment (CI / CD). V prvním kroku dojde k buildu aplikace a provedení testů. V případě že je předchozí krok úspěšný, tak dojde k nahrání aplikace na produkční server.

4.6.3 Zálohování

Aplikace využívá dvě databáze. Hlavní databáze HRM, ve které je uložena většina dat, je zálohována 2x denně. Po jednom týdnu se uchovávají zálohy jen 1x denně a po měsíci pouze 1x měsíčně. Odpovědnost za zálohování těchto databází, dle stanoveného rozvrhu přebírá interní IT podpora.

Druhá databáze DWH slouží pouze pro pár číselníků, kde je pouze read oprávnění. K žádnému zápisu dat nedochází, tudíž zálohování této databáze není pro chod aplikace vyžadováno a přechází na odpovědnost vlastníka databáze.

4.6.4 Cloud vs. on-premises

Na základě potřeb firmy bylo rozhodnuto že aplikace nebude využívat cloud z důvodů:

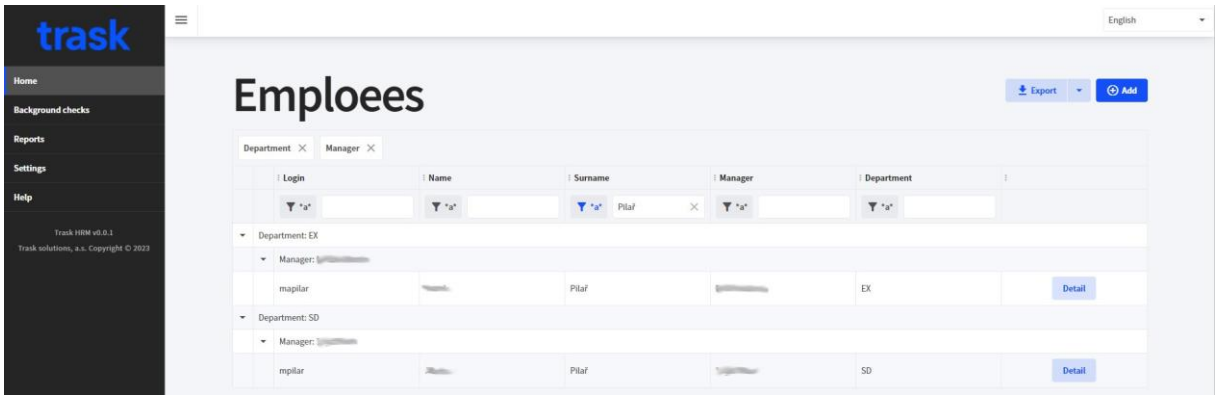
- strach o zabezpečení těchto citlivých dat
- již vybudovaná infrastruktura v rámci firemní sítě, která splňuje požadavky na chod aplikace
- díky již existující infrastruktuře dochází k nižším provozním nákladům
- aplikace je koncipována jako neveřejná, tudíž je dostupná pouze z interní sítě

5 Výsledky a diskuse

Na základě poznatků z teoretické části byly vybrány vhodné technologie pro vytvoření nové aplikace. Výčet zvolených technologií je následující:

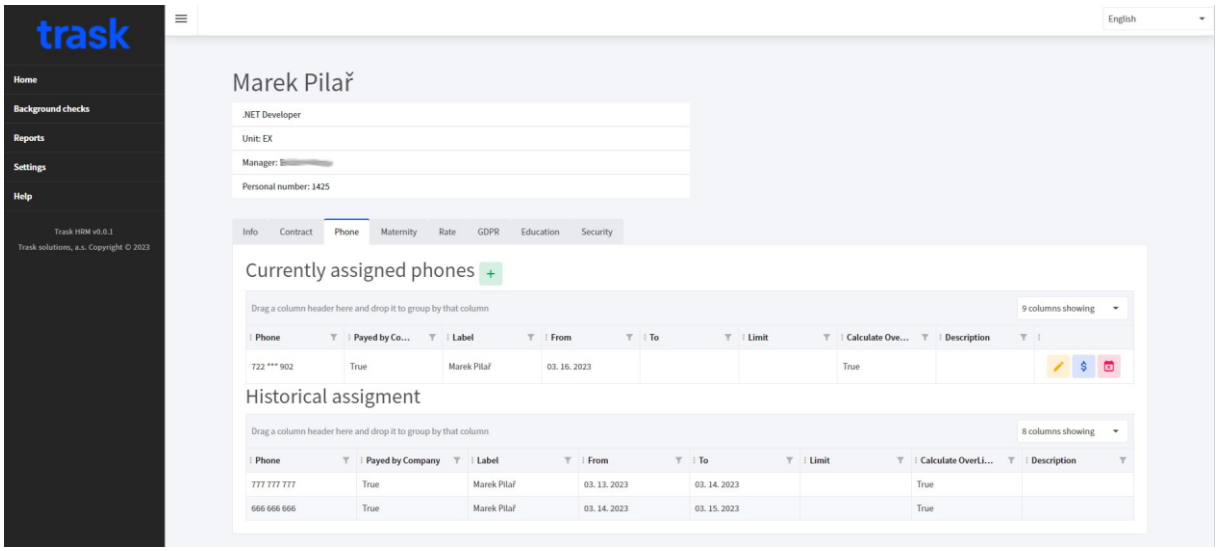
- Microsoft SQL Server
- REST API (ASP .NET, Entity Framework)
- klientské aplikaci (ASP .NET Blazor WASM + knihovna Radzen)
- Azure Active Directory
- Docker.

Níže je uvedeno několik snímků obrazovky z nové aplikace. Jako první je ukázka vyhledávání, která je obdobná jako v původní aplikaci (Obrázek 3)



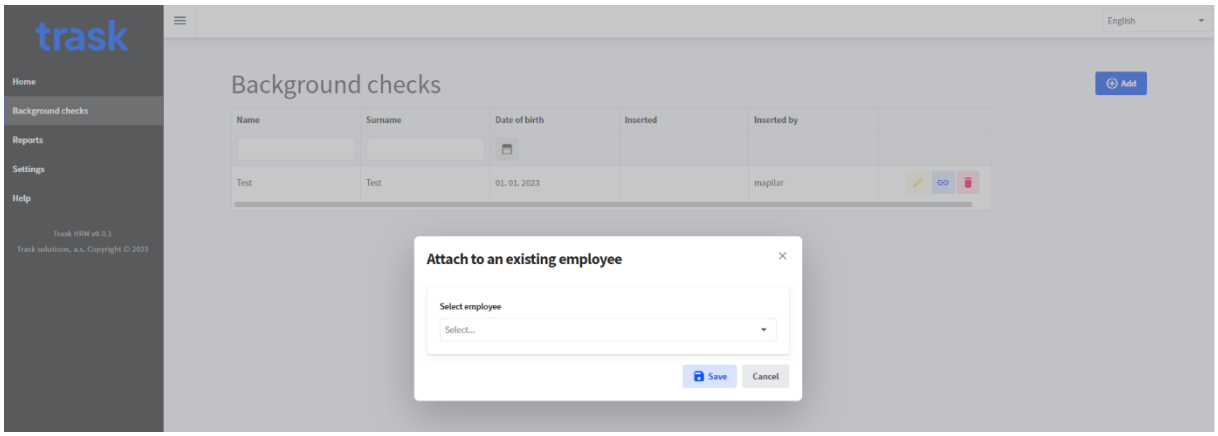
Obrázek 13- Ukázka vyhledávání osob v nové aplikaci (zdroj: autor)

Po přechodu na detail se uživateli zobrazí následující okno, které v původní aplikaci odpovídá obrázku číslo 4.



Obrázek 14- Detailní přehled v nové aplikaci - přiřazení mobilních sužeb (zdroj: autor)

Většina dat je nově zadávána pomocí tzv. dialogových oken. Příklad tohoto okna je na dalším obrázku.



Obrázek 15- Ukázka dialogového okna v nové aplikaci (zdroj: autor)

Výsledná aplikace je v akceptačním procesu a je plně funkční. V současnosti nebyly zmigrovány veškeré moduly, tudíž na uvedených obrázcích je možné spatřit, že některé funkcionality zde stále ještě chybí. Tyto drobné nedodělky vzniknuly nedostatkem časových kapacit. Zároveň nemají vliv na celkový chod aplikace a byly přesunuty do nadcházejících verzí aplikace.

6 Závěr

V rámci této práce bylo představeno mnoho moderních technologií, které jsou aktuálně využívány předními technologickými společnostmi, které se zabývají tvorbou webových aplikací. Z těchto technologií bych chtěl vyzdvihnout WebAssembly, jelikož se jedná o poměrně novou technologii s velkým potenciálem do budoucna.

Praktická část této práce může sloužit jako metodický postup pro migrace obdobných aplikací do webového rozhraní. Základem úspěšné migrace je porozumění původní aplikaci a podrobné prostudování technického řešení a následném návrhu nového řešení.

Na základě této analýzy se volí soubor technologií, který je nejvíce vhodný s ohledem na původní řešení. Většinou je zapotřebí udělat výraznější změny v celé aplikaci. Ve fungování celé aplikace v tomto případě byly výrazné změny v datovém modelu, přesunu aplikační logiky na jedno místo a systému zabezpečení. Dále byla aplikace rozšířena o několik dalších vlastností, jako je například auditování a exporty dat do formátu .csv nebo .xlsx.

K aplikaci byl vytvořen jednoduchý návod, který vysvětluje princip nasazení aplikace, mapuje závislosti aplikace a stanovuje nutné úkony pro korektní chod aplikace.

Závěrem bych chtěl říci, že tato práce mi ukázala obrovský rozdíl mezi tvorbou úplně nové aplikaci a upgradem, migrací případně integrací nějaké již stávající aplikace. Obecně lze říci, že při tvorbě a úpravě aplikací je důležité mít detailní znalost o fungování původní aplikace. Bez této znalosti je obtížné vytvářet nové prvky a segmenty, které budou zachovávat původní funkčnost aplikace a řešit problematické oblasti. Dále lze konstatovat, že tvorba nové aplikace je jednodušší, jelikož není potřeba dělat podrobnou analýzu původní aplikace a není třeba mapovat závislosti aplikace.

8 Seznam použitých zdrojů

.NET Platform · GitHub [online], 2023. California: Github [cit. 2023-03-15]. Dostupné z: https://github.com/dotnet

JAPIKSE, Philip, Kevin GROSSNICKLAUS a Ben DEWEY, 2017. *Building Web Applications with Visual Studio 2017: Using .NET Core and Modern JavaScript Frameworks*. 1st ed. Berkeley, CA: Apress. ISBN 978-1-4842-2478-6.

Microsoft Access [online], 2022. San Francisco: Wikipedia [cit. 2022-11-06]. Dostupné z: https://en.wikipedia.org/wiki/Microsoft_Access

MICROSOFT CORPORATION, 2022. Blazor | Build client web apps with C# | .NET. In: *Microsoft.com* [online]. Redmond: Microsoft [cit. 2022-11-17]. Dostupné z: https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor

MORGADO, Flavio, 2021. *Introducing Microsoft Access using macro programming techniques: an introduction to desktop database development by example*. 1st ed. New York, NY: Apress. ISBN 978-148-4265-550.

MURUGAN, Mukesh, 2022. Audit Trail Implementation in ASP.NET Core with Entity Framework Core. In: *Audit Trail Implementation in ASP.NET Core with Entity Framework Core* [online]. Kerala [cit. 2023-03-13]. Dostupné z: https://codewithmukesh.com/blog/audit-trail-implementation-in-aspnet-core/

PRICE, Mark J., 2021. *C# 10 and .NET 6 - modern cross-platform development: build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code*. Sixth edition. Birmingham: Packt. ISBN 978-1-80107-736-1.

Radzenhq · GitHub [online], 2023. California: Github [cit. 2023-03-15]. Dostupné z: https://github.com/radzenhq/radzen-blazor

RAMOS APOLINARIO, Vinicius, 2021. *Windows Containers for IT Pros: transitioning existing applications to containers for on-premises, cloud, or hybrid*. 1st ed. New York: Apress. ISBN 978-1-4842-6686-1.

WebAssembly [online], 2023. San Francisco: Mozilla Corporation [cit. 2023-03-13]. Dostupné z: https://webassembly.org/

WebAssembly | Can I use [online], 2023. US: Alexis Deveria [cit. 2023-03-15]. Dostupné z: https://caniuse.com/wasm

9 Seznam obrázků, tabulek, grafů a zkratek

9.1 Seznam obrázků

9.2 Seznam tabulek

9.3 Seznam grafů

9.4 Seznam použitých zkratek

AAD	Azure Active Directory
AD	Active Driectory
BP	Bakalářská práce
CD	Continuous delivery
CI	Continuous integration
CRM	Customer relationship management – řízení vztahů s klienty
DWH	Data Warehouse
EF	Entity Framework
HR	Human Resources – lidské zdroje
HRM	Human resource management
IIS	Internet Information Services
JS	Java Script
MVC	Model
OS	Operační systém
SDK	Software development kit
SQL	Stuctured Query Language
URL	Uniform Resource Locator
VBA	Visual Basic for Applications
VPN	Virtual Private Network
WASM	Web assembly