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

Realizace komplexních reportů na platformě Microsoft .NET

Diplomová práce

Autor: Bc. Tomáš Falta
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. Mgr. Tomáš Kozel, Ph.D.

Hradec Králové Duben 2020

Prohlášení:

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

vlastnoruční podpis

V Hradci Králové dne 27.4.2020 Bc. Tomáš Falta

Poděkování:

Děkuji vedoucímu diplomové práce doc. Mgr. Tomáši Kozlovi, Ph.D. za metodické vedení práce.

Anotace

Cílem této diplomové práce je seznámení s webovými technologiemi platformy .NET z pohledu vytváření komplexních a uživatelsky přívětivých reportů. Práce je rozdělena do tří částí. První část práce se věnuje popisu platformy .NET, jejím dostupným technologiím pro vývoj webových aplikací a plánovanému budoucímu rozvoji. Dále jsou popsány možnosti vizualizace a reportování dat ve webových aplikacích společně s dalšími technologiemi společnosti Microsoft, které umožňují reporting dat a možnosti datové analýzy. V druhé části práce jsou představeny vybrané dostupné balíky znovupoužitelných komponent pro webové technologie platformy .NET, které slouží k analýze a vizualizaci dat. Balíky komponent jsou mezi sebou porovnány na základě stanovených požadavků pro jejich funkcionalitu. Poslední část práce je zaměřena na návrh a implementaci vlastní znovupoužitelné komponenty tabulkového reportu za pomocí technologie ASP.NET Core Blazor.

Annotation

Title: Realization of complex reports on Microsoft .NET platform

The aim of this master thesis is to introduce web technologies of the .NET platform which are used for the development of complex user-friendly reports. The thesis is divided into three sections. The first section is dedicated to the basic information about the .NET platform, available technologies for the development of web applications and planned future development of the platform. Furthermore, the possibilities of data visualization and reporting in the web application, together with further Microsoft technologies, which are used for data reporting and data analysis. In the following section of the thesis, the available packages of reusable components for web technologies of the .NET platform, are introduced. These components are used for data analysis and data visualization. The packages are compared based on the requirements of their functionalities. The last section is focused on the design and implementation of the custom reusable component of the data grid report via the technology of the ASP.NET Core Blazor.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

S velkým rozvojem informačních technologií a systémů se rozvíjí také požadavky na možnosti zpracování a analýzy dat. Informační systémy zahrnují nejen software, hardware, lidi, ale i procesy a činnosti, které umožňují sběr, zpracování a šíření dat. Data uložená v systému mají svůj smysl a při použití vhodných procesů a technologií je možné je převést na informaci. Kvalitní informace jsou nedílnou součástí všech společností, umožňují rychlé a přesné rozhodování i dynamické přizpůsobování změnám na trhu. V rámci systému musí být ve správné formě v přesný čas dostupné každému požadovanému uživateli. Tyto procesy zajišťují metody business intelligence. Business intelligence je sada teorií, metodologií, architektur a technik, které transformují nezpracovaná data do smysluplné a užitečné informace pro obchodní účely.

Jednou z posledních fází procesů v rámci business intelligence je zobrazení zpracovaných dat uživateli na úrovni, která mu umožní jejich převedení na informace, následnou analýzu a podpoří správná rozhodnutí. V závislosti na použitém nástroji a také na charakteristice a komplexnosti reportu lze nalézt různé koncepty předání a zobrazení dat. Všechny koncepty ale popisují koncové rozhraní, které bude použito k analýze informací. Informační systémy pro interakci uživatelů obvykle používají webové aplikace. Zároveň jednou ze základních požadovaných komponent informačních systémů jsou možnosti reportingu dat a na jejich vývoj je kladen velký důraz. Setkávají se zde požadavky na rychlost zpracování a zobrazení velkého množství dat, možnosti různorodého dynamického přizpůsobení při zachování uživatelské přívětivosti a přehlednosti. Na druhé straně stojí vlastnosti webových technologií, které tento vývoj komplikují a je nutné je vyřešit. Při návrhu a vývoji reportů je nutné zachovat přehlednost při zobrazení na prostorném řídícím panelu i na mobilním telefonu s malým rozlišením displeje a nízkým výkonem. Pro podporu správného rozhodování ve společnosti je poskytování informací v reálném čase nezbytné. S moderním přístupem k vývoji aplikací je zároveň kladen velký důraz na rychlost i efektivitu vývoje a znovupoužitelnost jednotlivých vyvinutých komponent aplikace. Jejich sdílení může probíhat v rámci daného systému, mezi různými projekty i komunitou.

Nezávislí vývojáři i velké společnosti nabízí balíky znovupoužitelných komponent, které při správném použití mohou zásadně zjednodušit vývoj různých částí informačních systémů. V rámci těchto balíků komponent jsou obvykle nabízeny i komponenty uživatelského rozhraní, které slouží pro zpracování a vizualizaci dat v informačních systémech.

2 Cíl práce

Cílem této diplomové práce je prozkoumat a popsat platformu .NET z pohledu vytváření komplexních a uživatelsky přívětivých reportů. Práce je primárně zaměřená na možnosti webových technologií na této platformě. V teoretické části práce budou popsány aktuálně dostupné technologie platformy .NET pro tvorbu webových aplikací a jejich budoucí vývoj. Dále bude představen termín business intelligence, který zastřešuje reporting dat. V rámci této kapitoly budou popsány možnosti reprezentace a vizualizace dat ve webovém prostředí umožňující přehledné, intuitivní, avšak komplexní zobrazení dat a jejich následnou analýzu a vyhodnocení. Další část práce představí vybrané balíky znovupoužitelných komponent různých výrobců určené pro webové technologie platformy .NET, které slouží k analýze a vizualizaci dat. Balíky komponent budou mezi sebou porovnány na základě stanovených požadavků pro jejich funkcionalitu. Hodnotící kritéria budou stanovena s ohledem na požadavky moderních webových aplikací a vývoje těchto aplikací.

V rámci praktické části práce bude vytvořena znovupoužitelná komponenta tabulkového reportu. Komponenta bude umožňovat základní i pokročilou funkcionalitu jako stránkování, řazení, filtrování a seskupování dat a nabídne také možnost vykreslení diagramů. Pro vývoj komponenty bude použita technologie Blazor platformy ASP.NET Core, který využívá nového webového standardu WebAssembly. Společně s komponentou budou představeny možnosti nové a rychle se rozvíjející technologie.

3 Platforma .NET

Platforma .NET (dot net) společnosti Microsoft v sobě zahrnuje technologie pro vývoj desktopových, mobilních a webových aplikací, 2D a 3D her pro stolní počítače, mobilní telefony nebo konzole a mnoho dalších. První verze byla vydána v roce 2002 a během více než 18letého vývoje v ní došlo k mnoha zásadním změnám. Nejdůležitější z nich budou v této kapitole popsány. Společnost Microsoft platformu neustále rozvíjí a v závislosti na aktuálních požadavcích a trendech přidává nové možnosti a technologie. Platforma nabízí například i technologie pro práci s prediktivními modely, algoritmy počítačového vidění, rozpoznávání hlasu nebo rozšiřitelnou platformu pro strojové učení [1].

3.1 .NET Framework

Microsoft .NET Framework je systém komponent určených k vývoji různorodých aplikací primárně pro operační systém Windows. Cílem frameworku bylo zjednodušení tvorby aplikací, zpřehlednění jejich zdrojových kódů a zjednodušení správy instalací na koncových zařízeních. V rámci platformy vzniklo mnoho různých technologií. Mezi ně patří například Windows Forms a Windows Presentation Foundation (WPF) pro tvorbu desktopových aplikací a bohatého uživatelského rozhraní. Mezi další podstatné technologie se řadí ASP.NET pro vývoj webových aplikací nebo integrovaný jazyk LINQ pro dotazování nad libovolnými daty. Samotný .NET Framework je složen z několika úrovní. Základní úroveň reprezentuje operační systém. Nad ním je implementován Common Language Runtime (dále jen CLR). Ten zajišťuje základní infrastrukturu frameworku. Libovolný programovací jazyk, který podporuje CLR, není kompilován přímo do nativního kód, ale do jazyka Common Intermediate Language (dále jen CIL). Díky tomu je umožněno použití různých programovacích jazyků i zjednodušení přenosu kódu mezi platformami. Další úrovně frameworku jsou implementovány jako hierarchicky uspořádané knihovny. Základní je Base Class Library, zkráceně BCL, která poskytuje množství funkcí pro ošetřování výjimek, lokalizaci a globalizaci, metody pro přístup k datovým úložištím nebo k síťové komunikaci a další. Další rozšiřující úroveň obsahuje knihovnu ADO.NET pro přístup k datům v relačních databázích a knihovny pro práci s daty ve formátu XML. V dalších úrovních frameworku se již nachází

knihovny pro vývoj specifických aplikací. Framework umožňuje použití jakéhokoliv programovacího jazyka kompatibilního s CLR. Mezi ně se řadí nejen jazyk C#, ale i F#, Visual Basic, JScript .NET či Python pomocí implementace IronPython [2] [3].

3.2 ASP.NET

ASP.NET (Active Server Pages) je platforma společnosti Microsoft určená k vývoji webových aplikaci. Předchůdcem ASP.NET byla technologie s názvem „Classic ASP“. Ta umožňovala používat skriptovací jazyk VBScript ve webových stránkách. Při použití Classic ASP se však v jednom souboru prolínal značkovací jazyk HTML a příkazy VBScript. Výsledkem byl nepřehledný kód a velmi omezená znovu použitelnost kódu i jednotlivých komponent [4].

V roce 2002 Microsoft představil v té době revoluční technologii s názvem Web Forms. Autoři technologie se pokusili co nejvíce odstínit webové prostředí tak, aby k vývoji webových aplikací nalákali i vývojáře pracující primárně na desktopových aplikacích. Nicméně webové aplikace jsou založené na bezstavovém protokolu Hypertext transfer protocol (dále jen HTTP). Pro implementaci „stavovosti“ byl ve Web Forms implementován mechanismus pojmenovaný ViewState. Jedná se o blok dat, který obsahuje aktuální stav aplikace a v každém požadavku a odpovědi je přenášen mezi klientem a serverem. Vývojáři umožňuje pracovat se stránkou bez znalosti jejího mechanismu vykreslování a překreslování. Technologie sama zajistí překreslení celé stránky či pouze jejích částí [4].

Mezi výhody této technologie lze bezpochyby zařadit její dospělost a množství dostupných komponent. Od představení v roce 2002 vzniklo nespočet návodů, literatury i hotových řešení. Technologie je podobně jako desktopové aplikace založena na komponentách. Jedná se o znovupoužitelné části stránek, které lze v návrhu stránek přenášet i pomocí akce drag-and-drop, zanořovat a používat napříč různými projekty. Jednou ze základních a často používaných komponent dostupných ve Web Forms je GridView (v českém překladu datová mřížka). Ta se používá pro zobrazení dat formátovaných do tabulky. Již ve výchozí implementaci nabízí rozsáhlou nabídku operací. Umožňuje stránkování, řazení,

filtrování dat, ale i řádkovou editaci, vkládání či odstraňování záznamů nebo rozsáhlé úpravy vzhledu celé tabulky, řádků, sloupců i jednotlivých buněk. Je snadno rozšiřitelná o další funkcionalitu, například o dynamické zobrazování a přesouvání sloupců tabulky. Komponenta je ve webových aplikacích využívajících technologii Web Forms často používána pro základní reporting dat a bude zmíněna i v následujících kapitolách [2] [4].

Přístup k vývoji desktopových aplikací je však velmi rozdílný a upozadění webového prostředí přineslo i některé nevýhody. Rychlý rozvoj webových technologií a frameworků ukázal i zásadní nevýhody technologie Web Forms. Prvním problémem se zejména ve větších aplikacích stává ViewState, přenášející aktuální stav stránky mezi klientem a serverem. Ten je technicky implementován jako skrytý HTML element ve stránce. Pokud stránka obsahuje větší množství komponent a zobrazených dat, může jeho velikost nabývat i několika megabajtů dat, které jsou s každým požadavkem přenášeny z klientského prohlížeče na server a zpět. Při připojení více klientů zásadním způsobem stoupá vytížení sítě a zvyšuje se požadavek na výkon serveru i klientského prohlížeče. Druhá nevýhoda technologie vyplývá z ViewState mechanismu, a to samotné upozadění bezstavovosti protokolu HTTP. Vývojáři často potřebují více kontroly nad přenosem, vykreslením i zpracováním dat, které je často možné dosáhnout pouze celkovou reimplementací části aplikace a v některých případech ani není možné. Mezi další nevýhody se řadí i omezené možnosti testování webových stránek. Technologie často vývojáře vedla k implementaci aplikační logiky přímo do webových formulářů, kam ale dle pravidel vícevrstvé architektury webových aplikací nepatří a velmi tak omezuje další rozvoj aplikací i již zmíněné testování [4].

Od první verze ASP.NET 1.0 vydané počátkem roku 2002 bylo vydáno mnoho dalších verzí, které postupně přinesly množství úprav a rozšíření. Web Forms jsou úzce svázané s .NET Frameworkem a jejich životnost díky tomu kopíruje životnost samotného .NET Frameworku, který je nadále vyvíjen [4]. Microsoft sice stále tuto technologii podporuje, nový vývoj ale od roku 2015 neprobíhá a není plánován [5]. Přesto se jedná o důležitou kapitolu vývoje webových aplikací na .NET platformě

a některé koncepty byly použity při návrhu novějších technologiích, které budou představeny v následujících kapitolách této práce.

3.3 ASP.NET MVC

Technologie ASP.NET MVC vznikla jako alternativa k technologii Web Forms. Svým základem se jedná o implementaci architektonického vzoru Model-View-Controller (dále jen MVC). Tento vzor se používá přibližně od roku 1970, kdy byl navržen pro organizaci prvních komponent grafického uživatelského rozhraní společností Xerox PARC. Microsoft ASP.NET MVC je implementací tohoto návrhového vzoru v platformě ASP.NET. Nejednalo se však pouze o její rozšíření. Vývojáři postupem času upravili i části samotného ASP.NET frameworku. Funkcionalita v něm vyvinutá se tedy přenesla do Web Forms, což tuto starší technologii ovlivnilo jak pozitivně, tak negativně. Za pozitivní bylo považováno například rozšíření o podporu směrování požadavků. Některé moderní konstrukce kódu implementované primárně pro MVC naopak nezapadly do staršího konceptu Web Forms. Technologie MVC nebyla plánována jako náhrada Web Forms, ale jako alternativní technologie pro vývoj webových aplikací. Od verze .NET Frameworku 4.5.1 vydané v říjnu 2013 je možné využívat obě technologie společně v jednom projektu [2] [6].

Interakce s MVC vzorem sleduje přirozený cyklus uživatelských akcí a obnovení uživatelského pohledu (dále jako View), kde View je považováno jako bezstavové. To odpovídá i principu požadavků a odpovědí protokolu HTTP a obecnému principu webových aplikací. Vzor MVC zároveň vynucuje Separation of Concerns (dále jen SoC), v českém překladu Oddělení zodpovědností. Doménový model a aplikační logika jsou oddělené od uživatelského prostředí. Aplikace implementující vzor MVC tedy musí být rozdělená minimálně na tři části [6]:

- Model, obsahující nebo reprezentující uživatelská data.
- View, vykreslující části Modelu jako uživatelské rozhraní.
- Controller, zpracovávající příchozí požadavky. Ten následně provádí operace na Modelu a vybírá View, které se vykreslí uživateli.

Tento způsob rozdělení částí aplikace přináší několik zásadních výhod. Každá část architektury je přesně definovaná a nezávislá. To přináší výhody při úpravách

nebo rozšiřování aplikace, které jsou rychlé a jednoduché i v komplexních aplikacích. Další výhodou je možnost jednotkového testování, které lze realizovat i za pomocí nástrojů ve vývojovém prostředí Visual Studio nebo jiných nástrojů [2] [7].

Technologie naopak není příliš vhodná pro menší aplikace s jednoduchou aplikační logikou. V porovnání se starší technologií Web Forms nejsou k dispozici žádné znovupoužitelné komponenty. View engine Razor, který ASP.NET MVC primárně používá pro vykreslování View, obsahuje implementaci pouze několika základních HTML helperů. Jedná se o metody zjednodušující vykreslení běžně používaných formulářových polí, jako je tlačítko nebo textový vstup. Technologie však nenabízí další možnosti pro tvorbu bohatého uživatelského rozhraní nebo asynchronní komunikaci mezi prohlížečem a serverem. Pro tyto účely je nutné využít nástroje a knihovny třetích stran nebo implementovat vlastní funkcionalitu [6].

3.4 .NET Core a ASP.NET Core

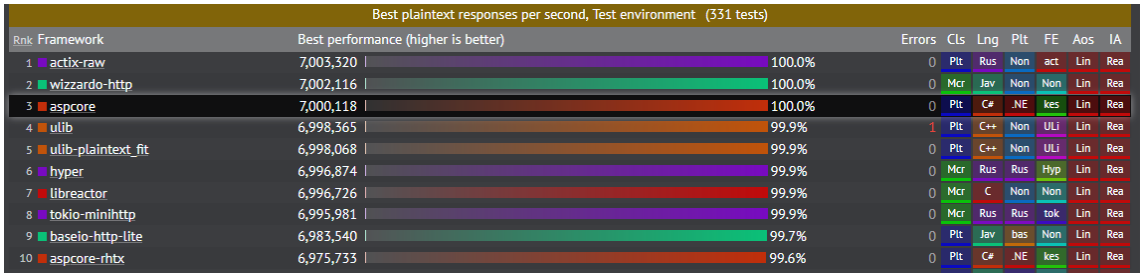
První verze .NET Core byla vydána v červnu 2016. Pro .NET vývojáře se jednalo o novou, revoluční platformu. Jejím základem je programovací jazyk C# a .NET Framework. Technologie však byla od základů přepracována a navržena multiplatformě. Již od prvního vydání ji lze provozovat na operačních systémech Windows, MacOS i Linuxu. Multiplatformní vývoj však není jedinou zásadní změnou v .NET Core. Zdrojové kódy platformy i všech souvisejících frameworků jsou zveřejněné jako open-source. Microsoft od prvních kroků vývoje platformy aktivně spolupracuje s vývojářskou komunitou, která intenzivně přispívá k vývoji. Nový vývoj i změny v kódu jsou od komunity nejen přijímané, ale i očekávané. K prvnímu vydání .NET Core přispělo svými nápady i prací více než 10 000 vývojářů. Vývoj, který u .NET Frameworku zajišťovaly relativně malé týmy, je nyní dostupný komunitě, která může poskytovat nejen dodatečnou funkcionalitu, ale i optimalizace výkonu nebo opravu chyb. Open-source iniciativa nepokrývá jen samotný vývoj, ale i celou dokumentaci, která je dostupná na portále Microsoft Docs [8].

Vývoj .NET Core nicméně nebyl jednoduchý a před jeho prvním vydáním bylo nutné vyřešit nespočet problémů a překážek. Jedním z prvních impulzů pro vývoj

nové platformy byl pro Microsoft značný počet požadavků na běh ASP.NET aplikací na jiných platformách než Windows. Na základě těchto požadavků a diskusí s týmy vyvíjejícími operační systém Windows byl vytvořen projekt s názvem „Project K“, jehož cílem bylo zpřístupnit ASP.NET i na další platformy. Tento cíl byl nakonec velmi komplexní a vyústil v přepsání celého frameworku. To provázelo i mnoho rozhodnutí o tom, jaká část frameworku má být zahrnuta ve výsledném projektu, aby tím ovlivnila jeho další vývoj. Bylo rozhodnuto, že nejlepším přístupem bude vytvořit jednu platformu, která může být použita pro webové aplikace a RESTful služby. Technologie ASP.NET MVC a ASP.NET WebAPI byly sloučeny do jednoho frameworku. Technologie Web Forms byla v novém frameworku vyloučena. První pojmenování nově vznikajícího projektu se stalo ASP.NET 5 [9].

Během postupujícího vývoje došlo k přejmenování na „Core“, celá platforma tedy byla pojmenována .NET Core, webové technologie pak ASP.NET Core [10]. Od prvního vydání bylo vydáno několik hlavních verzí, z nich v tuto chvíli poslední je verze .NET Core 3.1 vydaná 3. prosince 2019 s dlouhodobým obdobím podpory. Ten je označován jako Long Term Support (dále jen LTS) [1]. Tímto pojmenováním však postupný vývoj nekončí. Během tvorby této práce Microsoft oznámil, že další plánovaná verze po .NET Core 3.1 ponese název .NET 5. Mělo by se jednat o zpřehlednění celé situace mezi .NET Frameworkem a .NET Core. Bude existovat pouze jeden .NET, který bude možné použít k cílení na Windows, Linux, MacOS, iOS, Android, tvOS, watchOS, WebAssembly a další [11].

Multiplatformnost je pouze jeden z mnoha cílů celé .NET Core platformy. Jedním z dalších cílů, který se vývojářům podařilo splnit, je výkon a rychlost celé platformy. ASP.NET Core aplikace se pravidelně umisťují na předních příčkách všech relevantních testů výkonů. To je umožněno hlavně díky tomu, že výkon platformy byl brán v potaz již během návrhu i vývoje. Výkon je základním kritériem při rozhodování o dalším vývoji. Výsledek jednoho z výkonových testů nezávislé společnosti TechEmpower je zachycen na Obrázek 1 [12].



Best plaintext responses per second, Test environment (331 tests)

Rnk	Framework	Best performance (higher is better)		Errors	Cls	Lng	Plt	FE	Aos	IA
1	actix-raw	7,003,320	100.0%	0	Plt	Rus	Non	act	Lin	Rea
2	wizzardo-http	7,002,116	100.0%	0	Mcr	Jav	Non	Non	Lin	Rea
3	aspcore	7,000,118	100.0%	0	Plt	C#	.NE	kes	Lin	Rea
4	ulib	6,998,365	99.9%	1	Plt	C++	Non	ULi	Lin	Rea
5	ulib-plaintext_fit	6,998,068	99.9%	0	Plt	C++	Non	ULi	Lin	Rea
6	hyper	6,996,874	99.9%	0	Mcr	Rus	Rus	Hyp	Lin	Rea
7	libreactor	6,996,726	99.9%	0	Mcr	C	Non	Non	Lin	Rea
8	tokio-minihttp	6,995,981	99.9%	0	Mcr	Rus	Rus	tok	Lin	Rea
9	baseio-http-lite	6,983,540	99.7%	0	Plt	Jav	bas	Non	Lin	Rea
10	aspcore-rhtx	6,975,733	99.6%	0	Plt	C#	.NE	kes	Lin	Rea

Obrázek 1 – Výsledek benchmarku webových frameworků [12]

Otevření .NET Core platformy zároveň přineslo nutnost doplnit i plnou podporu příkazového řádku. Pro vývoj aplikací již nebylo nadále zamýšleno s úzkým propojením mezi frameworkem a vývojovým prostředím Visual Studio, které bylo vždy pro vývoj aplikací nad .NET Frameworkem doporučované a v častých případech i nutné. Naopak vznikl prostor pro použití jakéhokoli textového editoru a .NET Core Command Line Interface (dále jen CLI), v českém překladu příkazový řádek [8].

Dalším splněným cílem vývoje platformy byly různé módy nasazení. Prvním z nich je „Portable“. V tomto módu se při vývoji a publikaci aplikace specifikuje cílová verze .NET Core, která je zároveň nainstalovaná v běhovém prostředí. Výsledná publikovaná aplikace obsahuje pouze své zdrojové kódy a konečná velikost balíčku je velmi malá. Druhým módem nasazení je „Stand-alone“. Tento způsob ve finálním balíčku nezahrnuje jen aplikační soubory, ale i celé běhové prostředí pro cílovou platformu. To sice vytváří větší instalační balíček, avšak aplikace je izolovaná od zbytku běhového prostředí [8].

.NET Core podporuje reálnou „side-by-side“ instalaci. Na jednom běhovém prostředí může být nainstalováno více aplikací v různých verzích frameworku, které se navzájem neovlivňují. Jedná se o zásadní rozdíl a vylepšení proti .NET Frameworku, u kterého při instalaci více verzí docházelo k nežádoucím vedlejším efektům [8]. Verze .NET Core 3.0 přinesla funkcionalitu s názvem „Trimming“, v českém překladu oříznutí, která při publikaci aplikace umožňuje odstranění nepotřebných knihoven a tím zmenšuje velikost výsledného balíčku [13].

3.5 Razor Pages

Společně s verzí ASP.NET Core 2.0 byl vydán i framework pro vývoj webových aplikací s názvem Razor Pages. Jeho hlavním zaměřením jsou jednodušší a kompaktní aplikace s menším množstvím závislostí. Je zde zároveň předpokládána jednodušší aplikační logika stránky soustředěná do jednoho místa. Jak již z názvu vyplývá, Razor Pages využívají pro vykreslování engine Razor, používaný i v ASP.NET MVC. S touto technologií sdílí většinu své infrastruktury. Je tedy dostatečně flexibilní a umožňuje kombinaci obou technologií. Ze svého návrhu nesplňuje architektonický vzor MVC. Model a Controller jsou zde nahrazeny třídou PageModel, která obsahuje jak data modelu vykreslovaná do View, tak akční metody obsluhující příchozí požadavky. PageModel může být přímo součástí View, případně extrahován do zvláštní třídy a View na něj následně odkazuje direktivou @model. Pro oddělení interních vlastností a vlastností nesoucí data pro View je zde použit atribut BindProperty. Tím lze označit vlastnosti, u kterých bude probíhat binding dat. Svou strukturou je podobný původní technologii Web Forms [14] [15].

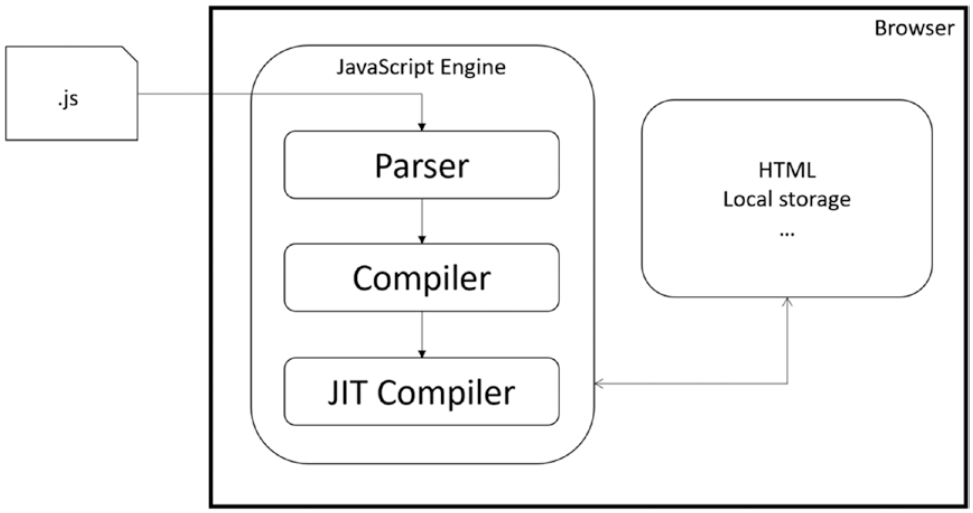
3.6 Blazor

Blazor je webový framework poprvé představený v roce 2017. Původně se jednalo o osobní projekt Steva Sandersona, který ve společnosti Microsoft pracuje v týmu vyvíjejícím ASP.NET. Cílem frameworku je nabídnout možnost vývoje webových aplikací založených na platformě .NET, které lze spouštět na klientské straně ve webových prohlížečích. K tomu využívá webový standard WebAssembly [16].

3.6.1 WebAssembly

Pro vývoj klientské části webových aplikacích se v dnešní době v naprosté většině případů používá JavaScript. Ten byl vytvořen v roce 1995 a od té doby velkým dílem přispívá k určování směru vývoje webových aplikací. Až do poslední doby se jednalo o jediný programovací jazyk, který lze nativně používat ve webovém prohlížeči. Jsou sice k dispozici jiné programovací jazyky na vyšší úrovni jako Dart nebo TypeScript, všechny jsou však následně kompilovány do JavaScriptu. Ten stojí za podobou moderních webových stránek s bohatým a interaktivním uživatelským rozhraním. U nich je předpokladem nerušená práce bez opětovného načínání celých

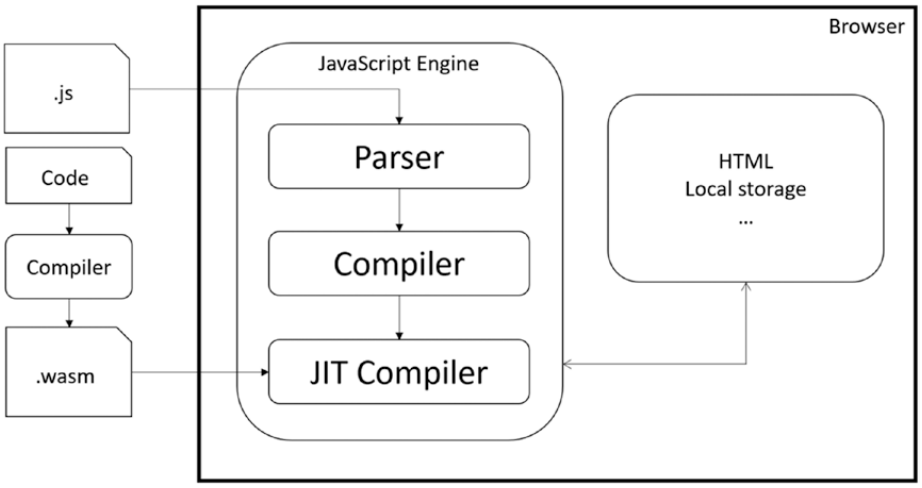
stránek s výkonem blížícím se nativním aplikacím dané platformy. Ve všech moderních prohlížečích se prvořadým cílem stává maximální výkon JavaScriptu. Jádra prohlížečů používají různé optimalizace, jako například Just-in-Time (dále jen JIT) komplikaci, kdy je JavaScript konvertován přímo do nativního kódu. Proces vykonávání kódu je znázorněn na Obrázek 2 [17] [18].



Obrázek 2 – Proces vykonání JavaScriptového kódu [16]

Tento proces vyžaduje mnoho síťových i výpočetních prostředků. Kód musí být nejprve stažen do prohlížeče, tam je následně parsován a zkompilován do nativního kódu, který se vykonává [18].

WebAssembly dovoluje přesunout parsování dat a kompilaci na server. Jedná se o nový binární formát optimalizovaný pro běh ve webovém prohlížeči. S WebAssembly se kód kompiluje do formátu nazvaného WASM. Takto zkompilovaný kód je následně stažen do prohlížeče, kde jej JIT kompilátor kompiluje do nativního kódu. Rozdíl proti vykonávání JavaScriptu je znázorněn na Obrázek 3. Tento způsob kompilace a přenosu dat umožňuje provádět kód s téměř stejnou rychlostí, jako při vykonávání nativního kódu [18].



Obrázek 3 – Proces vykonání WebAssembly [16]

Všechny hlavní webové prohlížeče nyní WebAssembly podporují. Řadí se mezi ně Chrome, Edge, Safari a Firefox, také s podporou v mobilních verzích těchto prohlížečů. Podpora WebAssembly v Internet Exploreru plánovaná není. Aktuálně je považován jako doplněk k JavaScriptu, kterým je možné optimalizovat a zrychlit důležité části webové aplikace. Stejně tak dovoluje použití různých programovacích jazyků jako C, C++ nebo Rust [16]. Bude záležet na dalším vývoji a rozvoji této technologie, díky které se může stát plnohodnotnou alternativou JavaScriptu nebo i jeho úplnou náhradou [18].

3.6.2 ASP.NET Core Blazor

Název Blazor vznikl spojením slov Browser a Razor. Blazor umožňuje vytváření klientských webových aplikací pomocí jazyka C# a vykreslovacího enginu Razor. Pro kompilaci kódu do WebAssembly používá open-source projekt Mono. Mono je implementací .NET CLI specifikace, která umožňuje spouštět .NET assemblies. Původním cílem tohoto projektu byla možnost spouštění .NET aplikací na Linuxu. Nyní je používáno zejména v technologii Xamarin pro tvorbu mobilních aplikací, které umožňuje spouštět nejen v mobilních operačních systémech Android a iOS. Mono je naprogramováno v jazyce C++, který je možné kompilovat do WebAssembly. Tým vyvíjející Mono jej zkusil do WebAssembly kompilovat a výsledek byl pozitivní. Tento pokus přinesl možnosti a nápady pro další pokusy spouštět .NET assemblies ve webovém prohlížeči [16].

Jednou z experimentálních technologií je i Blazor. Umožňuje vytvářet aplikace za pomocí prověřených ASP.NET technologií a následně celou aplikaci jedním příkazem spustit ve webovém prohlížeči. Jedná se nepochybně o dobrou zprávu pro tisíce .NET vývojářů, kteří při vývoji webových aplikací nemusí používat JavaScript. Zajímavé možnosti tato technologie ale přináší i dalším vývojářům. Při vývoji lze využít plnou podporu .NET ekosystému. Blazor obsahuje kód pro práci s Document Object Model (dále jen DOM). Zároveň lze pro vývoj aplikací použít nepřeberné množství již existujících tříd a knihoven dostupných na této platformě. Mimo jiné lze využít rozsáhlé nabídky podpůrných nástrojů pro tuto platformu. Při vývoji pomocí Visual Studia je například používán plnohodnotný debugger i pro aplikaci běžící pouze ve webovém prohlížeči [16].

V některých situacích použití Blazoru ještě není vhodné nebo je zbytečně komplikované či náročné. Pro tyto účely je implementována funkcionalita s názvem JavaScript interoperability. Ta umožňuje jednoduché volání statických .NET metod z JavaScriptového kódu i opačné volání JavaScriptových funkcí z Blazoru [19].

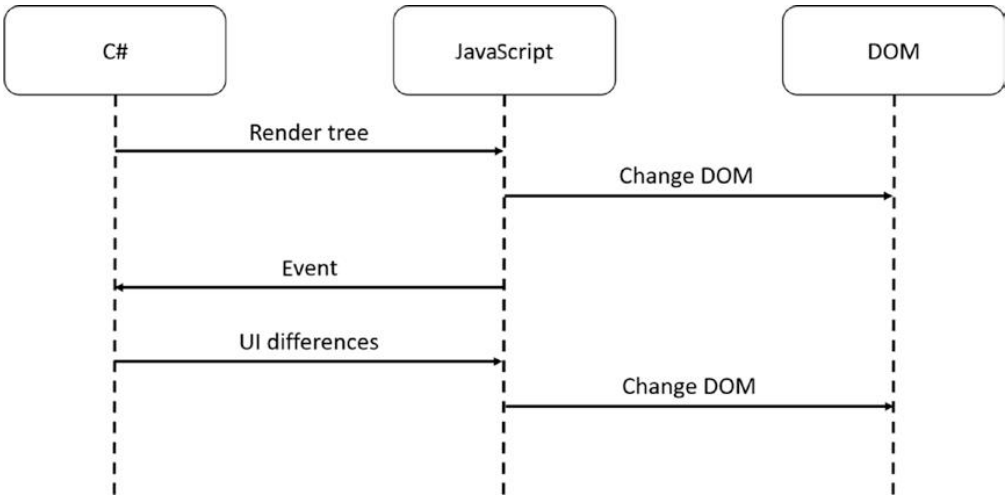
Podobně jako JavaScriptové frameworky je Blazor založen na systému komponent. Ty rozdělují části uživatelského rozhraní a mohou být do sebe zanořovány. Tento způsob organizace zjednodušuje znovupoužitelnost částí vytvořené aplikace. Pro tvorbu uživatelského rozhraní zatím framework nabízí jen několik základních komponent. Jedná se zejména o komponenty pro autentizaci a autorizaci uživatele a jednoduchá formulářová pole. Microsoft se v tuto chvíli soustředí primárně na základní funkcionalitu frameworku a nechává prostor koncovým vývojářům i výrobcům komponent třetích stran. Z tohoto pohledu lze spatřit podobný přístup jako u ASP.NET MVC. Nicméně tvorba a použití vlastních komponent je v Blazoru velmi jednoduché a intuitivní, takže tuto strategii nelze označit za zásadní nevýhodu [16].

Během tříletého vývoje této technologie došlo k mnoha zásadním rozhodnutím a změnám. Blazor je stejně jako celý .NET Core ekosystém vyvíjen Microsoftem jako zdarma dostupné open-source řešení. Nápady, opravy, vylepšení i implementace nové funkcionality jsou od komunity pozitivně přijímány. Blazor je nyní dostupný ve dvou různých modelech nasazení, Blazor Server a Blazor WebAssembly. První z nich je již oficiální součástí .NET Core 3.1 vydaného

v lednu 2020. Aktuální verze Blazor WebAssembly je nyní označená jako beta s plánovaným datem vydání v květnu 2020 [20]. Microsoft již Blazor neoznačuje jako experimentální, což potvrzuje jeho důvěru nejen ve framework, ale i celý standard WebAssembly. Tato situace může v pozitivním smyslu přesvědčit mnoho dalších vývojářů o budoucnosti této technologie. Oba modely nasazení budou detailněji popsány v následujících kapitolách [14] [16].

3.6.3 Blazor WebAssembly

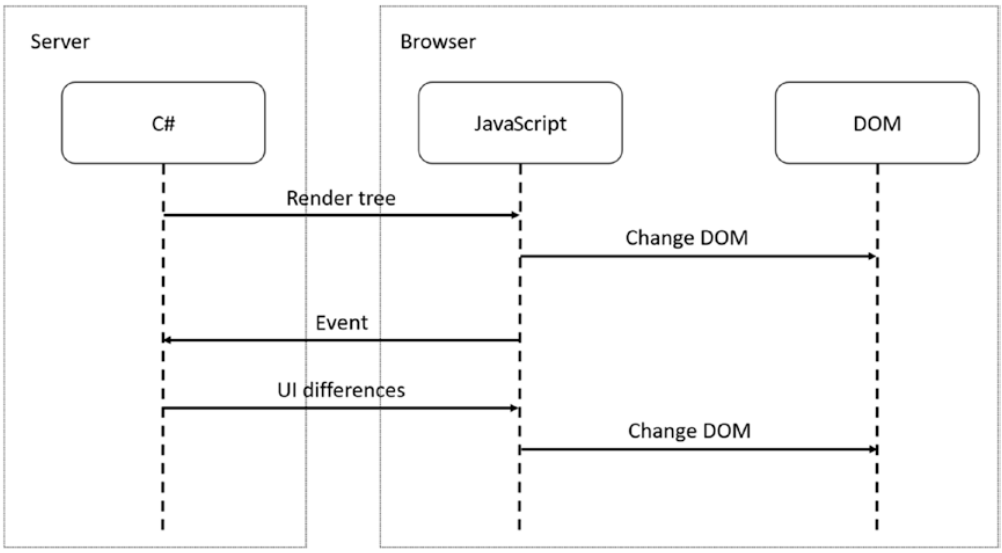
WebAssembly společně s Mono umožňuje spouštět .NET kód přímo v prohlížeči. Při použití toho modelu dojde k vykonání zdrojového kódu a dynamickému sestavení webové stránky přímo v prohlížeči. Zdrojový kód stránky je kompilován do .NET kódu a následně vykonán v Blazor enginu [14]. Výsledkem toho vykonání je hierarchická struktura dat nazývaná render tree. Ta obsahuje detailní informace o všech prvcích zobrazených na stránce. Data jsou předána JavaScriptové části Blazoru, který zajistí změnu DOM. Při vyvolání události, například při kliknutí na tlačítko, se provede přegenerování render tree. Jakékoliv změny jsou následně zaslány do JavaScriptové částí, která provede aktualizaci DOM. Tento model je velmi flexibilní. Průběh vykreslování a překreslování stránky je zobrazen na Obrázek 4. Umožňuje vytvářet progresivní webové aplikace a zároveň může být zahrnut například do aplikací využívající framework Electron, který slouží k vývoji desktopových aplikací [16] [19].



Obrázek 4 – Generování DOM v Blazor [16]

3.6.4 Blazor Server

Alternativním modelem nasazení je Server. Při použití tohoto modelu nasazení je klientská logika vykonávána na serveru. Přenos dat mezi klientským prohlížečem a serverem probíhá pomocí SignalR. SignalR je framework, který ve webových aplikacích umožňuje asynchronní přenos zpráv v reálném čase a taktéž patří do technologií platformy .NET. Render tree je sestavován na serverové straně, následně je serializován a odeslán do prohlížeče. Tam je pomocí JavaScriptu deserializován a provede se aktualizace DOM. Při interakci uživatele se stránkou se pomocí JavaScriptu požadovaná data serializují a odešlou na server. Ten aktualizuje render tree a případné změny odešle zpět do prohlížeče. Průběh vykreslování a překreslování stránky je detailně znázorněn na Obrázek 5. Zásadním rozdílem je zde to, že není potřeba do klientského prohlížeče posílat běhové prostředí pro Mono ani Blazor assemblies. Aplikace i způsob programování nicméně zůstávají stejné [16] [19].



Obrázek 5 – Model nasazení Blazor Server [16]

Tyto dva zmíněné modely nasazení jsou na první pohled znatelně odlišné a každý z nich má určité výhody i nevýhody. Framework je nicméně navržený tak, že při správném návrhu a implementaci aplikace je možné mezi jednotlivými modely jednoduše přepnout. Aplikaci vykonávanou na serveru lze následně provozovat jako Single-page aplikaci a opačně. Při použití serverového modelu se výrazně zmenší množství přenášených dat při prvním spuštění aplikace.

Ta se načte mnohem rychleji. Nicméně pro používání aplikace musí být uživatel stále připojený k serveru. Pokud spojení mezi prohlížečem a serverem z jakéhokoli důvodu selže, aplikace přestane fungovat a uživatel může ztratit svou práci. Server zároveň musí obsluhovat veškeré požadavky klientských aplikací, což při větším počtu klientů může způsobovat výkonové problémy. Na serveru je nutné udržovat stav každé připojené klientské aplikace, což znamená zvýšené nároky na paměť a výkon serveru [16] [19].

4 Vizualizace dat a reporty

Následující kapitola nejprve popisuje oblast Business Intelligence. Dále pak vybrané možnosti reprezentace a vizualizace dat relevantní k řešenému tématu této práce a technologie k vytváření reportů z těchto dat. K vybraným typům reportů budou v dalších kapitolách popsány možnosti jejich implementace ve webových technologiích na platformě .NET Core. Dále budou představeny i řešení business intelligence společnosti Microsoft, které lze do webových aplikací integrovat.

Jedním ze základních požadavků na funkcionalitu informačních systémů je reporting. Obzvlášť v rychle se měnícím prostředí je pro úspěšný chod společnosti nutné mít aktuální přehled o všech potřebných informacích. Schopnost vizualizovat data společně s jejich významem je základním klíčem k jejich převedení na informaci, která může být použita pro lepší rozhodování [26]. V závislosti na činnosti společnosti a zaměření konkrétního informačního systému musí systém zpracovat a reprezentovat požadovaná data v aktuálním čase. Reprezentace dat musí být zároveň uživatelsky přívětivá a snadno přizpůsobitelná. Například z pohledu výrobních procesů mohou reporty zobrazovat informace o stavu skladových zásob, aktuální stavy výrobních linek, informace o prodejích nebo situaci na trhu. U každé z reportovaných oblastí je často požadován hierarchický rozpad informací od obecného pohledu až na nejnižší úroveň. Příkladem mohou být reporty prodejů nadnárodní korporace za posledních několik let, které je možné seskupit a filtrovat podle kontinentů, zemí, oblastí, konkrétních obchodních zástupců a na nejnižší úrovni zobrazit konkrétní prodej každého produktu. Reporting se často využívá nejen jako pohled na historická či aktuální data, ale i k plánování a strategickým rozhodnutím do budoucna. Možnosti reprezentace těchto dat jsou velmi rozmanité a při návrhu reportu je nutné se zaměřit na jejich správný výběr. Reporting dat může mít zároveň různou podobu, ať už prezentaci formou grafu, datové tabulky, kontingenční tabulky či pokročilejších nástrojů, ale i jednoduchého hlášení nebo výkazu zasílaného například emailem [23] [26].

Při návrhu nového reportu je vhodné nejprve stanovit odpovědi na tři základní otázky: Kdo, Co, a Jak. První z nich stanoví cílovou skupinu osob, která bude data zobrazená reportem používat. Druhou otázkou je, co je cílem

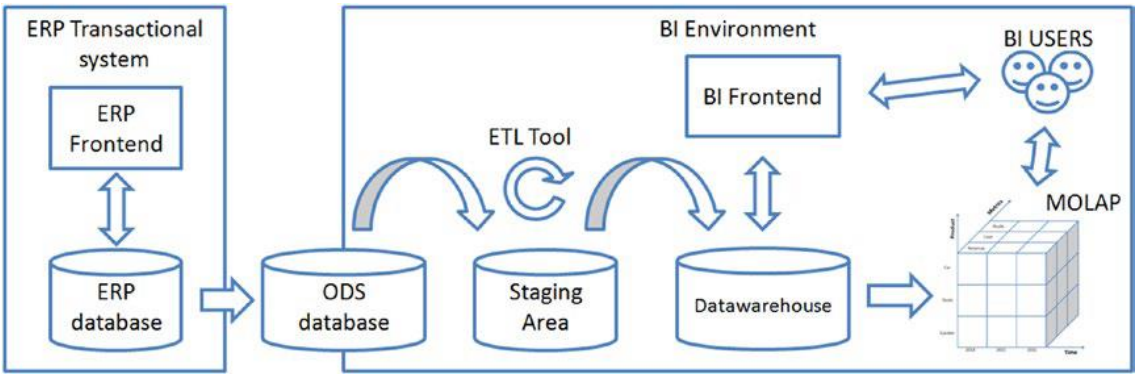
daného reportu a jaké informace má zpracovávat a zobrazovat. Poslední otázka řeší problematiku jak použít dostupná data, aby splnila svůj účel. Po zodpovězení těchto tří obecných otázek lze mnohem snáze stanovit detailní specifikaci reportu [21]. Lze stanovit způsob reprezentace dat, omezit jejich výběr jen na určitou část, nabídnout možnost filtrování či seskupování dat nebo interaktivní možnosti rozpadu jednotlivých úrovní [23].

4.1 Business intelligence

Reportování dat, jejich prezentace a vizualizace spadá do oblasti Business intelligence, často nazývané zkratkou BI. Pro vysvětlení pojmu Business intelligence neexistuje jednotná definice. Kniha Practical Business Intelligence používá definici, že Business intelligence je proces dodávání akceschopných obchodních rozhodnutí z analytické manipulace a prezentace dat v mezích obchodního prostředí [24]. Další definice zní, že Business intelligence představuje specifický typ úloh podnikové informatiky, které podporují analytické, plánovací a rozhodovací činnosti a využívají principy, které těmto činnostem nejvíce odpovídají [25]. Jedna z nejlépe vyhovujících definic je, že Business intelligence je sada teorií, metodologií, architektur a technik, které transformují nezpracovaná data do smysluplné a užitečné informace pro obchodní účely [23]. Tato definice popisuje celkovou hloubku dané problematiky, která nepojednává pouze o reprezentaci dat a informací, ale o celkovém procesu. Obecná architektura Business intelligence je znázorněná na Obrázek 6. Základním prvkem technického řešení jsou zdrojové systémy zaznamenávající data z operací. Jsou často označovány jako transakční, Online Transaction Processing (dále jen OTLP) nebo také primární systémy. OTLP je označení pro systémy ukládající data v databázi, které umožňují přístup více uživatelů a zpracování jejich operací v reálném čase. Data jsou následně zpracována nástroji ETL (Extract, Transform, Load). Tyto nástroje extrakce, transformace a nahrávání dat zajišťují jejich přenos mezi libovolnými databázemi nebo soubory. Data jsou následně uložena do datového skladu [23]. Následnou úlohou je jejich reporting. Jedná se o přeměnu dat v informace, jejich vizualizace a poskytnutí uživatelům ve vhodné formě. Samotné reporty lze rozdělit do dvou skupin. První z nich jsou statické reporty, které lze pouze číst. Druhou skupinou jsou

interaktivní reporty, které je možné ovládacími a interaktivními prvky přizpůsobovat. Jinou možností analýzy je Online Analytical Processing (dále jen OLAP). Jedná se o možnost online zpracování a analyzování velkého objemu dat do přístupné a srozumitelné formy. Souvisejícím případem analýzy dat je taktéž dolování dat (Data mining), při kterém se hledají doposud nedefinované vztahy a skutečnosti. Zároveň je možné z dat odvozovat i prediktivní informace [23] [28].

Samotné řešení BI se při produkčním nasazení přizpůsobuje požadavkům a podmínkám daného podniku. Organizace komponent, jejich využití i použité technologie jsou variabilní a realizace může být provedena různými způsoby [23].

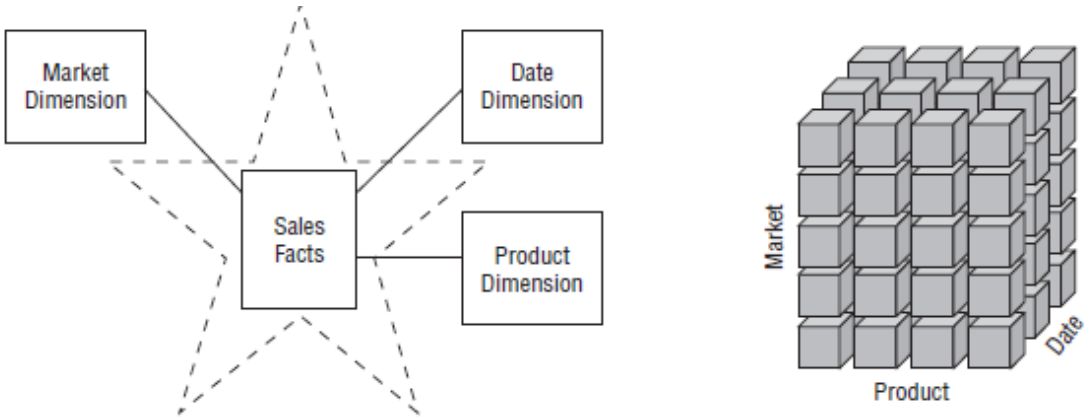


Obrázek 6 – Obecné rozložení komponent BI systému [23]

4.2 Dimenzionální modelování

Dimenzionální modelování je v rámci business intelligence obecně preferovanou technikou pro prezentaci analytických dat. Umožnuje současně vykonávat dva základní požadavky, a to poskytnutí dat ve srozumitelné formě pro podnikové uživatele společně s vysokým výkonem dotazů. Model zahrnuje stejná data, která jsou v transakčních systémech uložená v relačních databázích nebo v datových skladech. Rozdílem je zde schéma jejich uložení. V dimenzionálním modelu jsou dle způsobu nahlížení na podniková data stanoveny dva typy tabulek: tabulky faktů a tabulky dimenzí. Zároveň jsou zde používány různé topologie modelů: schéma vločky (snowflake schema), schéma hvězdy (star schema) nebo OLAP krychle (OLAP cubes) [32]. Poslední dvě jmenovaná schémata jsou znázorněna na Obrázek 7. V tabulkách faktů v dimenzionálním modelu jsou uloženy metriky zaznamenávané v čase. Tato data, která mohou obsahovat například bankovní

operace provedené v určitém čase, bývají obvykle uložena v největších a nejširších tabulkách. V tabulkách faktů jsou obsaženy cizí klíče do tabulek dimenzí, které jsou těmito tabulkami úzce spojené, a dále hodnoty metrik. Dimenzionální tabulky slouží k bližšímu popsání a kategorizaci záznamů ve faktových tabulkách. Každá dimenze je definována primárním klíčem, který zachovává referenční integritu s tabulkou faktů. Jednotlivé atributy dimenzí slouží jako primární zdroj dotazů či pro seskupování dat. Úroveň podrobnosti reprezentovaných dat se nazývá granularita [29]. Mezi těmito úrovněmi je následně možné provádět operace vnořování a vynořování (drill down a roll up). Dimenzionální model lze realizovat jak v relační databázi, tak v multidimenzionální databázi [30] [32].



Obrázek 7 – Schéma hvězdy a OLAP krychle [29]

Technologií související s dimenzionálním modelováním v multidimenzionálních databázích je OLAP. Nástroje OLAP umožňují uživatelům interaktivně analyzovat multidimenzionální data z různých perspektiv. Databáze je reprezentovaná jako vícerozměrná krychle, v některých případech nazývaná kostka. Ta odpovídá tabulce relační databáze. Hranami krychle jsou jednotlivé dimenze. Výsledné záznamy se nachází na průsečících. Při reálném použití je často specifikováno více dimenzí. V těchto případech je velmi obtížné si daný model představit jako krychli. Pro analytické potřeby je možné s datovou krychlí provádět několik analytických operací, které zajišťují zpracování a projekci dat pro jejich snazší pochopení. Mezi základní operace s datovými kostkami patří [31] [32]:

- Vnořování a vynořování (drill down a roll up) – posun datovou hierarchií v obou směrech.

- Krájení a kostkování (slicing a dicing) – omezení jedné nebo více dimenzí na podmnožinu obsahující jeden nebo více prvků.
- Pivotování – otočení krychle pro získání jiné perspektivy pohledu na vztahy dat.
- Agregace – konsolidace dat pomocí specifikovaného vzorce.

U OLAP je kladen velký důraz na rychlost zpracování uživatelských požadavků. Některá data jsou ukládána duplicitně, zároveň se využívá více indexů. V závislosti na analyzovaných datech existují tři způsoby pro zpracování a komprimaci dat [31] [32]:

- Relační databázový OLAP (dále jen ROLAP) – při ROLAP analýze jsou využívána data uložená pouze v relační databázi. Tabulky faktů a dimenzí jsou uložené jako relační tabulky a každá operace se provádí pomocí SQL příkazů. Operace krájení a kostkování spočívají v přidání podmínky SQL dotazu. Samotné použití SQL v některých případech může omezovat výkon i dostupnou funkcionalitu. Výhodou tohoto způsobu analýzy je možnost zpracování velkého množství dat, které je limitováno pouze základní relační databází.
- Multidimenzionální OLAP (dále jen MOLAP) – data jsou ukládána do multidimenzionálních struktur. Společně s daty jsou ukládány i částečné výpočty, agregace dat a jejich sumarizace. MOLAP tak nabízí rychlou odezvu na uživatelské dotazy. Nevýhodou je redundantní ukládání dat v relační i multidimenzionální databázi.
- Hybridní OLAP (dále jen HOLAP) – jedná se o kombinaci předchozích dvou způsobů. HOLAP zprostředkovává data uložená v relační databázi a umožňuje provádět agregační funkce v OLAP datových kostkách.

4.3 Datové tabulky

Tabulky jsou jedním ze základních způsobů uspořádání dat do řádků a sloupců, případně do složitějších struktur. Jsou široce používané v komunikaci, výzkumu i datové analýze. Při čtení tabulek uživatel obvykle postupuje systematicky

po řádcích a sloupcích v závislosti na datech, které tabulka obsahuje a které uživatel hledá. Jedná se o způsob reprezentace, který je vhodný pro detailní zobrazení většího množství dat, zejména pak při reprezentaci v rozdílných jednotkách měření. Naopak není vhodná pro úlohy, ve kterých je požadováno na první pohled snadno a rychle rozpoznat výsledek nebo provést rychlé rozhodnutí. Nevhodným scénářem pro použití tabulkového reportu může být například hlavní nástěnka informačního systému nebo prezentace [21].

Pro zpřehlednění tabulkových reportů se v některých případech používá zobrazení pomocí teplotní mapy, v anglickém překladu Heatmap. V teplotní mapě se kromě zobrazovaných hodnot využívají i barevné buňky, které reprezentují relativní velikost hodnot. Nejčastěji se pro zachování přehlednosti používá spojité spektrum určité barvy, kdy světlý odstín barvy může značit nejmenší hodnotu, nejtmavší odstín naopak nejvyšší hodnotu. Toto barevné rozlišení oproti běžné tabulce zjednodušuje a urychluje hledání potencionálně nedůležitějších informací. Rozdíl mezi běžnou tabulkou a teplotní mapou je znázorněn na Obrázek 8 [21].

Table

	A	B	C
Category 1	15%	22%	42%
Category 2	40%	36%	20%
Category 3	35%	17%	34%
Category 4	30%	29%	26%
Category 5	55%	30%	58%
Category 6	11%	25%	49%

Heatmap

LOW-HIGH

	A	B	C
Category 1	15%	22%	42%
Category 2	40%	36%	20%
Category 3	35%	17%	34%
Category 4	30%	29%	26%
Category 5	55%	30%	58%
Category 6	11%	25%	49%

Obrázek 8 – Zobrazení dat v tabulce [21]

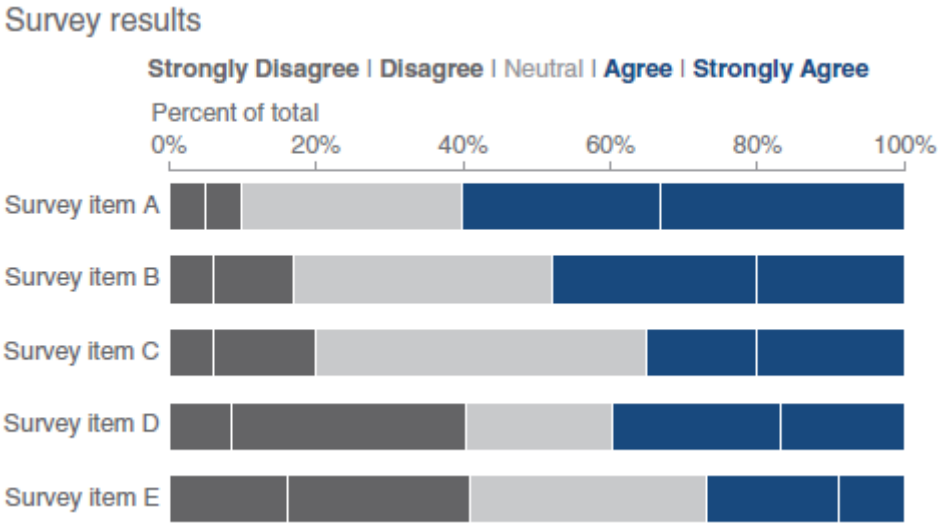
Výhodou dat reprezentovaných v tabulkách bývá rozsáhlá možnost jejich přizpůsobení. Uživatelé často dostávají možnost změnit viditelnost a pořadí zobrazovaných sloupců. Dále je obvykle možné řadit a filtrovat data dle hodnot v každém sloupci za různých podmínek nebo seskupit data i v několika úrovních a následně zobrazit data spadající do určité úrovně. Na datové tabulky bývají v informačních systémech kladeny vysoké nároky. Díky tomu, že se jedná o jednu z nejuniverzálnějších a nejpoužívanějších forem zobrazení dat, je nutné se na jejich tvorbu detailně soustředit. V tabulkách je často nutné zobrazit stovky tisíc záznamů.

Zároveň zde existuje množství uživatelských požadavků na jejich dynamickou funkcionalitu. Pro samotnou tabulku je nutné zajistit dostatečně výkonný a optimalizovaný datový zdroj a v prezentační vrstvě aplikace pak optimalizovanou a uživatelsky přívětivou komponentu. V rámci uživatelských požadavků je s ohledem na množství dat nutné jejich stránkování, možnost jejich filtrování i stanovení filtrovacích pravidel a řazení dat. Dalším častým požadavkem je možnost dynamického zobrazení, přesouvání a odstranění jednotlivých sloupců v tabulce tak, aby každý uživatel mohl danou tabulku přizpůsobit svým požadavkům [22]. V některých případech je požadována taktéž podpora dynamického seskupení dat dle vybraného sloupce, což s ohledem na množství zobrazovaných dat zvyšuje nároky výkonu datového zdroje i komponenty uživatelského prostředí. S ohledem na způsob vývoje moderních aplikací je u těchto komponent požadována i jejich znovupoužitelnost nejen v jiných částech aplikace, ale i v dalších systémech [21] [26].

4.4 Diagramy

Diagramy slouží ke grafické reprezentaci dat. Dobře navržený diagram umožňuje mnohem rychlejší zpracování obsažených informací než v tabulkách. Jedná se o grafické znázornění dat, pojmů, vztahů nebo historického vývoje. V českém jazyce se pro diagramy používá taktéž synonymum grafy. V rámci této práce budou používány oba výrazy. V oblasti reprezentace dat se nejčastěji používají čtyři kategorie diagramů: bodový, liniový, sloupcový a plošný. Často je také používán kruhový diagram, někdy také nazývaný koláčový nebo výsečový. Nicméně tento způsob zobrazení je pro praxi nedostatečný. Jedním z hlavních důvodů je mnohem náročnější rozlišení velikostí jednotlivých položek zobrazených v diagramu [21]. Diagramy jsou většinou statické a slouží pouze pro prezentaci dat bez další uživatelské funkcionality. Některé diagramy je možné doplnit o jednoduché skrývání či zvýraznění jednotlivých položek nebo případně o rozpad do další úrovně. Příkladem může být zobrazení sloupcového grafu o zalidnění kontinentů, kdy při kliknutí na daný kontinent dojde k zobrazení zalidnění v zemích ležících v daném kontinentu. Nicméně tato funkcionalita nebývá příliš obvyklá [33]. Jedním ze základních doporučení při vizualizaci dat je nepoužívat v diagramu 3D efekty,

pokud není nutné vizualizovat i třetí dimenzi. Trojrozměrné efekty jsou v diagramech vizuálně velmi složité a komplikují jejich interpretaci [21]. Na Obrázek 9 je zobrazen horizontální diagram s plnou šířkou, který pro lepší přehlednost používá rozdílné odstíny barev [26] [27].



Obrázek 9 – Ukázka horizontálního sloupcového diagramu s plnou šířkou [21]

4.5 Microsoft SQL Server

SQL Server společnosti Microsoft je databázový a analytický systém s velmi komplexní nabídkou podporované funkcionality a služeb. Nejen z pohledu business intelligence se jedná o jeden nejkomplexnějších produktů pro správu a analýzu dat. Nejnovější dostupná verze popisovaná v této kapitole je SQL Server 2019. Z historického vývoje produktu se jedná o revoluční verzi, která obsahuje nejen vylepšení výkonu, funkcí a spolehlivosti, ale přidává taktéž možnost provozu na Linuxových operačních systémech jako Ubuntu, Red Hat Enterprise Linux a SUSE. Zároveň nově nabízí možnost provozu v Docker kontejnerech [35].

Základní komponentou produktu je SQL Server Database engine. Tato základní služba zajišťuje ukládání, zpracování a zabezpečení dat a další funkcionalitu nutnou pro běh databázového serveru. Pro správu a analýzu dat jsou v SQL serveru zahrnuty následující služby [35]:

- Analysis Services (dále jen SSAS): sada nástrojů pro vytváření a správu online analytického zpracování a aplikace pro dolování dat.

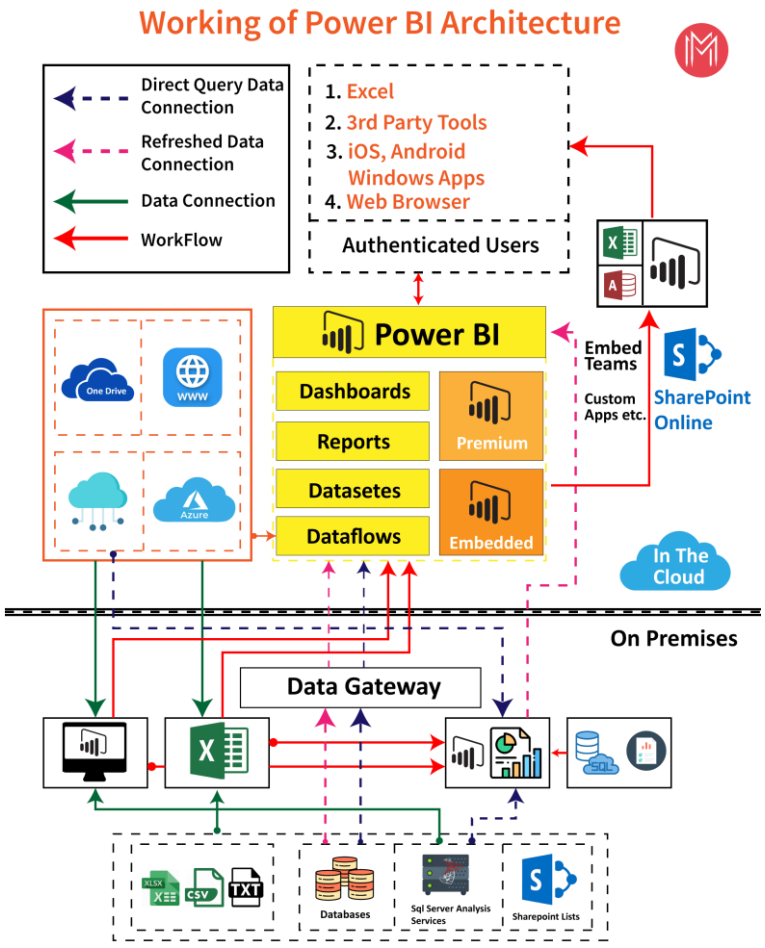
- Reporting Services (dále jen SSRS): serverové a klientské komponenty pro vytváření, správu a nasazení tabulkových, maticových, grafických a formulářových reportů. SSRS je také rozšiřitelná reportovací platforma, kterou je možné použít pro vývoj reportingových aplikací.
- Integration Services (dále jen SSIS): sada grafických nástrojů a programovatelných objektů pro přesouvání, kopírování a transformaci dat.
- Master Data Services (dále jen MDS): řešení pro správu Master dat. MSD může být konfigurován pro správu různých domén (účtů, zákazníků, produktů) a umožňuje zahrnout jejich hierarchie, transakce, verzování dat a obchodní pravidla.

Mimo výše zmíněné služby produkt obsahuje další služby pro strojové učení, podporu programovacích jazyků R a Python a množství nástrojů pro jeho konfiguraci a správu. Mezi ně patří i SQL Server Data Tools, který nabízí vývojové prostředí pro vytváření řešení BI komponent z Analysis Services, Reporting Services a Integration Services. Formálně je tento nástroj nazýván Business Intelligence Development Studio [35] [36].

SQL Server je nabízen v několika edicích, které mají rozdílnou nabídku služeb, modely licencování i cenu. Nejnižší verzí nabízející základní nástroje business intelligence je edice Standard [34].

4.6 Power BI

Microsoft Power BI je cloudová technologie pro reporting dat a datovou analýzu. Jedná se o technologii, která není určená pouze vývojářům pro vytváření reportů, ale i pro pokročilé uživatele a business analytiky. Pro uživatele nabízí jednoduché, snadno použitelné uživatelské rozhraní pro vytváření reportů. Technologie je však na pozadí založena na několika výkonných komponentách, které dovolují vytvořit reporty a provádět datovou analýzu i pro velmi komplexní scénáře. Každá z těchto komponent je zodpovědná za specifickou část. V technologii jsou dostupné komponenty pro sestavování reportů, provádění výpočtů, připojení k datovým zdrojům, sdílení reportů a další. Architektura technologie a rozvržení komponent jsou znázorněné na Obrázek 10 [37].



Obrázek 10 – Komponenty technologie Power BI [38]

Vytvořené reporty mohou být zobrazeny ve webovém prohlížeči, mobilní aplikaci nebo integrovány do vlastních aplikací pomocí API. Mnoho společností z různých důvodů preferuje ponechání jejich dat ve vlastních úložištích. Naopak Power BI je cloudová technologie, která požaduje uložení dat ve stejném prostředí. Pro řešení tohoto problému byl vyvinut Power BI Report Server, který umožňuje instalaci, provoz a sdílení reportů ve vlastním prostředí společnosti. Jedná se o specifickou verzi služby SQL Server Reporting Services. V tuto chvíli v ní zatím není implementována veškerá funkcionalita dostupná v cloudové verzi, i přesto se již jedná o plnohodnotné řešení. Power BI je komplexní a velmi výkonná komerční technologie, která nabízí několik úrovní funkcionality. Od toho se odvíjí i cena daného řešení. Základní verzí je Power BI Desktop, která zdarma nabízí základní funkcionalitu pro tvorbu reportů, jejich publikaci na web

nebo ve formátu csv. Power BI Pro nabízí sdílené výpočetní prostředky s rozšířenou funkcionalitou [37] [40].

Tato verze je nabízena s cenou 8,40 € za uživatele měsíčně. Oproti základní verzi nabízí možnosti sdílení mezi uživateli, integraci s aplikací Excel, podporu více pracovních prostředí nebo přístup k Power BI Apps. Verze Power BI Premium v sobě zahrnuje dedikované výpočetní prostředky a úložiště, podporu pokročilých analytických funkcí nebo podporu big data. Její ceníková cena je 4 212 € měsíčně za každý pronajatý výpočetní uzel [39].

5 Balíky komponent pro vizualizaci dat a tvorbu reportů

Při vývoji software jsou kladeny velké nároky na rychlost i efektivitu vývoje, modularitu a znovupoužitelnost různých částí vyvinuté aplikace. Na základě těchto požadavků jsou různé části softwaru sdílené mezi jednotlivými vývojáři, týmy v rámci společností nebo mezi odbornou komunitou. Mezi sdílené komponenty lze zařadit jednoduché prvky uživatelského rozhraní i komplexní řešení různých modulů aplikace. Vzhledem k velmi bohaté a dlouhé historii .NET platformy je na trhu k dispozici nepřeberné množství komponent třetích stran. Jedná se o malé, specifické komponenty i rozsáhlé balíky komponent, pomocí kterých lze sestavit komplexní uživatelské aplikace. Zásadní rozdíly jsou taktéž v licenčních podmínkách, podpoře, rychlosti nebo možnostech lokalizace a globalizace. V následující kapitole budou představeny a popsány nejrozšířenější komponenty umožňující tvorbu reportů ve webových technologiích ASP.NET Core, konkrétně v ASP.NET Core MVC a Blazor. Ty budou dále analyzovány a porovnány dle stanovených požadavků. Výběr komponent byl proveden na základě doporučení společnosti Microsoft, popularity ve správci balíčků Nuget Package Manager a pořadí v žebříčku ComponentSource [41] [42] [43].

5.1 DevExpress

Jednou z velmi populárních společností nabízející nástroje a komponenty primárně zaměřené na .NET platformu je společnost DevExpress. Vývoji a podpoře s tímto zaměřením se společnost věnuje od roku 1998 a během této doby vybudovala širokou škálu nabízených produktů i rozsáhlé centrum podpory. Nabídka produktů je rozdělena do několika kategorií. V rámci této podkapitoly budou představeny pouze produkty týkající se tématu této práce. Mezi důležité kategorie produktů patří zejména komponenty uživatelského rozhraní a nástroje pro reporting a dashboardy. Mezi další kategorie patří rozhraní pro práci se soubory Office, Business App Framework a nástroje pro zjednodušení vývoje, testování webových aplikací a monitoring aplikací [44].

Komponenty uživatelského rozhraní jsou nabízeny pro různá cílová zařízení. V nabídce společnosti jsou balíky komponent WinForms, WPF a UWP pro Windows

aplikace a dále balíky komponent zaměřené na webové technologie platformy .NET. Do této kategorie spadají komponenty pro ASP.NET WebForms i MVC, ASP.NET Core, Blazor a klientské komponenty využívající pouze JavaScript. Balík vyvíjený pro technologii ASP.NET Core v sobě obsahuje více než 70 výkonných komponent uživatelského rozhraní společně s řešením reporting platformy. V rámci zpracování a analýzy dat jsou v balíku zahrnuty komplexní komponenty pro tvorbu interaktivních grafů, datových a kontingenčních tabulek, mapy nebo Ganttova diagramu. Dále jsou k dispozici prvky pro vytváření stromových seznamů, vývojových diagramů, formulářů a jejich editačních prvků nebo dialogových oken [44].

Komponenta datové tabulky s názvem Data Grid nabízí pokročilou funkcionalitu a možnosti přizpůsobení. Umožňuje stránkování a řazení dat, pokročilé řádkové filtrování se specifikováním podmínek, dynamické seskupování dat s podporou rozbalení každého řádku i dynamické přesouvání a skrývání sloupců s možností změny jejich velikost. Dále podporuje výpočet celkových sumářů, sumářů skupin nebo export všech či výběru dat do různých formátů. Datový zdroj komponenty může být různého typu. Komponenta umožňuje zpracování dat z pole uloženého v paměti, z AJAX dotazů Web API služeb, SignalR služeb nebo služeb využívajících ODATA protokol. Komponenta Pivot Grid umožňuje vícedimenzionální datovou analýzu. Mezi její klíčové vlastnosti patří, přizpůsobitelné rozložení jednotlivých prvků, datové filtrování i podpora různých režimů výpočtů. Z vykreslovaných dat je možné současně také vykreslit různé druhy grafů. Nabídka v rámci technologie Blazor je více omezená. Balík komponent nyní obsahuje 15 různých prvků s možností nasazení v serverovém i klientském řešení. Jsou dostupné podstatné komponenty jako datová a kontingenční tabulka, grafy nebo formulářové prvky. Jejich funkcionalita je však v některých případech omezená [44] [45].

V rámci webového reportingu dat společnost nabízí kompletní platformu s podporou frameworků .NET i .NET Core. Pro vytváření šablon je k dispozici doplněk do vývojového prostředí Visual Studio nebo nástroj Web Report Designer, který je spouštěn přímo ve webovém prohlížeči uživatele. Oba nástroje umožňují velmi podobným způsobem definovat jednotlivé prvky, jejich pozici, vzhled,

pokročilé výpočty i výrazy popisující plnění dat z datového zdroje. Šablony je následně možné naplnit daty a ve webovém prostředí zobrazit pomocí nástroje Web Document Viewer. Připravené reporty lze v různých formátech exportovat nebo vytisknout [44].

Distribuce komponent probíhá pomocí vlastního instalačního souboru, který při instalaci nainstaluje taktéž rozšíření do Visual Studia nebo pomocí privátního NuGet kanálu. Cena komponent je rozdělena do několika úrovni. V rámci webových technologií je nabízena základní úroveň ASP.NET s ročním poplatkem 999 $. Vyšší úrovní je DXperience, která zahrnuje komponenty i pro Windows aplikace a zdrojové kódy komponent, je nabízena za 1499 $. Nejvyšší úroveň s názvem Universal za 2199 $ mimo jiné zahrnuje i prioritní podporu pro řešení problémů. Komponenty pro Blazor jsou nyní dostupné zdarma [44].

5.2 Telerik

Další společností vyvíjející komplexní balíky komponent pro platformu .NET je společnost Telerik. Ta nabízí produkt s názvem Telerik DevCraft zahrnující .NET komponenty, dále pak JavaScriptové komponenty uživatelského rozhraní a nástroje pro reporting a podporu automatizovaného testování. Kromě tohoto produktu nabízí samostatné JavaScriptové komponenty uživatelského rozhraní Kendo UI umožňující integraci s technologiemi a frameworky jako jsou Angular, React nebo Vue. V neposlední řadě jsou k dispozici také produkty pro technologie JavaServer Pages, mobilní aplikace, Windows aplikace nebo nástroje pro práci s dokumenty. Telerik UI for ASP.NET Core v sobě zahrnuje více než 80 komponent založených na knihovně jQuery pro tvorbu moderního uživatelského rozhraní. Patří mezi ně datové i kontingenční tabulky, formulářové a editační komponenty, více než 16 druhů grafů, dále pak navigační prvky, komponenty pro rozvržení uživatelského rozhraní nebo plánování [46].

Komponenta Grid vykreslující datovou tabulku poskytuje mnoho pokročilé funkcionality. Nabízí podporu stránkování, řazení, řádkového filtrování s pokročilými podmínkami, změnu pozice, šířky i viditelnosti sloupců, seskupení dat, při kterém je možné využít základních agregačních funkcí nebo zobrazení jednotlivých skupin. Komponenta umí využít různé druhy datových zdrojů. Kromě

načtení dat uložených v paměti je možné použít Web API, načtení pomocí AJAX dotazů nebo komunikaci realizovanou pomocí SignalR. Komponenta PivotGrid umožňuje vytvořit dynamickou definovanou kontingenční tabulku. Podporuje vytváření šablon, export dat do formátu PDF nebo XLS, načítání dat uložených v paměti i ze vzdáleného datové zdroje, filtrování a řazení zobrazených sloupců a řádků. Telerik UI for Blazor nabízí více než 30 nativních komponent. Z pohledu reprezentace dat nyní nabízí pouze komponentu Grid, která zatím neobsahuje části pokročilé funkcionality používané v ekvivalentní komponentě pro ASP.NET Core [46].

V rámci produktu Telerik Reporting je umožněno navrhovat šablony pomocí webového nástroje Web Report Designer. Ten nabízí komplexní možnosti úprav vzhledu, pozice a plnění dat z datových zdrojů. Do šablony je možné vložit množství prvků, jako jsou tabulky, grafy, mapy nebo vnořené reporty. V rámci komponent pro ASP.NET Core i Blazor je k dispozici komponenta Report Viewer Control určená pro zobrazení reportu, která zároveň umožňuje jeho tisk nebo export do různých formátů [46].

Komponenty pro jednotlivé technologie je možné zakoupit jednotlivě. Telerik UI ASP.NET MVC nebo ASP.NET Core jsou nabízeny za cenu 1048 $. Telerik UI for Blazor jsou nabízeny za 899 $. V nabídce jsou dále tři cenové úrovně balíčků produktů, které se liší zahrnutou nabídkou produktů a rozsahem nabízené podpory. Nejlevnější úrovní s cenou 1299 $ je DevCraft UI obsahující veškeré komponenty společnosti a základní úroveň podpory s dobou odpovědí 72 hodin a limitem 10 hlášení ročně. Druhá úroveň DevCraft Complete nabízená za částku 1499 $ mimo jiné přidává reportovací platformu Telerik reporting, vyšší úroveň podpory s reakční dobou 24 hodin a neomezený počet hlášených incidentů. Nejvyšší úroveň DevCraft Ultimate v sobě zahrnuje reportovací server, přidává možnost telefonické asistence a eskalace hlášených problémů s vyšší prioritou. Její cena je 2199 $. Produkty jsou licencovány pro každého vývojáře zvlášť s ročním obnovováním [46] [47].

5.3 Syncfusion

Mezi další výrobce komponent pro .NET platformu patří společnost Syncfusion. S více než 19 lety vývoje se řadí mezi přední společnosti, mezi jejíž zákazníky patří například IBM, Visa, Intel nebo Siemens. Vývojářům nabízí více než 1600 komponent pro různé platformy. Ve webových technologiích jsou k dispozici komponenty pro ASP.NET Core, Blazor, ASP.NET MVC a Web Forms, Angular, React a další. V rámci mobilních technologií nabízí řešení pro Xamarin, Flutter i UWP. Komponenty pro desktopové aplikace jsou dostupné pro technologie WinForms, WPF, UWP. Knihovna komponent pro ASP.NET Core v sobě obsahuje více než 70 komponent s širokým zaměřením. Komponenty jsou navrženy s ohledem na výkon i uživatelskou přívětivost s podporou lokalizace a globalizace a jsou rozděleny do několika základních kategorií. Patří mezi ně formuláře, jejich vstupní pole, editory textu, navigační a notifikační prvky, tlačítka, kalendářové a komponenty pro podporu plánování. Dále jsou dostupné komponenty pro tvorbu datových a kontingenčních tabulek a množství komponent pro vizualizaci dat pomocí různých typů grafů nebo teplotní mapy. Pro Blazor je dostupných více než 65 komponent, které plně podporují serverové i klientské nasazení. Mimo jiné jsou k dispozici komponenty DataGrid a Pivot Table i s podporou pokročilých funkcí. Komponenta Pivot Table umožňuje připojení k multidimenzionálním datům poskytovaným službou SSAS. Na základě toho lze vytvářet výstupy používající dimenze, hierarchie, vypočítané členy z OLAP kostky za běhu přímo ve webové aplikaci [49].

Nástroje pro vytváření reportingových šablon nejsou v rámci komponent dostupné. Společnost v tomto směru nabízí dva nezávislé produkty Bold Reports pro reporting a Bold BI pro komplexní řešení business intelligence v rámci společnosti. Pro zobrazování a exportování SSRS, RDL a RDLC reportů je pro všechny platformy nabízena komponenta Bold Report Viewer [50] [51]. Report Definition Language (dále jen RDL) je XML reprezentace definic reportů služeb SQL Server Reporting Services [48].

Společnost nabízí jednodušší cenové úrovně. Při zakoupení jednotlivých technologií je jejich cena 995 $ za vývojáře a rok. V případě zakoupení kompletního

balíku komponent pro všechny podporované technologie je cena 2495 $. Všechny cenové modely zahrnují 24hodinovou podporu, neomezený počet hlášených incidentů, neomezený počet požadavků na nový vývoj nebo vzdálenou podporu. Pro nezávislé vývojáře nebo malé společnosti s méně než pěti vývojáři a hrubým ročním příjmem nepřekračujícím 1 milion dolarů společnost nabízí licenci s názvem Syncfusion Community License. Ta poskytuje volný přístup ke všem produktům společnosti s uživatelskou podporou a aktualizací produktů, které mohou být používány i pro vývoj komerčních aplikací [49].

5.4 Radzen

Společnost Radzen nabízí vlastní WYSIWYG nástroj pro tvorbu webových aplikací využívajících technologie Blazor nebo Angular. Mimo tento produkt společnost vývojářům nabízí samostatnou knihovnu komponent pro Blazor. Jedná se o balík více než čtyřiceti nativních komponent, který je dostupný zdarma. V balíku se nachází množství znovupoužitelných komponent pro zjednodušení vývoje uživatelského rozhraní. Komponenty jsou rozdělené do pěti základních skupin. První z nich jsou obecné komponenty jako jsou tlačítka, hierarchické menu nebo komponenta pro zobrazení notifikací uživateli. Druhou jsou kontejnery pro organizaci uživatelského rozhraní. Do této kategorie spadají komponenty panelů, záložek nebo karet. Třetí a čtvrtá skupina komponent spolu úzce souvisí, jedná se o formulářové prvky a validátory. Ve formulářových prvcích knihovna nabízí řešení pro všechny běžně používané prvky, jakou jsou textové vstupy, dále vstupy pro zadávání čísel, hesel, datumů, odesílání souborů nebo zaškrtávací pole. Zajímavou komponentou je zde DropDownDataGrid, který u rozbalovacího pole umožňuje výběr prvku za pomocí stránkovaného a filtrovatelného gridu. Je tedy možné použít toto rozbalovací pole i pro výběr nad velkým množstvím dat. Díky tomu uživatel snadno vyhledá a vybere požadovanou hodnotu [52].

Mezi validátory v knihovně patří validátor pro vyžadované pole, délku zadaného textu nebo validátor emailové adresy. Poslední skupinou komponent v knihovně jsou komponenty pracující s daty, které jsou z pohledu tématu diplomové práce nejdůležitější. Jedná se o tři komplexní komponenty DataGrid, DataList a Tree. První komponenta DataGrid umožňuje zobrazit komplexní datovou

tabulku. Komponenta podporuje řazení, filtrování a stránkování dat, detailní přizpůsobení každé zobrazené buňky, lokalizaci i globalizaci. Při použití vhodného datového zdroje implementujícího rozhraní IQueryable umožňuje provádět dotazy do datového zdroje na serverové straně, čímž zásadně optimalizuje výkon tabulky. Ke komponentně existuje mnoho hotových pokročilých řešení například pro řádkovou editaci dat nebo implementaci vnořeného gridu. Seskupování dat v komponentě není podporováno. Druhá komponenta pro práci s daty je DataList umožňující pro každý řádek dat zobrazit komplexnější informace. Nejedná se o datovou tabulku, takže jsou zásadně omezeny možnosti práce s daty. Podporováno je pouze stránkování dat na serverové i klientské straně. Poslední komponentou je Tree, která umožňuje zobrazení datového stromu. V komponentě je implementováno několik pokročilých metod pro načítání smíšených dat, načtení a vykreslení pouze částí stromu nebo definování šablon pro různé úrovně stromu. Dříve zmíněná možnost práce s datovým zdrojem implementujícím rozhraní IQueryable na serverové straně je umožněna ve většině komponent pracujících s daty. Seznam všech komponent je dostupný v dokumentaci knihovny [52].

Instalace knihovny komponent do projektu je možná přes NuGet Package Manager bez jakékoliv registrace. Komponenty jsou šířeny zdarma i pro komerční použití. Podpora je v tomto případě pouze v rámci komunity. Společnost v případě zájmu nabízí i placené úrovně podpory, které v sobě však zahrnují i licenci na jejich primární produkt. Roční cena úrovně Professional je 599 $ s dedikovanou podporou a garantovanou odpovědí do 24 hodin. Vyšší úroveň Enterprise pak za roční poplatek 1299 $ nabízí prioritní podporu s reakční dobou 16 hodin. Společnost Radzen svůj produkt vyvíjí od konce roku 2015. Samotné Blazor komponenty jsou dostupné od konce roku 2018 a dle statistik v NuGet Package Manageru je počet stažení vyšší než 41 tisíc [52] [53].

5.5 NReco

Název NReco vychází z názvu .NET Reusable Components. Jedná se o projekt nabízející různorodé komponenty a knihovny pro platformu .NET vyvíjené od roku 2008 týmem ukrajinských vývojářů. Mezi společnosti používající některé z produktů tohoto týmu patří například Microsoft, Samsung nebo Adobe. Produkty

jsou řazeny do tří různých kategorií. První z nich jsou knihovny a technologie pro přístup k datům a jejich zpracování. Druhou kategorií je formátování souborů v .NET technologiích. Třetí kategorií jsou nástroje a služby business intelligence a reportingu dat. V rámci první kategorie přístupu a zpracování dat NReco nabízí například knihovnu NReco Data Access library umožňující výkonný přístup k datům pro platformy .NET a .NET Core. Dále nabízí například Recommendation Engine, rychlý filtrovací engine, který na základě uživatelského chování umožňuje sestavit nabídku relevantních doporučených položek. Do kategorie formátování souborů NReco řadí zejména populární knihovny HTML-to-PDF generator, HTML-to-Image Generator a PDF-to-Image Renderer. První ze jmenovaných knihoven umožňuje jednoduché generování HTML stránek do souborů ve formátu PDF. Knihovna nabízí jednoduchou integraci s aplikacemi využívajícími ASP.NET MVC i .NET Core. Pro vykreslování stránek používá renderovací engine WebKit a vývojářům umožňuje různorodé nastavení výstupu, jako orientace a velikost stránek, šablony pro záhlaví a zápatí stránek nebo automatické generování obsahu [54].

Třetí kategorií projektu NReco jsou nástroje a služby business intelligence a reportingu dat. Do této kategorie spadají následující čtyři produkty: NReco PivotData Toolkit, PivotData Microservice, Javascript Pivot Table a Javascript Query Builder. NReco PivotData Toolkit je sada vývojových nástrojů pro agregaci dat a generování pokročilých kontingenčních tabulek. Implementuje v paměti procesované datové kostky, OLAP operace a umožňuje sestavení i zobrazení kontingenční tabulky ve webovém prostředí. Sestavenou tabulku je zároveň možné v různých formátech exportovat nebo integrovat do dalších aplikací. Nástroj nabízí mnoho pokročilých agregačních funkcí a optimalizací výpočtů. K dispozici je rozšíření BI toolkit, které nabízí další možnosti vizualizace kontingenční tabulky, samotného procesování dat i jejich exportu. Použití je možné jak v aplikacích postavených na platformě .NET i .NET Core, tak ve webových technologiích ASP.NET. Na nástroji NReco PivotData Toolkit je založena mikroslužba s názvem PivotData Microservice. Ta pomocí web API zpřístupňuje pokročilou funkcionalitu pro zpracování dat dalším aplikacím. Mikroslužba využívá platformu .NET Core a je možné ji provozovat ve vlastním prostředí [54].

Každý z nabízených produktů projektem NReco má specifické licenční podmínky. Z pohledu zaměření této diplomové práce budou popsány pouze produkty související se zpracováním a analýzou dat, tedy PivotData Toolkit a PivotData Microservice. Samotný PivotData Toolkit je nabízen zdarma za podmínky, že projekt, ve kterém se nástroj používá, je nasazen pouze na jedné instanci. Zároveň je zde při řešení problémů dostupná pouze komunitní podpora. Rozšířená licence pro nasazení na více instancích společně s roční emailovou podporou je nabízena za cenu 300 $. Rozšíření s názvem BI Toolkit je licencováno v závislosti na počtu vývojářů v několika úrovních. Základní balíček pro maximálně tři vývojáře je nabízen za 299 $. Pro neomezený počet vývojářů s omezením pouze na jednu instanci je nabízena licence s názvem SaaS s cenou 499 $. Další možnosti licencování jsou již vytvářeny na míru zákazníkům [54].

6 Požadavky pro testované komponenty

V rámci testování komponent od vybraných vývojářů a společností byla stanovena hodnotící kritéria jejich nabízené funkcionality. Tyto kritéria vycházejí z obecných požadavků na tvorbu moderních aplikací, které jsou používány uživateli po celém světě. Další hodnotící kritéria vychází ze zkušeností se zakázkovým vývojem a rozvojem komplexních informačních systémů a znalosti běžných požadavků zákazníků na zobrazení a možnosti analýzy dat. Aplikace je nutné lokalizovat, globalizovat a zajistit podporu písem zapisovaných zprava doleva. V rámci lokalizace musí být umožněno nastavení překladů pro daný jazyk a region ve všech uživatelských prvcích komponent. Mezi tyto prvky patří veškeré popisky, texty tlačítek, nápovědy nebo položky v kontextových nabídkách. Globalizace musí nabízet schopnost reagovat na vybranou kulturu, které se musí přizpůsobit formátování textu, zejména číselných hodnot, formát datumu a času nebo formátování měny. Podpora písma zapisovaného zprava doleva je často značena anglickou zkratkou RTL vycházející z názvu Right-to-Left. Tato podpora je podstatná zejména pro arabské a hebrejské písmo.

Dalším parametrem komponent je úroveň zákaznické podpory. Během používání komponent zejména s rozsáhlou škálou funkcionality se při jejich integraci a následném provozu může vyskytnout řada chyb nebo nežádoucích či nedokumentovaných stavů. Tyto problémy je obvykle nutné vyřešit s co nejmenší časovou prodlevou a s jejich řešením by zákaznická podpora měla pomoci. Jedná se o jedno ze základních kritérií, zejména v případě, kdy jsou externí komponenty použity v kritických částech vyvíjené aplikace. Jedním z dalších parametrů, který bude u balíků komponent porovnáván je jejich cena. Společnosti a týmy vývojářů nabízející komponenty nabízí různé cenové úrovně svých produktů v závislosti na cílové platformě, množství zahrnutých komponent a dalších nástrojů i na úrovni podpory. Některé společnosti nabízejí při splnění specifických podmínek své produkty bez poplatku.

S ohledem na aktuálně plánovaný rozvoj platformy .NET budou v rámci hodnocení zahrnuty komponenty pouze pro technologie ASP.NET Core MVC a Blazor. Balíky pro .NET Framework jsou sice díky své delší historii komplexnější,

odladěné a je pro ně k dispozici větší množství již hotových řešení, nicméně z pohledu budoucího vývoje .NET Core a plánovaného vydání .NET 5 není vývoj nových aplikací na .NET Frameworku příliš perspektivní a v rámci této kapitoly nebudou komponenty pro tuto platformu zahrnuty. V analýze komponent bude zohledněna skutečnost, že Blazor je velmi mladá technologie, jejíž klientský mód nasazení je stále ve vývoji. Někteří výrobci nabízí pouze menší sadu komponent s omezeným rozsahem funckionality. Balíky pro ASP.NET Core MVC a Blazor budou hodnoceny zvlášť.

S ohledem na téma diplomové práce budou porovnány komponenty ze tří kategorií. První oblasti je zobrazení a analýza dat, kde byly vybrány dvě nejpoužívanější a velmi komplexní komponenty datové a kontingenční tabulky Data Grid a Pivot Grid. V oblasti vizualizace dat byly vybrány komponenty pro tvorbu plošných, sloupcových, řádkových a liniových grafů.
Pro komponenty Data Grid byla stanovena následující hodnotící kritéria:

- Podpora stránkování a řazení dat
- Podpora řádkového filtrování s možností specifikovat rozšířené filtry
- Podpora dynamického seskupení dat dle vybraného sloupce a možnost rozbalení každé skupiny
- Podpora lokalizace a globalizace
- Možnost exportu dat do PDF či jiného datového formátu
- Možnost dynamického zobrazování, přesouvání a skrývání jednotlivých sloupců

Pro komponenty Pivot Grid byla stanovena tato kritéria:

- Podpora lokalizace a globalizace
- Podpora načtení dat z Microsoft SSAS
- Možnost exportu dat do PDF či jiného datového formátu
- Možnost dynamického nastavení dimenzí, filtrů a řazení

V rámci vizualizace dat byla hodnocena obecná podpora jednotlivých typů grafů a podpora komponenty pro vizualizaci teplotní mapy.

7 Analýza a porovnání komponent

Vybrané komponenty uživatelského rozhraní pro technologie ASP.NET Core MVC a Blazor budou hodnoceny dle požadavků stanovených v předchozí kapitole. Primárním zdrojem informací o komponentách jednotlivých výrobců bude uživatelská dokumentace. Některé z komponent zejména pro technologii Blazor budou pro ověření jejich funkcionality otestovány v praktickém projektu. Vzhledem k jejich rychlému vývoji není v některých případech dokumentace aktuální a může obsahovat pouze základní popis. Další informace budou získávány z webových stránek výrobců a ukázkových (demo) aplikací demonstrujících použití komponent v reálném nasazení. U open source komponent bude jako další zdroj použit jejich repozitář.

7.1 ASP.NET Core MVC

Pro technologii ASP.NET Core MVC byly analyzovány komponenty čtyř výrobců. Komponenty společnosti Radzen jsou v aktuální verzi k dispozici pouze pro technologii Blazor a v rámci této kapitoly nejsou relevantní. V rámci analýzy bude nejprve představena cena a počet komponent v balíku, dále bude v jednotlivých podkapitolách zhodnocena funkcionalita každé analyzované komponenty. Výsledky obecného porovnání jsou shrnuty v Tabulka 1.

Tabulka 1 – Obecné porovnání výrobců komponent pro technologii ASP.NET Core MVC

	Počet komponent	**Cena**
DevExpress	70	999 $
Telerik	80	899 $
Syncfusion	70	Zdarma/995 $ [1)]
Radzen [2)]	-	-
NReco	1	Zdarma/299 $ [3)]

[Zdroj: autor]

1) Komponenty Syncfusion jsou zdarma pro nezávislé vývojáře nebo malé společnosti s méně než pěti vývojáři a hrubým ročním příjmem nepřekračujícím 1 milion dolarů
2) Radzen nenabízí komponenty pro ASP.NET Core MVC
3) Dostupné zdarma pro projekt nasazený na jedné instanci. V případě redistribuce nebo nasazení na více instancích je nutné zakoupit komerční licenci

7.1.1 Komponenta Data Grid

Následující Tabulka 2 shrnuje podporu funkcionality komponenty Data Grid v technologii ASP.NET Core MVC.

Tabulka 2 – Porovnání komponenty Data Grid pro ASP.NET Core MVC

	DevExpress	Telerik	Syncfusion
Stránkování a řazení	Ano	Ano	Ano
Řádkové filtrování	Ano	Ano	Ano
Seskupení dat	Ano	Ano	Ano
Lokalizace a globalizace	Ano	Ano	Ano
Export dat	Ano	Ano	Ano
Dynamická práce se sloupci	Ano	Ano	Ano

[Zdroj: autor]

7.1.2 Komponenta Pivot Grid

V Tabulka 3 je uvedena podpora funkcionality komponenty Pivot Grid v technologii ASP.NET Core MVC.

Tabulka 3 – Porovnání komponenty Pivot Grid pro ASP.NET Core MVC

	DevExpress	Telerik	Syncfusion	NReco
Lokalizace a globalizace	Ano	Ano	Ano	Ano
Načtení dat z Microsoft SSAS	Ano	Ano	Ano	Ano
Export dat	Ano	Ano	Ano	Ano
Dynamické nastavení	Ano	Ano	Ano	Ano

[Zdroj: autor]

7.1.3 Komponenty vizualizace dat

Tabulka 4 obsahuje porovnání komponent vizualizace dat v technologii ASP.NET Core MVC.

Tabulka 4 – Porovnání komponent vizualizace dat pro ASP.NET Core MVC

	DevExpress	Telerik	Syncfusion
Plošné grafy	Ano	Ano	Ano
Sloupcové grafy	Ano	Ano	Ano
Řádkové grafy	Ano	Ano	Ano
Liniové grafy	Ano	Ano	Ano
Teplotní mapy	Ne	Ne	Ano

[Zdroj: autor]

7.2 ASP.NET Core Blazor

V rámci technologie ASP.NET Core Blazor nejsou dostupné komponenty projektu NReco. Komponenty dalších výrobců budou analyzovány dle stejných požadavků stanovených pro technologii ASP.NET Core MVC. Obecné porovnání je shrnuto v Tabulka 5.

Tabulka 5 – Obecné porovnání výrobců komponent pro technologii ASP.NET Core Blazor

	Počet komponent	Cena
DevExpress	15	Dočasně zdarma
Telerik	30	899 $
SyncFusion	65	Zdarma/995 $ [1)]
Radzen	40	Zdarma
NReco [2)]	-	-

[Zdroj: autor]

1) Komponenty Syncfusion jsou zdarma pro nezávislé vývojáře nebo malé společnosti s méně než pěti vývojáři a hrubým ročním příjmem nepřekračujícím 1 milion dolarů
2) NReco nenabízí komponenty pro ASP.NET Core Blazor

7.2.1 Komponenta Data Grid

Tabulka 6 shrnuje funkcionalitu komponenty Data Grid pro ASP.NET Core Blazor.

Tabulka 6 – Porovnání komponenty Data Grid pro ASP.NET Core Blazor

	DevExpress	Telerik	Syncfusion	Radzen
Stránkování a řazení	Ano	Ano	Ano	Ano
Řádkové filtrování	Ano	Ano	Ano	Ano
Seskupení dat	Ano	Ano	Ano	Ne

	DevExpress	Telerik	Syncfusion	Radzen
Lokalizace a globalizace	Ano	Ano	Ano	Ano
Export dat	Ne	Ne	Ano	Ne
Dynamická práce se sloupci	Ano	Ano	Ano	Ne

[Zdroj: autor]

7.2.2 Komponenta Pivot Grid

V Tabulka 7 je popsána funkcionalita komponenty Pivot Grid pro ASP.NET Core Blazor.

Tabulka 7 – Porovnání komponenty Pivot Grid pro ASP.NET Core Blazor

	DevExpress	Telerik [1)]	Syncfusion	Radzen [1)]
Lokalizace a globalizace	Ano	-	Ano	-
Načtení dat z Microsoft SSAS	Ne	-	Ano	-
Export dat	Ne	-	Ano	-
Dynamické nastavení	Ne	-	Ano	-

[Zdroj: autor]

[1)] Telerik a Radzen nenabízí komponentu Pivot Grid pro ASP.NET Core Blazor

7.2.3 Komponenty vizualizace dat

Porovnání funkcionality komponent pro vizualizaci dat je shrnuto v Tabulka 8.

Tabulka 8 – Porovnání komponent vizualizace dat pro ASP.NET Core Blazor

	DevExpress	Telerik	Syncfusion	Radzen [1)]
Plošné grafy	Ano	Ano	Ano	-
Sloupcové grafy	Ano	Ano	Ano	-
Řádkové grafy	Ne	Ano	Ano	-
Liniové grafy	Ano	Ano	Ano	-
Teplotní mapy	Ne	Ne	Ano	-

[Zdroj: autor]

[1)] Radzen nenabízí komponenty vizualizace dat pro ASP.NET Core Blazor

7.3 Zhodnocení komponent

V rámci analýzy komponent je zřejmý velký rozdíl mezi oběma zkoumanými technologiemi. ASP.NET Core MVC je k dispozici od první verze .NET Core vydané v polovině roku 2016. Během této doby získal velkou popularitu nejen u vývojářů .NET platformy ale i jiných technologií. Na tento trend zareagovali i výrobci komponent. Z výsledků analýzy komponent vyplývá, že všichni zahrnutí výrobci nabízejí robustní, spolehlivé komponenty s komplexní funkcionalitou. Komerční výrobci DevExpress, Syncfusion i Telerik zároveň nabízejí balíky s podobným počtem komponent i cenou. Projekt NReco v rámci ASP.NET Core MVC nabízí pouze komponentu Pivot Grid společně s jejími rozšířeními. Zajímavým rozdílem v rámci analýzy je pouze nativní komponenta pro vytvoření teplotní mapy, která je nabízena pouze společností Syncfusion. Další hodnocené komponenty jsou srovnatelné.

Naopak velmi odlišná situace je u nové technologie ASP.NET Core Blazor. Vzhledem k její velmi krátké historii a intenzivnímu vývoji nejsou některé komponenty dostupné. Nabízené komponenty často ve srovnání s jejich alternativami pro ASP.NET Core MVC zahrnují jen část funkcionality. Zároveň jsou pro Blazor mnohem zásadnější rozdíly v ceně komponent. Společnost DevExpress v balíku aktuálně nabízí 15 komponent. Analyzované komponenty nenabízí plnohodnotnou funkcionalitu, avšak společnosti je i s ohledem na tento stav nabízí dočasně zdarma i pro použití v komerčních projektech. Společnost Telerik v balíku komponent pro Blazor zatím vůbec nenabízí komponentu Pivot Grid. Ostatní komponenty nabízí v porovnání s ASP.NET Core MVC omezenou funkcionalitu, přesto je však jejich cena stanovena pro obě technologie stejně. Společnost Syncfusion v balíku komponent zahrnuje 40 komponent. Balík obsahuje všechny analyzované komponenty, které zároveň nabízejí veškerou pokročilou funkcionalitu zkoumanou v rámci této práce. Cena za balík komponent je shodná pro obě technologie. Poslední zkoumaný balík komponent je od společnosti Radzen. Ta nabízí zejména komponenty uživatelského rozhraní. Analyzovaná komponenta Data Grid nabízí pouze část pokročilé funkcionality, avšak použití těchto komponent není zpoplatněno ani pro komerční projekty.

8 Vlastní implementace tabulkového reportu

Součástí této diplomové práce je také návrh a implementace vlastního tabulkového reportu (gridu). Cílem této praktické části práce je vytvořit znovupoužitelnou komponentu, kterou bude možné snadno integrovat do nových či existujících aplikací.

Při návrhu reportu bylo stanoveno několik základních požadavků, které musí komponenta implementovat. Mezi funkční požadavky komponenty bylo zařazeno dynamické vygenerování sloupců dle datového zdroje, podpora stránkování, řazení dat dle všech sloupců v datovém zdroji, podpora seskupení a sumarizování dat. Doplňkovým funkčním požadavkem byla možnost zobrazení jednoduchého sloupcového grafu seskupených dat. Pro komponentu byly zároveň stanoveny dva nefunkční požadavky. Prvním z nich je maximální doba odezvy dvě sekundy pro datový zdroj obsahující 10 000 záznamů pro jakoukoli operaci implementovanou v komponentně. Druhým požadavkem je modulární, udržitelná a rozšiřitelná architektura, která umožní snadnou údržbu a rozvoj komponenty.

Pro implementaci byla zvolena open-source framework .NET Core ve verzi 3.1.1. Jedná se o verzi s prodlouženou dobou podpory do prosince 2022. Použití této verze frameworku je doporučené a vhodné pro všechny projekty s plánovaným dlouhodobým vývojem. Pro vývoj klientské části komponenty byl zvolen framework Blazor v módu Blazor Server. Mód Blazor WebAssembly je v době implementace této diplomové práce pouze ve verzi označené jako Preview. Není tedy oproti módu Server plně podporován a může v něm ještě docházet k zásadním změnám. Při vývoji této komponenty budou taktéž demonstrovány možnosti tohoto nového frameworku. K vykreslování grafů bude použita open-source knihovna ChartJS.Blazor. Jedná se o nadstavbu populární knihovny Chart.JS pro snadné použití v aplikacích využívající Blazor. Použití knihovny ChartJS.Blazor je možné jak v klientském, tak serverovém módu nasazení, do budoucna tedy nebude omezovat možnosti dalšího vývoje [55].

Pro vývoj komponenty bylo použito vývojové prostředí Microsoft Visual Studio 2019 ve verzi Community a databázový server Microsoft SQL Server 2017. Pro otestování možností integrace, kompletní funkcionality a ověření splnění

požadavků komponenty byly využity dvě existující aplikace. V následujících podkapitolách jsou detailně popsány možnosti komponenty, její technické řešení, vybrané implementační detaily a integrace do již existující aplikace. V textu jsou pro zachování přehlednosti zahrnuty a okomentovány pouze vybrané podstatné nebo zajímavé části zdrojového kódu. Kompletní a okomentované zdrojové kódy jsou pak umístěny v příloze této diplomové práce.

8.1 Rozvržení projektu komponenty

Komponenta je rozvržena do dvou základních projektů: GridReport.Core a GridReport.Component. První z projektů v sobě obsahuje definici všech datových struktur, rozhraní a aplikační logiku potřebnou pro operace s daty, jejich stránkování, seskupení a rozšiřující metody pro vykreslení samotné tabulky a podpůrných komponent. Podstatné části služeb, rozhraní a definic dat budou detailně popsány v následující kapitole Aplikační logika. Rozvržení jednotlivých souborů v projektech je znázorněno na Obrázek 11.

Druhý projekt GridReport.Component obsahuje samotné Razor komponenty vykreslující tabulku. Tabulka je sestavena z hierarchicky umístěných komponent a jejich struktura je podobná jako u běžných tabulek. Hlavní komponentou je GridComponent, která zajišťuje zpracování dat, vstupů od uživatele, vykreslení základní obálky tabulky a předání dat do komponent na nižší úrovni. Tou je komponenta GridRow reprezentující řádek tabulky. Komponenta GridRow zajišťuje vykreslení daného řádku a buněk pro všechny sloupce v tomto řádku. Pro vykreslení buňky se používá komponenta GridCell. V její metodě GetObjectValue() se provede kompilace výrazového stromu z lambda výrazu nesoucího ukazatel na danou proměnnou, která má být v tomto sloupci zobrazena. Kompilací se provede vyhodnocení tohoto výrazu, jejímž výsledkem je požadovaná hodnota.

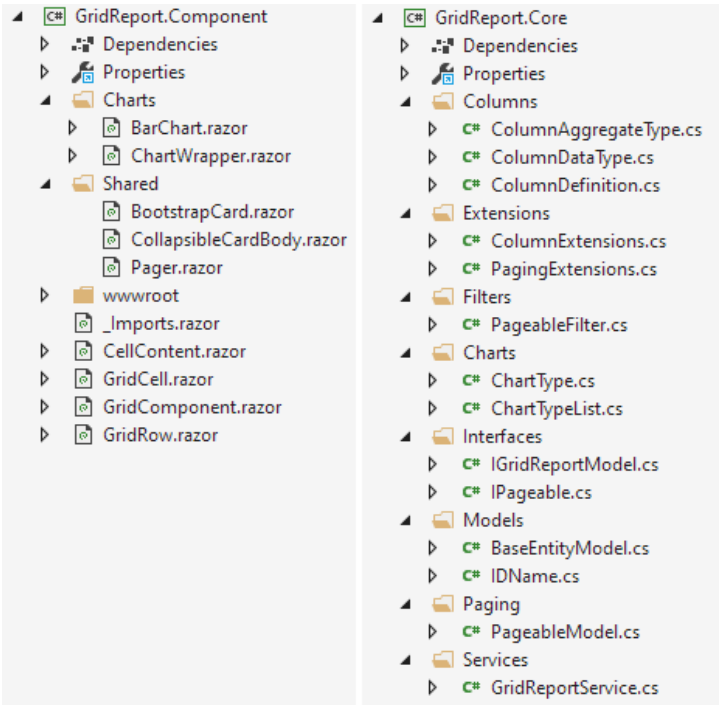
GridReport.Component/GridCell.razor.cs:

```
[Parameter]
public ColumnDefinition<TItem> Column { get; set; }
[Parameter]
public TItem RowItem { get; set; }

public object GetObjectValue()
{
     return Column.Property.Compile()(RowItem);
}
```

Poslední Razor komponentou na nejnižší úrovni je CellContent. Ta zajišťuje formátování a vykreslení hodnoty ve specifikovaném datovém typu sloupce.

Mimo tyto základní komponenty tabulky jsou v projektu použity tři další komponenty. První je komponenta Pager zajišťující vykreslení seznamu stránek a zpracování uživatelských požadavků na změnu stránky nebo načtení první či poslední stránky. Další dvě komponenty slouží pro zpřehlednění uživatelského rozhraní. Jedná se komponenty BootstrapCard a CollapsibleCardBody. První z nich vykresluje stylovaný panel uživatelského rozhraní, který je pro zpřehlednění stránky možné sbalit do úzkého pruhu. Pro podporu sbalení a rozbalení obsahu uživatelského rozhraní je zde použita podpůrná komponenta CollapsibleCardBody. Sbalení a rozbalení obsahu je zde implementováno pouze v zdrojovém kódu C# bez použití jazyka JavaScript, přesto však stejně jako v ostatních komponentách dochází k dynamickému překreslení části stránky bez jejího kompletního přenačtení.



Obrázek 11 – Rozvržení souborů komponenty [Zdroj: autor]

8.2 Aplikační logika

Jedním ze základních prvků komponenty GridReport je generická třída ColumnDefinition<T>, která obsahuje vlastnosti daného sloupce výsledné tabulky. U sloupce lze definovat devět následujících vlastností:

- ColumnId (int) – číselný identifikátor daného sloupce
- LocalizedName (string) – lokalizovaný název sloupce zobrazený v hlavičce tabulky
- Property (Expression<Func<T, object>>) – Pevně typovaný lambda výraz jako datová struktura ve formě výrazového stromu ukazující na proměnnou datového zdroje zpracovávanou v daném sloupci
- CssClass (string) – CSS třída pro použití v buňkách sloupce
- AllowGrouping (bool) – příznak určující, zdali je možné dle daného sloupce data seskupit
- AllowSorting (bool) – příznak určující, zdali je možné dle daného sloupce data řadit
- AlwaysHidden (bool) – příznak určující, zdali má být sloupec ve výsledné tabulce vždy skrytý. V tuto chvíli se používá pro systémové sloupce a do budoucna je možné jeho využití pro implementaci na sobě závislých sloupců
- DataType (ColumnDataType) – datový typ daného sloupce
- AggregateType (ColumnAggregateType) – použitá agregační funkce na datech v daném sloupci. Její použití je možné pouze v případě, že sloupec má povolený příznak seskupení dat a jeho datový typ je číslo nebo desetinné číslo

Komponenta umožňuje definovat sloupce pěti datových typů. Typy jsou vyjmenovány ve výčtu ColumnDataType a jedná se o:

- String – text
- Number – číslo
- Decimal – desetinné číslo
- DateTime – datum a čas
- Percent – procenta

Implementované agregační funkce dat ve sloupci jsou definovány ve výčtu ColumnAggregateType:

- Min – minimum
- Max – maximum
- Sum – suma
- Avg – průměr

Mimo tyto čtyři agregační funkce se ve výčtu ještě nachází hodnota None pro specifikování výchozí hodnoty, která určuje, že na daném sloupci není možné agregační funkce provádět.

Pro definování typu vykreslovaného grafu je definovaný výčet ChartType. Ten nyní obsahuje pouze hodnotu Bar definující sloupcový graf. Třída ChartTypeList umožňuje načtení všech typů grafů, které komponenta podporuje.

V projektu jsou dále implementovány rozšiřující metody pro podporu stránkování, metody pro zpracování dat ve sloupcích a dále několik rozhraní a modelů používaných v komponentně. Jedním z modelů generická třída PageableModel, který obsahuje vlastnosti uživatelského rozhraní pro reprezentaci stránkovaného seznamu a pro možnost vygenerování komponenty umožňující uživateli přepínání jednotlivých stránek. Pro samotné filtrování datového zdroje je implementována statická třída PagingExtensions s generickou metodou Paged<T>. Ta dle poskytnutého filtru provede dotaz do datového zdroje a zároveň vrátí celkový počet prvků v datovém zdroji.

GridReport.Core/Paging/PageableModel.cs:

```
public class PageableModel<T>
{
    public int PageSize { get; set; }
    public int TotalPages => PageSize > 0 ? (int)Math.Ceiling(Count /
(double)PageSize) : PageSize;
    public int Count { get; set; } = 0;
    public IEnumerable<T> Data { get; set; } = new List<T>();
}
```

GridReport.Core/Extensions/PagingExtensions.cs:

```
public static async Task<PageableModel<T>> Paged<T>(this IQueryable<T> query, IPageab
le filter)
{
    return new PageableModel<T>
    {
        Count = await query.CountAsync(),
        PageSize = filter.PageSize,
        Data = await query.Skip((filter.PageNumber - 1) * filter.PageSize)
.Take(filter.PageSize).ToListAsync()
    };
}
```

Datový zdroj, který je komponentně předáván z aplikace musí implementovat jednoduché rozhraní IGridReportModel. Toto rozhraní v sobě definuje dvě základní vlastnosti Id a Cnt. První z nich je identifikátor daného řádku datového zdroje libovolného datového typu. Druhý z nich je celočíselná proměnná s názvem Cnt, kterou není nutné definovat. Tato proměnná se používá uvnitř komponenty pro podporu seskupování dat a pro vykreslení sloupce s počtem dat v každé skupině.

Komponenta uživatelského rozhraní je pak definována jako generická s podmínkou, že datový typ vstupující do komponenty musí implementovat rozhraní IGridReportModel. Generické komponenty je v Blazoru možné definovat pomocí direktivy @typeparam Titem. V projektu se používá oddělení aplikační logiky od uživatelského rozhraní, známe již z technologie Web Forms pod názvem code-behind. To spočívá ve vytvoření partial třídy se stejným názvem jako je soubor aplikačního rozhraní. Code-behind třída pro komponentu GridComponent.razor má tedy název souboru GridComponent.razor.cs. Vytváření partial code-behind tříd je možné až od aktuální verze ASP.NET Core 3.1. Definice generické třídy code-behind je v tomto případě následující:

GridReport.Component/GridComponent.razor.cs:

```
public partial class GridComponent<TItem> where TItem: class, IGridReportModel
```

Ve zdrojovém kódu je definováno několik vlastností označených atributem [Parameter]. Hodnoty těchto vlastností mohou být definovány z rodičovské komponenty. V případě komponenty GridReport se jedná o hodnoty, které musí být poskytnuty z externí aplikace a komponenta se na základě těchto hodnot vykreslí. Jedná se o vlastnosti:

- Columns (IList<ColumnDefinition<TItem>> Columns) – seznam definic sloupců, které komponenta zpracovává a zobrazuje
- TableId (string) – identifikátor vykreslené tabulky
- Data (IQueryable<TItem>) – generický datový zdroj implementující rozhraní IQueryable
- AllowPaging (bool) – příznak určující, zdali je možné povolit stránkování. V případě hodnoty false jsou vykreslena všechna data v datovém zdroji
- TableCssClass (string) – CSS třída pro použití v tabulce
- ShowHeader (bool) – příznak určující, zdali se v tabulce má vykreslit její záhlaví
- ShowFooter (bool) – příznak určující, zdali se v tabulce má vykreslit její zápatí
- AllowGrouping (bool) – příznak určující, zdali je možné povolit seskupení dat
- ShowChart (bool) – příznak určující, zdali se v komponentě má při seskupení dat vykreslit graf

Při použití komponenty je ještě nutné definovat generický datový typ TItem. Další zanořené komponenty tabulky jsou taktéž definovány jako generické se stejnými podmínkami. Díky použití generičnosti je komponenta velmi obecná a pro práci s daty i vykreslení tabulky i dalších komponent nepotřebuje žádné další informace o zobrazovaných datech.

Uživatelské rozhraní komponenty je rozděleno do několika částí nazývaných panely. Prvním panelem je Nastavení, které v případě povolení seskupení dat umožňuje vybrat sloupec umožňující seskupení dat a data dle toho sloupce seskupit.

Tento panel je možné pro zpřehlednění stránky sbalit. Druhým panelem je pak samotná datová tabulka. V hlavičce tabulky jsou vykresleny lokalizované názvy sloupců. Pokud je v definici daného sloupce povolené řazení, je jeho název vykreslený jako aktivní komponenta. Po kliknutí na název dojde k seřazení dat dle tohoto sloupce. Samotná hlavička je vykreslena pouze v případě, že je její vykreslení povolené v definici komponenty. Následně jsou vykreslena data pro danou stránku. Na konci tabulky je vykreslena patička tabulky. V případě, že daný sloupec má povolenou agregační funkci Sum, je v odpovídající buňce patičky vykreslen součet hodnot zobrazených v tomto sloupci. V opačném případě je i v patičce vykreslen lokalizovaný název sloupce. Na konci panelu je pak v případě nastavení seskupení dat vykreslen požadovaný graf.

8.3 Integrace s existující aplikací

Použití komponenty v existující aplikaci je jednoduché a nevyžaduje zásadní úpravy ve zbytku aplikace. Základním předpokladem pro použití komponenty je použití technologie Blazor. Vývojář musí nejprve ve zdrojovém kódu aplikace definovat seznam sloupců, které bude report zpracovávat. Pro každý sloupec je nutné definovat jeho identifikátor, lokalizovaný název, ukazatel na vlastnost v datovém zdroji pomocí lambda expression a datový typ sloupce. Dále je možné definovat informace o tom, jestli má sloupec být zobrazený v reportu, možnosti agregační funkce, CSS třídu a příznak nesoucí informaci o možnosti řazení dat dle tohoto sloupce. Zároveň lze pomocí příznaku AllowGrouping nastavit možnost seskupení dat dle daného sloupce. Následně je nutné definovat datový zdroj komponenty. Ta očekává předání generického seznamu implementujícího rozhraní IQueryable. Rozhraní při použití společně s nástrojem objektově-relačního mapování Entity Framework Core umožňuje přímé dotazování na data uložená v databázi. Komponenta tento způsob dotazování používá pro optimalizaci a zrychlení stránkování, řazení a seskupení zobrazovaných dat. V Razor komponentách je nutné nejprve přidat direktivu @using do souboru _Imports.razor pro možnost použití GridReport komponenty ve všech komponentách aplikace, případně tuto direktivu přidat pouze do specifické komponenty či některého modulu aplikace. Následně je možné GridReport komponentu pomocí definice <GridComponent /> použít.

V definici komponenty lze specifikovat několik povinných a volitelných parametrů, které byly detailně popsány v předchozí kapitole. Integraci lze provést dle následujících částí kódu. V Příloze č. 1 je zobrazeno uživatelské rozhraní komponenty s výchozím stránkovaným zobrazením dat v tabulce. V Příloze č. 2 je zachyceno uživatelské rozhraní při seskupení dat dle vybraného a zobrazení sloupcového grafu těchto seskupených hodnot.

DPReporting/Shared/_Imports.razor:

```
@using GridReport.Component
```

DPReporting/Pages/DataGrid.razor:

```
<GridComponent Data="OrderItems" AllowPaging="true" Columns="GridColumns"
        TableId="OrderItemsTable" ShowHeader="true" ShowFooter="true"
        TItem="DPReporting.DataAccess.Entities.Orders.OrderItemEntity"
        TableCssClass="table table-bordered table-striped"
        AllowGrouping="true" ShowChart="true" />
```

DPReporting/Pages/DataGrid.razor.cs:

```
GridColumns = new List<ColumnDefinition<OrderItemEntity>>
{
        new ColumnDefinition<OrderItemEntity>()
        {
            ColumnId = 1,
            LocalizedName = "ID",
            Property = entity => entity.Id,
            DataType = ColumnDataType.Number,
            AllowSorting = true
        },
        new ColumnDefinition<OrderItemEntity>()
        {
            ColumnId = 2,
            LocalizedName = "Název položky",
            Property = entity => entity.ServiceItemDisplayName,
            DataType = ColumnDataType.String,
            AllowGrouping = true
        },
        ...
        new ColumnDefinition<OrderItemEntity>()
        {
            ColumnId = 10,
            LocalizedName = "Cena",
            Property = entity => entity.PriceTotal,
            DataType = ColumnDataType.Decimal,
            AllowGrouping = true,
            AggregateType = ColumnAggregateType.Sum,
            AllowSorting = true
        }
};

OrderItems = _reportService.GetOrderItems();
```

8.4 Shrnutí a budoucí rozvoj

Komponenta GridReport byla po dokončení implementace integrována s interní testovací aplikací, kde byla zároveň otestována její funkcionalita a provedena kontrola splnění zadaných funkčních požadavků. Komponenta zároveň byla integrována do komerční aplikace autora, jejíž první verze se nyní nachází v produkčním nasazení. Základní funkcionalita komponenty je pro dané potřeby odladěná a funkční. Komponenta splňuje všechny funkční i nefunkční požadavky stanovené v úvodu kapitoly. Rychlost zpracování a vykreslení dat komponentou je úzce závislá na poskytnutém datovém zdroji aplikací.

Na základě interního testování a požadavků zákazníka bylo popsáno několik scénářů, které komponenta nepokrývá a budou implementovány v další etapě vývoje. Řadí se mezi ně zejména podpora více typů vykreslovaných grafů. Druhým požadavkem je rozšíření definice komponenty o možnost nastavení detailního naformátování výstupu dat vykreslovaného do každé buňky tabulky. Třetím požadavkem je přidání možnosti podmíněného formátování buněk tabulky pro možnost vytvoření teplotní mapy. Díky komponentovému návrhu architektury a oddělení aplikační logiky od uživatelského rozhraní nebude žádný z těchto požadavků znamenat zásadní úpravy stávajícího zdrojového kódu, ale pouze jeho rozvoj. Pro přidání nového typu grafu bude nutné rozšířit výčet ChartType, který obsahuje seznam podporovaných typů grafů a o tento nový typ grafu rozšířit taktéž seznam dostupných typů grafů ChartTypeList. Následně je nutné vytvořit novou uživatelskou komponentu, ve které probíhá vytvoření definice a nastavení tohoto nového typu grafu společně s jeho uživatelským rozhraním. Úpravy v uživatelském rozhraní samotné GridReport komponenty nebudou nutné. Rozšíření možností obsahu buněk tabulky bude možné implementovat přidáním nové vlastnosti výrazového stromu typu Expression do definice sloupce tabulky. Následně bude nutné rozšířit komponent GridCell a CellContent o kompilaci, provedení a vykreslení hodnoty dané vlastnosti. Tímto způsobem bude možné z externích aplikací definovat lokalizované hodnoty v buňkách, specifikovat textové reprezentace výčtových typů nebo vyhodnocovat libovolné podmínky například při překročení definované hranice číselných hodnot. Poslední rozšíření formátování buněk může

být realizováno taktéž pomocí nové vlastnosti výrazového stromu typu Expression, kde její komplice bude prováděna v komponentě GridCell. Zde následně může být definována CSS třída dané vykreslované buňky.

9 Závěry a doporučení

Hlavním cílem této diplomové práce bylo představit a popsat možnosti realizace komplexních reportů dat ve webových technologiích platformy Microsoft .NET. V teoretické části práce byla nejprve popsána platforma .NET a její technologie umožňující vývoj serverových i klientských webových aplikací. V rámci této kapitoly byly představeny koncepty starších technologií této platformy, které ovlivnily a stále ovlivňují její vývoj. Dále byl shrnut aktuální stav platformy a její plánovaný budoucí vývoj. Následující kapitola představila termín business intelligence, který reporting dat zastřešuje. Následně byly popsány používané možnosti reprezentace a vizualizace dat ve webových technologiích. V rámci této kapitoly byly taktéž popsány další řešení business intelligence společnosti Microsoft, které lze integrovat do webových aplikací. Poslední kapitola teoretické části práce zahrnovala představení balíků znovupoužitelných komponent pro platformu .NET.

V rámci praktické části diplomové práce byly komponenty různých výrobců analyzovány a porovnány podle stanovených požadavků na jejich funkcionalitu. Hodnotící kritéria byla stanovena s ohledem na jejich použití v komplexních webových aplikacích používaných uživateli po celém světě. Byl stanoven důraz na možnosti lokalizace, globalizace, ale i na pokročilou funkcionalitu, jako dynamické seskupování dat nebo filtrování dat dle uživatelsky definovaných pravidel. Z porovnání vyplynuly podstatné rozdíly v nabídce komponent pro technologie ASP.NET Core MVC a Blazor. Pro první ze zmíněných technologií jsou dostupné robustní komponenty s komplexní funkcionalitou od všech zahrnutých výrobců. Tato skutečnost je dána zejména dlouhodobě vyvíjenou a podporovanou technologií, která si získala oblibu v široké komunitě vývojářů. Odlišná situace je u technologie Blazor. S ohledem na její velmi krátkou historii a stále probíhající intenzivní vývoj je dostupné pouze omezené množství komponent. Někteří výrobci, jejichž komponenty byly analyzovány, zatím nestihli zareagovat na rychlý rozvoj této technologie a nabízí často pouze omezené sady komponent se základní funkcionalitou. Porovnání komponent poskytuje zejména technický pohled na jejich poskytovanou funkcionalitu. Při výběru komponent pro použití ve vyvíjených

aplikacích je nutné zohlednit i další hlediska, jako je cena komponent, jejich licenční podmínky i výhodnost a riziko vyplývající z jejich použití.

Součástí praktické části práce byla rovněž implementace vlastní znovupoužitelné komponenty tabulkového reportu. Pro komponentu byly stanoveny podobné požadavky jako pro ekvivalentní komponenty jiných výrobců analyzovaných v předchozí kapitole. Vlastní komponenta byla implementována pomocí nové, rychle se rozvíjející a perspektivní technologie ASP.NET Blazor, která využívá výhody nového webového standardu WebAssembly. V rámci popisu komponenty tak byly výhody této technologie představeny teoreticky i prakticky. Pro komponentu byly nejprve stanoveny funkční a nefunkční požadavky, které implementuje. Řadí se mezi ně například možnosti seskupení dat nebo možnost vykreslení grafu. Komponenta byla navržena a vyvinuta jako snadno rozšiřitelná. Zároveň byl kladen velký důraz na její snadnou integraci do dalších aplikací. Možnosti komponenty a způsob integrace byly popsány v podkapitole této části diplomové práce. Komponenta byla po dokončení vývoje otestována a v současné době je používána v komerční aplikaci autora a je možné ji snadno použít i v dalších projektech.

10 Seznam použité literatury

[1] TROELSEN, Andrew W. a Philip JAPIKSE. *Pro C# 7: with .net and .net core*. Eighth edition. New York, NY: Apress, [2017]. ISBN 978-1-4842-3017-6.

[2] TROELSEN, Andrew a Philip JAPIKSE. *C# 6.0 and the .NET 4.6 Framework*. Seventh edition. New York, NY: Apress, [2015]. ISBN 978-1-4842-1333-9.

[3] IronPython.net / . *IronPython.net /* [online]. Copyright © [cit. 18.02.2020]. Dostupné z: https://ironpython.net/

[4] PENBERTHY, William. Beginning ASP.NET for Visual Studio 2015. Indianapolis: John Wiley, 2016. ISBN 978-1-119-07742-8.

[5] .NET Core is the Future of .NET | .NET Blog. DevBlogs - Microsoft Developer Blogs [online]. Dostupné z: https://devblogs.microsoft.com/dotnet/net-core-is-the-future-of-net/

[6] FREEMAN, Adam. Pro ASP.NET MVC 5 platform. New York: Apress Media, 2014. Expert's voice in Web development. ISBN 978-1-4302-6541-2.

[7] MUNRO, Jamie. ASP.NET MVC 5 with bootstrap and knockout.js: building dynamic, responsive web applications. Sebastopol: O'Reilly Media, [2015]. ISBN 978-149-1914-397.

[8] PRICE, Mark J. *C# 8.0 and .NET Core 3.0: Modern Cross-Platform Development*. Fourth Edition. Birmingham, UK: Packt Publishing, 2019. ISBN 978-1-78847-812-0.

[9] FREEMAN, Adam. Pro ASP.NET Core MVC 2. Seventh edition. London: Apress, [2017]. ISBN 978-1-4842-3149-4.

[10] Announcing .NET Core 1.0 | .NET Blog. *DevBlogs - Microsoft Developer Blogs* [online]. Dostupné z: https://devblogs.microsoft.com/dotnet/announcing-net-core-1-0/

[11] Introducing .NET 5 | .NET Blog. *DevBlogs - Microsoft Developer Blogs* [online]. Dostupné z: https://devblogs.microsoft.com/dotnet/introducing-net-5/

[12] TechEmpower Framework Benchmarks. *TechEmpower* [online]. Dostupné z:

https://www.techempower.com/benchmarks/#section=test&runid=8ca46892-e46c-4088-9443-05722ad6f7fb&hw=ph&test=plaintext

[13] Making a tiny .NET Core 3.0 entirely self-contained single executable - Scott Hanselman. Scott Hanselman - Coder, Blogger, Teacher, Speaker, Author [online]. Copyright © Copyright 2019, [cit. 23.02.2020]. Dostupné z: https://www.hanselman.com/blog/MakingATinyNETCore30EntirelySelfcontainedSingleExecutable.aspx

[14] JOSHI, Bipin. *Beginning Database Programming Using ASP.NET Core 3: With MVC, Razor Pages, Web API, jQuery, Angular, SQL Server, and NoSQL*. Berkeley, CA: Apress, 2019. ISBN 978-1-4842-5508-7.

[15] Introduction to Razor Pages in ASP.NET Core | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 01.03.2020]. Dostupné z: https://docs.microsoft.com/en-gb/aspnet/core/razor-pages/?view=aspnetcore-3.1&tabs=visual-studio

[16] HIMSCHOOT, Peter. Blazor revealed: building web applications in .net. New York, NY: Springer Science Business Media, 2019. ISBN 978-148-4243-428.

[17] GALLANT, Gerard. WebAssembly in Action: WITH EXAMPLES USING C++ AND EMSCRIPTEN. Shelter Island, NY: Manning Publications Co., 2019. ISBN 978-1-61729-574-4.

[18] What is WebAssembly? The next-generation web platform explained | InfoWorld. *InfoWorld - Technology insight for the enterprise* [online]. Copyright © 2020 [cit. 24.02.2020]. Dostupné z: https://www.infoworld.com/article/3291780/what-is-webassembly-the-next-generation-web-platform-explained.html

[19] LITVINAVICIUS, Taurius. Exploring Blazor: Creating Hosted, Server-side, and Client-side Applications with C#. New York, NY: Apress, 2019. ISBN 978-1-4842-5445-5.

[20] Blazor | Build client web apps with C# | .NET. *.NET | Free. Cross-platform. Open Source.* [online]. Dostupné z: https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor

[21] KNAFLIC, Cole Nussbaumer. *Storytelling with data: let's practice!*. Hoboken, New Jersey: John Wiley, 2015. ISBN 978-1119002253.

[22] SAYED, Asif. Client-Side Reporting with Visual Studio. Berkeley, CA: Apress, 2007. ISBN 978-1-59059-854-2.

[23] NOGUÉS, Albert a Juan VALLADARES. *Business Intelligence Tools for Small Companies: A Guide to Free and Low-Cost Solutions*. Berkeley, CA: Apress, 2017. ISBN 978-1-4842-2567-7.

[24] SHERIF, Ahmed. Practical Business Intelligence. Birmingham, UK: Packt Publishing, 2016. ISBN 978-1-78588-543-3.

[25] GÁLA, Libor, Jan POUR a Zuzana ŠEDIVÁ. Podniková informatika. 2., přeprac. a aktualiz. vyd. Praha: Grada, 2009. Expert (Grada). ISBN 978-80-247-2615-1.

[26] BENOIT, Gerald. Introduction to information visualization: transforming data into meaningful information. Rowman & Littlefield: Lanham, Maryland, [2019]. ISBN 978-153-8118-351.

[27] SOSULSKI, Kristen. Data visualization made simple: insights into becoming visual. New York, NY: Routledge, 2019. ISBN 978-1-138-50387-8.

[28] WILLIAMS, Steve a Nancy WILLIAMS. *The profit impact of business intelligence*. Boston: Elsevier/Morgan Kaufmann, 2007. ISBN 978-0-12-372499-1.

[29] KIMBALL, Ralph a Margy ROSS. The data warehouse toolkit: the definitive guide to dimensional modeling. 3rd ed. Indianapolis: Wiley, c2013. ISBN 978-1-118-53080-1.

[30] ZHANG, Chao a Jiawei HAN. Multidimensional Mining of Massive Text Data. Morgan & Claypool, 2019. ISBN 978-1681735207.

[31] NAGABHUSHANA, S. Data Warehousing: OLAP and Data Mining. New Delhi: New Age International (P) Ltd., Publishers, 2006. ISBN 978-81-224-2705-9.

[32] THOMSEN, Erik. OLAP Solutions: Building Multidimensional Information Systems. Second Edition. New York, NY: John Wiley, 2002. ISBN 0-471-40030-0.

[33] Perceptual Scaling of Map Symbols | Making Maps: DIY Cartography. Making Maps: DIY Cartography | Resources and Ideas for Making Maps [online]. Dostupné z: https://makingmaps.net/2007/08/28/perceptual-scaling-of-map-symbols/

[34] Editions and supported features of SQL Server 2019 - SQL Server | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 08.03.2020]. Dostupné z: https://docs.microsoft.com/cs-cz/sql/sql-server/editions-and-components-of-sql-server-version-15?view=sql-server-ver15

[35] GORMAN, Kellyn, Allan HIRT, Dave NODERER a kol. Introducing Microsoft SQL Server 2019. Birmingham, UK: Packt Publishing, 2019. ISBN 978-1-83882-621-5.

[36] KOROTKEVITCH, Dmitri. Expert SQL Server In-Memory OLTP. 2nd ed. Berkeley, CA: Apress, 2017. ISBN 978-1-4842-2771-8.

[37] RAD, Reza. Pro Power BI Architecture. New York, NY: Springer Science Business Media, 2018. ISBN 978-148-4240-144.

[38] Power BI Architecture: A Complete Tutorial - Mindmajix. Online Certification Training | Corporate Training - Mindmajix [online]. Copyright © 2020 Mindmajix Technologies Inc. All Rights Reserved [cit. 19.03.2020]. Dostupné z: https://mindmajix.com/power-bi-architecture

[39] Power BI | Microsoft Power Platform. [online]. Copyright © 2020 Microsoft [cit. 08.03.2020]. Dostupné z: https://powerbi.microsoft.com/cs-cz/

[40] LARSON, Brian. Data Analysis with Microsoft Power BI. McGraw-Hill Education, 2020. ISBN 978-1-26-045862-6.

[41] Blazor | Build client web apps with C# | .NET. .NET | Free. Cross-platform. Open Source. [online]. Dostupné z: https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor

[42] NuGet Gallery | Packages matching report components . NuGet Gallery | Home [online]. Dostupné z: https://www.nuget.org/packages?prerel=false

[43] Components / ASP.NET Controls - Best Selling. ComponentSource - Software Superstore for Developers & IT Pros [online]. Copyright © 1996

[cit. 08.03.2020]. Dostupné z: https://www.componentsource.com/search/products?f[0]=ft%3A710890&f[1]=at%3A913900&f[2]=at%3A913564&ha=1&sort=bestseller

[44] .NET UI Controls for Developers of Mobile, Desktop, Web & Reporting Apps. .NET UI Controls for Developers of Mobile, Desktop, Web & Reporting Apps [online]. Copyright © 1998 [cit. 08.03.2020]. Dostupné z: https://www.devexpress.com/

[45] KIMMEL, Paul, Joe KUNK a Julian BUCKNALL. Professional DevExpress ASP.NET controls. Indianapolis, IN: Wiley Publishing, 2010. Programmer to programmer. ISBN 04-705-0083-2.

[46] Telerik UI for ASP.NET AJAX, MVC, Core, Xamarin, Angular, HTML5 and jQuery . Telerik UI for ASP.NET AJAX, MVC, Core, Xamarin, Angular, HTML5 and jQuery [online]. Copyright © 2020, Progress Software Corporation and [cit. 08.03.2020]. Dostupné z: https://www.telerik.com/

[47] PAZ, José Rolando Guay. Pro Telerik ASP.NET and Silverlight Controls: Master Telerik Controls for Advanced ASP.NET and Silverlight Projects. New York: Apress, [2010]. ISBN 978-1-4302-2940-7.

[48] Report Definition Language - SQL Server Reporting Services (SSRS) | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 08.03.2020]. Dostupné z: https://docs.microsoft.com/en-us/sql/reporting-services/reports/report-definition-language-ssrs?view=sql-server-ver15

[49] .NET, Xamarin, JavaScript, Angular UI components | Syncfusion. .NET, Xamarin, JavaScript, Angular UI components | Syncfusion [online]. Copyright © 2001 [cit. 08.03.2020]. Dostupné z: https://www.syncfusion.com/

[50] Business Dashboard Software & Analytics Platform | Bold BI. Business Dashboard Software & Analytics Platform | Bold BI [online]. Dostupné z: https://www.boldbi.com/

[51] Report Management System and Reporting Tools | Bold Reports. Report Management System and Reporting Tools | Bold Reports [online]. Dostupné z: https://www.boldreports.com/

[52] Rapid Application Development for the Web | Radzen. Rapid Application Development for the Web | Radzen [online]. Copyright © 2016 [cit. 08.03.2020]. Dostupné z: https://www.radzen.com/

[53] Free Blazor Components | 40+ controls by Radzen. Free Blazor Components | 40+ controls by Radzen [online]. Dostupné z: https://blazor.radzen.com/

[54] NReco: .NET REusable COmponents. NReco: .NET REusable COmponents [online]. Copyright © [cit. 08.03.2020]. Dostupné z: https://www.nrecosite.com/

[55] GitHub - mariusmuntean/ChartJs.Blazor: Brings ChartJs charts to Blazor. The world's leading software development platform · GitHub [online]. Copyright © 2020 GitHub, Inc. [cit. 08.03.2020]. Dostupné z: https://github.com/mariusmuntean/ChartJs.Blazor

11 Přílohy

1) Komponenta GridReport – výchozí stránkované zobrazení dat v tabulce
2) Komponenta GridReport – seskupení dat se zobrazením grafu

1) Komponenta GridReport – výchozí stránkované zobrazení dat v tabulce

Report objednávek

Nastavení

Seskupení dat -- --

Nastavit seskupení

Typ grafu -- --

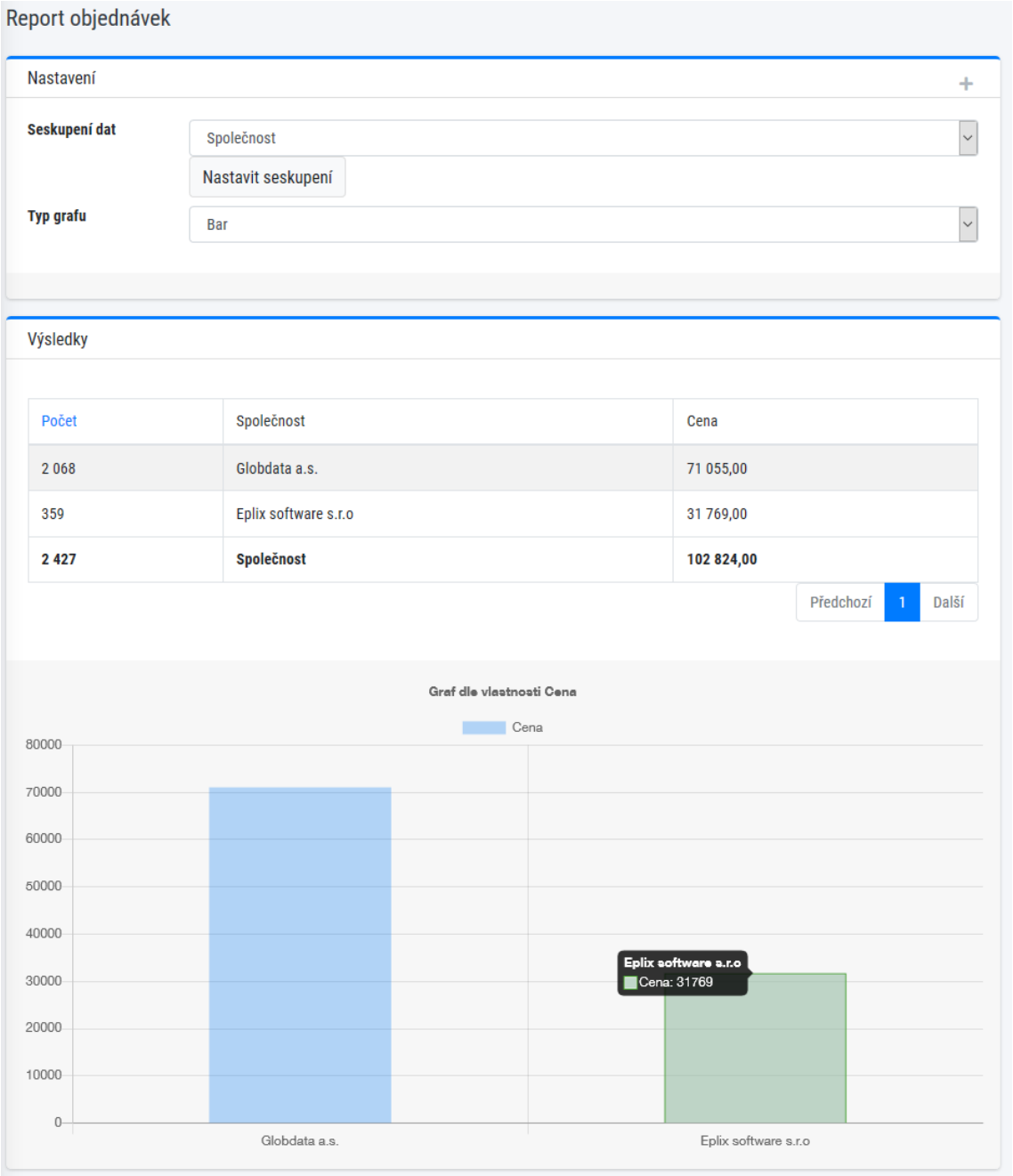
Výsledky

ID	Název položky	Název místa	Název služby	ID objednávky	Vytvořeno	Zaplaceno	Status	Společnost	Cena
204	01 Dlouhá ulice	Litoměřice	Parkování	315	06.12.2019 10:35	06.12.2019 10:36	Done	Globdata a.s.	60,00
205	01 Dlouhá ulice	Litoměřice	Parkování	318	06.12.2019 11:18	06.12.2019 11:19	Done	Globdata a.s.	60,00
206	Kalinovo nábřeží	Havlíčkův Brod	Parkování	319	06.12.2019 14:43	06.12.2019 14:45	Done	Globdata a.s.	13,00
207	Kalinovo nábřeží	Havlíčkův Brod	Parkování	320	06.12.2019 14:46	06.12.2019 14:47	Done	Globdata a.s.	13,00
208	Kalinovo nábřeží	Havlíčkův Brod	Parkování	321	06.12.2019 14:49	06.12.2019 14:50	Done	Globdata a.s.	13,00
209	Kalinovo nábřeží	Havlíčkův Brod	Parkování	322	06.12.2019 14:51	06.12.2019 14:52	Done	Globdata a.s.	13,00
210	Kalinovo nábřeží	Havlíčkův Brod	Parkování	323	06.12.2019 15:12	06.12.2019 15:14	Done	Globdata a.s.	13,00
211	Kalinovo nábřeží	Havlíčkův Brod	Parkování	324	06.12.2019 15:33	06.12.2019 15:34	Done	Globdata a.s.	13,00
212	Kalinovo nábřeží	Havlíčkův Brod	Parkování	325	06.12.2019 15:37	06.12.2019 15:38	Done	Globdata a.s.	13,00
213	Kalinovo nábřeží	Havlíčkův Brod	Parkování	326	06.12.2019 15:50	06.12.2019 15:51	Done	Globdata a.s.	13,00
214	Kalinovo nábřeží	Havlíčkův Brod	Parkování	327	06.12.2019 15:53	06.12.2019 15:54	Done	Globdata a.s.	13,00
215	Kalinovo nábřeží	Havlíčkův Brod	Parkování	328	06.12.2019 15:57	06.12.2019 15:58	Done	Globdata a.s.	13,00
216	Kalinovo nábřeží	Havlíčkův Brod	Parkování	329	06.12.2019 16:54	06.12.2019 16:57	Done	Globdata a.s.	13,00
217	Kalinovo nábřeží	Havlíčkův Brod	Parkování	330	06.12.2019 17:26	06.12.2019 17:27	Done	Globdata a.s.	13,00
218	Kalinovo nábřeží	Havlíčkův Brod	Parkování	331	06.12.2019 17:38		Created	Globdata a.s.	13,00
219	Kalinovo nábřeží	Havlíčkův Brod	Parkování	332	06.12.2019 17:38		Created	Globdata a.s.	13,00
220	Kalinovo nábřeží	Havlíčkův Brod	Parkování	333	06.12.2019 17:40	06.12.2019 17:42	Done	Globdata a.s.	13,00
221	Kalinovo nábřeží	Havlíčkův Brod	Parkování	334	06.12.2019 17:49	06.12.2019 17:49	Done	Globdata a.s.	13,00
222	Kalinovo nábřeží	Havlíčkův Brod	Parkování	335	06.12.2019 19:00	06.12.2019 19:01	Done	Globdata a.s.	13,00
223	Kalinovo nábřeží	Havlíčkův Brod	Parkování	336	06.12.2019 19:01	06.12.2019 19:02	Done	Globdata a.s.	13,00
ID	**Název položky**	**Název místa**	**Název služby**	**ID objednávky**	**Vytvořeno**	**Zaplaceno**	**Status**	**Společnost**	**354,00**

Předchozí 9 10 11 12 13 Další

[Zdroj: autor]

2) Komponenta GridReport – seskupení dat se zobrazením grafu



[Zdroj: autor]

Oskenované zadání práce

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. Tomáš Falta**
Studium:	I1700484
Studijní program:	N1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název diplomové práce:	**Realizace komplexních reportů na platformě Microsoft .NET**
Název diplomové práce AJ:	Realization of Complex Reports on Microsoft .NET platform

Cíl, metody, literatura, předpoklady:

Cílem práce je popsat a porovnat možnosti vývoje komplexních reportů na platformě Microsoft .NET. V teoretické části práce budou představeny použité technologie, zvolené druhy reportů a balíky komponent určené pro tvorbu reportů. V praktické části budou následně stanoveny požadavky pro jednotlivé reporty, které budou následně analyzovány a bude provedeno zhodnocení výsledků. Součástí praktické části práce bude projekt s implementovanými reporty a vlastní implementace stránkovaného a filtrovaného tabulkového reportu. Obsah: Teoretická část 1) Úvod 2) Technologie .NET framework, ASP.NET 3) Reporty 4) Testované balíky komponent pro tvorbu reportů Praktická část 5) Stanovení požadavků pro jednotlivé reporty 6) Analýza a porovnání komponent dle stanovených požadavků 7) Vlastní implementace stránkovaného gridu 8) Závěr

bude upřesněno

Garantující pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	doc. Mgr. Tomáš Kozel, Ph.D.

Datum zadání závěrečné práce: 15.10.2018