UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Aplikace pro oceňování motorových vozidel

David Kapeš

Bakalářská práce
2024

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**David Kapeš**
Osobní číslo:	**I21164**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Aplikace pro oceňování motorových vozidel**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Bakalářská práce se zabývá vytvořením programu, který má schopnost provádět oceňování motorových vozidel v souladu se Znaleckým standardem. Hlavním cílem je vytvoření nástroje pro výpočet časové a obvyklé ceny motorových vozidel napříč různými kategoriemi. Ocenění vozidla bude zahrnovat aspekty jako amortizace během doby provozu, počet ujetých kilometrů a hodnocení technického stavu vozidla. Aplikace bude využívat oceňovací tabulky dle Znaleckého standardu jako hlavní zdroj dat pro výpočty. V teoretické části student popíše problematiku oceňování vozidel a používané standardy, v praktické části navrhne aplikaci a implementuje ji v programovacím jazyku C# ve spojení s frameworkem Entity Framework a .NET knihovnami. Výsledný program by měl zjednodušit proces oceňování motorových vozidel.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

BORY, Pavel. C# bez předchozích znalostí. 2. vydání. V Brně: Computer Press, 2022. ISBN 978-80-251--5061-0.
SMITH, John P. Entity Framework Core in Action. 2nd edition. Manning, 2021. ISBN 1617298360.

Vedoucí bakalářské práce: **Ing. Jan Merta, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlášení autora

Prohlašuji:
Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.
Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 27. 04. 2024 David Kapeš

Poděkování

Rád bych moc poděkoval všem blízkým a mentorům, kteří mi byli oporou a poskytli mi cenné rady během mého studia a psaní bakalářské práce.

Speciální poděkování patří mému vedoucímu práce, Ing. Janu Mertovi, Ph.D., jehož odborné vedení a skvělé rady byly důležité jak pro mě, tak pro tuto bakalářskou práci.

Dále bych chtěl vyjádřit hluboký dík svým rodičům, Pavle Kapešové a Martinu Kapešovi, za jejich odborné znalosti, neustálou motivaci a podporu.

Anotace

Práce je zaměřena na vytvoření softwarové aplikace pro oceňování motorových vozidel v souladu se Znaleckým standardem. Cílem je vytvoření aplikace, která umožní uživateli podle zadaných specifikací vozidla zhotovit jeho časovou a tržní hodnotu. Projekt využívá programovací jazyk C# a framework Entity Framework pro implementaci dynamických výpočtů a uživatelsky přívětivého rozhraní. Projekt kombinuje teoretické poznatky o oceňování s praktickou realizací a umožňuje export výsledků do formátu .docx.

Klíčová slova

Ocenění vozidel, softwarová aplikace, Znalecký standard, C#, Entity Framework, dynamické výpočty, uživatelské rozhraní, tržní hodnota.

Title

Development of a Motor Vehicle Valuation Tool Based on Expert Standards

Annotation

The work is focused on the creation of a software application for the valuation of motor vehicles in accordance with the Valuation Standard. The aim is to create an application that will allow the user to produce a time and market value of the vehicle according to the given specifications. The project uses the C# programming language and the Entity Framework to implement dynamic calculations and a user-friendly interface. The project combines theoretical knowledge of valuation with practical implementation and allows the export of results to .docx format.

Keywords

Vehicle valuation, software application, Expertise standard, C#, Entity Framework, dynamic calculations, user interface, market value.

Obsah

Seznam zkratek

EF	Entity Framework (Rámec pro práci s databázemi v .NET)
WPF	Windows Presentation Foundation (Rámec pro vytváření grafických uživatelských rozhraní v .NET)
C#	Programovací jazyk vyvinutý společností Microsoft
.NET	.NET Framework, platforma pro vývoj aplikací
PDs	Poměrné díly skupiny
ZAs	Základní amortizace
PSs	Přirážka/srážka
ZUs	Zbytková užitnost
PZUs	Poměrná zbytková užitnost
COB	Cena obvyklá
VUs	Výchozí užitnost
ZAD	Základní amortizace za dobu provozu
ZAP	Základní amortizace za počet ujetých kilometrů
UI	Uživatelské rozhraní
PDs	Poměrné díly skupiny (poměrná část ceny vozidla připadající na jeho jednotlivé části)
ZUrv	Zbytková užitná hodnota redukovaného vozidla

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době je oceňování motorových vozidel velmi náročný úkol. To je důvodem, proč byla vyvinuta aplikace, která zjednodušuje tento proces v souladu s aktuálními Znaleckými standardy. Aplikace, vyvinutá v rámci této bakalářské práce, umožňuje uživatelům efektivně a přesně vyhodnotit obvyklou cenu vozidla na základě řady specifikovaných parametrů jako jsou značka, model, datum první registrace a najeté kilometry.

Programy podobné tomuto jsou na dnešním trhu často velmi staré a ve velmi malém množství, proto jsem se rozhodl podobnou aplikaci navrhnout, nicméně věděl jsem, že to nebude snadný úkol, jelikož celková problematika oceňování vozidel je poměrně složitá a rozsáhlá.

První část práce se zabývá teoretickým základem oceňování vozidel, kde je představen používaný standard, podle kterého byla aplikace vyvinuta, a uvedení do samotné problematiky oceňování vozidel. Tato část zkoumá komplexní proces určení finanční hodnoty motorového vozidla, který zahrnuje analýzu technického stavu, věku, najetých kilometrů a dalších faktorů ovlivňujících tržní cenu. Důležitým prvkem je definice a použití Znaleckého standardu I/2022.

Tento standard je klíčový pro stanovení hodnoty motorových vozidel. Tento standard poskytuje metodický rámec, který zahrnuje systematické postupy a metody pro oceňování silničních a zvláštních vozidel, a také určení majetkové újmy způsobené jejich poškozením.

Druhá část se věnuje praktické implementaci aplikace v programovacím jazyku C#, s využitím Entity Framework pro práci s databází a WPF pro grafické rozhraní. Je zde podrobně popsána architektura aplikace a její funkcionalita. Výsledný software je schopen dynamicky generovat odhady ceny, které mohou být upravovány a exportovány pro další analýzu nebo prezentaci.

V této části se také podíváme na klíčové výpočty a strukturu kódu, které jsou základem pro ocenění vozidel v naší aplikaci. Prozkoumáme, jak aplikace rozlišuje různé kategorie vozidel a adaptuje se na jejich specifické parametry pro přesné a efektivní sestavení znaleckého posudku.

Cílem práce je tedy poskytnout nástroj, který profesionalizuje proces oceňování a zároveň zvyšuje jeho přesnost a spolehlivost, což je důležité hlavně pro zainteresované strany v obchodním i soukromém sektoru.

1 Oceňování vozidel

Oceňování vozidel je komplexní proces určení finanční hodnoty motorového vozidla založený na různých faktorech, jako je stáří vozidla, jeho technický stav, počet najetých kilometrů a další charakteristiky ovlivňující tržní cenu. Tento proces je nezbytný pro řadu transakcí a situací, kde je požadována objektivní hodnota vozidla. [1]

Mezi tyto situace patří nejen prodeje a nákupy vozidel, ale také jejich pojišťování, zajištění úvěrů, a dokonce i v rámci soudních sporů, dědických řízení nebo v případech daňového ocenění. Správné oceňování je klíčové pro určení spravedlivé tržní hodnoty, což pomáhá předejít finančním ztrátám nebo nespravedlnostem v právních a obchodních transakcích. [1]

1.1 Znalecký standard

Znalecký standard pro oceňování vozidel je soubor metod a postupů, které byly vytvořeny za účelem oceňování motorových vozidel. Tyto standardy přinášejí jednotnost a přesnost do procesu ocenění tím, že zavádějí systematický přístup k analýze vozidla.

„Novela znaleckého standardu vychází ze zákona o oceňování majetku ve znění účinném od 1.1. 2021 a stanovuje doporučené metodické postupy pro oceňování silničních a zvláštních vozidel a určení výše majetkové újmy způsobené jejich poškozením.“ [2]

Kromě technických inspekcí, které zjišťují fyzický stav vozidla a jeho funkčnost, zahrnuje proces také porovnání s aktuálními tržními daty, která odhalují momentální poptávku a cenové hodnoty srovnatelných vozidel. Dále se využívají profesionálně vypracované tabulky pro amortizaci a odpisy, které reflektují míru stárnutí a opotřebení vozidla v závislosti na jeho věku, typu, modelu a použití. Tyto tabulky pomáhají zajistit, že hodnocení je konzistentní a založené na uznávaných principech, čímž se snižuje prostor pro subjektivní zaujatost oceňovatele. [2]

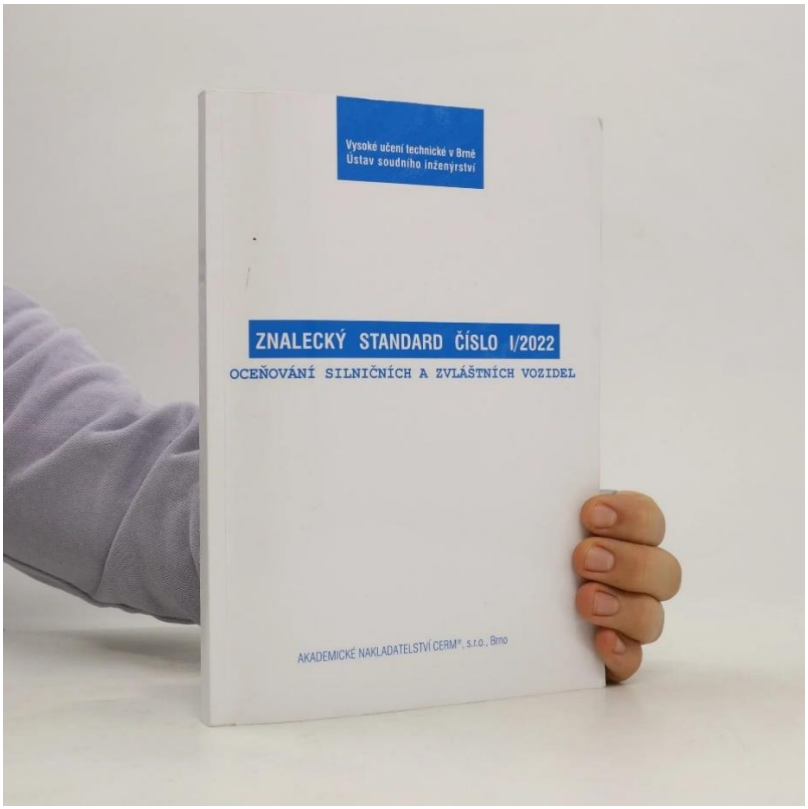
1.1.1 Použitý znalecký standard

Pro oceňování vozidel v tomto projektu byl využit "Znalecký Standard Číslo I/2022" (Tištěná verze). Tento dokument stanovuje doporučené metody a postupy, které jsou uznané odborníky v oblasti oceňování motorových vozidel. [2]

Metodiky použité ve znaleckém standardu nejenže umožňují stanovit tržní cenu vozidla, ale také poskytují možnosti pro výpočet majetkové újmy, nákladů na opravy a jiné finanční kalkulace související s poškozením vozidla. Tato práce se však primárně zaměřuje pouze na část, která se týká stanovení časové a tržní hodnoty vozidla.

Důležitou součástí je také porovnání s aktuálními tržními daty, která reflektují poptávku a situaci na trhu se srovnatelnými vozidly. To pomáhá objektivizovat ocenění vozidla ve srovnání s aktuálním trhem v rozhodném období.

V rámci tohoto standardu jsou také použity profesionálně vypracované tabulky pro amortizaci a odpisy, které odrážejí stáří, typ, model a použití vozidla. Tyto tabulky jsou nezbytné pro kalkulaci finanční hodnoty vozidla, která odpovídá jeho skutečnému opotřebení a očekávané životnosti. Znalecký standard tak poskytuje pevný základ pro spravedlivé a transparentní ocenění vozidel, minimalizuje prostor pro subjektivní zaujatost a zajišťuje, že všechna ocenění jsou konzistentní a založená na uznávaných metodách. Díky poskytnuté metodice jsou výsledné výpočty také dobře přezkoumatelné. [3]



Obrázek 1 - Znalecký standard číslo I/2022, zdroj: [2][4]

1.2 Problémy v oceňování vozidel

Proces oceňování může být komplikovaný kvůli následujícím důvodům:

- Odhad ceny může být ovlivněn osobními názory oceňovatele, pokud nejsou striktně dodržovány standardizované metody.
- Různé typy vozidel (osobní, nákladní, speciální účely atd.) vyžadují různé přístupy k ocenění.
- Tržní hodnoty vozidel se neustále mění, což vyžaduje aktuální znalost trhu.
- Technologie vozidel se v průběhu času neustále vyvíjí [3]

1.3 Význam Znaleckého standardu

Znalecký standard je metodikou, jak objektivně hodnotit a určit cenu vozidla. Standardizace procesu pomáhá eliminovat subjektivní faktory a zajišťuje, že ocenění bude spravedlivé a odpovídající. Použití těchto standardů ve vývoji softwarové aplikace pro oceňování znamená, že výsledky jsou spolehlivější a urychlují proces zpracování ocenění nebo znaleckého posudku. [3]

2 Aplikace znaleckého posudku ve specifických odvětvích

Znalecký posudek motorových vozidel lze uplatnit v mnoha situacích, kde je třeba objektivně stanovit hodnotu vozidla. Tato hodnota má klíčový význam pro řadu odvětví a situací. Níže jsou uvedeny hlavní oblasti, kde se znalecký posudek nejčastěji využívá. [3]

2.1 Soudní spory a právní řízení

Znalecké posudky jsou často nezbytné v případech, kdy je potřeba určit hodnotu vozidla pro účely soudních sporů. Toto může zahrnovat spory týkající se dopravních nehod, rozvodové řízení, kde jsou vozidla brána jako součást majetkového vyrovnání, nebo spory o dědictví. Posudek poskytuje nezaujatý a odborně podložený odhad hodnoty, což je klíčové pro spravedlivé a transparentní rozhodování. [3]

2.2 Pojištění

V pojišťovnictví se znalecké posudky využívají k určení správné hodnoty vozidla pro stanovení výše pojistného, nebo při pojistných událostech, jako je havárie nebo krádež, k vyčíslení škody a následného pojistného plnění. [3]

2.3 Prodej a nákup vozidel

Při koupi nebo prodeji vozidel, zejména u ojetých aut, poskytuje znalecký posudek důležité informace o technickém stavu a tržní hodnotě vozidla. Tím pomáhá oběma stranám dosáhnout spravedlivé dohody a zabraňuje možným sporům způsobeným nesrovnalostmi v ocenění. [3]

2.4 Leasing a financování

Pro finanční instituce poskytující leasing nebo úvěry na nákup vozidel je znalecký posudek důležitým nástrojem pro ověření hodnoty zastaveného majetku. Tím se minimalizuje riziko pro poskytovatele financí a zajišťuje, že financovaná částka odpovídá skutečné hodnotě vozidla. [3]

2.5 Daňové ocenění a účetnictví

Pro účely účetnictví a daní musí firmy správně vykázat hodnotu svých vozidel. Znalecké posudky zde hrají roli v přesném stanovení odpisů a hodnoty aktiv pro bilanční účely. [3]

3 Nedostatky stávajících znaleckých standardů

Znalecké standardy pro oceňování vozidel jsou klíčové pro zajištění objektivního a spravedlivého hodnocení. Avšak, jako každý systém, i tyto standardy mají své nedostatky, které mohou ovlivnit přesnost a relevanci ocenění v určitých situacích. Níže jsou uvedeny některé z hlavních nedostatků současných znaleckých standardů.

3.1 Neschopnost efektivně hodnotit vozidla s elektrickým a hybridním pohonem

Jedním z nejvýznamnějších omezení současných znaleckých standardů je jejich nedostatečná adaptace na rychle se rozvíjející trh s elektromobily (EV). Vzhledem k tomu, že technologie a tržní podmínky elektromobilů se vyvíjejí mnohem rychleji než u tradičních spalovacích motorů, často zastaralé tabulky a metody nedokážou správně odrážet jejich skutečnou tržní hodnotu. Amortizace elektrických vozidel, náklady na údržbu, životnost baterií a jejich výměna jsou aspekty, které vyžadují specifické znalosti a aktualizované metody hodnocení. V rámci znaleckého standardu je absence jakýchkoliv amortizačních stupnic pro vozidla s elektrickým a hybridním pohonem. [3]

3.2 Nedostatek univerzálnosti

Současné standardy nemusí být vždy aplikovatelné na všechny typy vozidel (např. vozidla se specifickými nástavbami, speciální vozidla atd.). Dobrým příkladem může být autojeřáb. [3]

4 Technologie

Vývoj aplikace pro oceňování vozidel zahrnoval výběr technologií, které podporují komplexní datové operace a zároveň umožňují vytvoření intuitivního uživatelského rozhraní. Tento odstavec popisuje rozdělení aplikace na jednotlivé komponenty a roli, kterou každá technologie hraje v architektuře projektu. Zaměřuje se na programovací jazyk C#, framework Entity Framework pro databázové operace, WPF pro grafické rozhraní a SQLite databáze pro uložení dat.

4.1 C#

C# byl zvolen jako hlavní programovací jazyk projektu, což je odůvodněno několika faktory. Jedná se o silně typovaný, objektově orientovaný jazyk, který je podporován bohatým ekosystémem .NET frameworku. C# je zvláště vhodný pro vývoj desktopových aplikací díky své integraci s Microsoft technologiemi, což usnadňuje vývoj a zajišťuje vysokou úroveň bezpečnosti a výkonu. Navíc, jeho syntaktická jasnost a moderní funkce, jako jsou LINQ (Language Integrated Query) a asynchronní programování, umožňují vývojářům psát čistý a dobře udržovatelný kód, což je důležité pro správu a rozšíření aplikace [5]

4.2 Entity Framework

Entity Framework (EF) byl zvolen pro správu databáze kvůli jeho schopnosti zjednodušit databázové operace prostřednictvím ORM (Object-Relational Mapping). Tato technologie umožňuje vývojářům manipulovat s databázovými objekty přímo v C#, což eliminuje potřebu psát rozsáhlé množství SQL kódu a usnadňuje správu dat. Tato vlastnost zefektivňuje vývoj, zjednodušuje testování a podporuje čistší kód.

EF také podporuje automatické migrace, což usnadňuje aktualizace databázových schémat bez přerušení služeb, což je velmi užitečná věc při vývoji dynamicky se vyvíjejících aplikací. [6]

4.2.1 Výhody Entity Framework

Tento framework přináší několik výhod.

4.2.1.1 Abstrakce SQL

EF poskytuje vrstvu abstrakce, která skrývá složitost SQL příkazů. Vývojáři mohou manipulovat s daty pomocí standardních operací objektově orientovaného programování, což zvyšuje čitelnost a udržitelnost kódu. [6]

4.2.1.2 LINQ podpora

EF podporuje LINQ (Language Integrated Query), což je výkonný dotazovací jazyk integrovaný přímo do C#. LINQ umožňuje vývojářům psát komplexní dotazy na databázi deklarativním způsobem, který je dobře integrovaný s ostatními funkcemi C#. [6]

4.2.1.3 Zachování konzistence

EF udržuje konzistenci mezi datovým modelem v kódu a skutečnou databází, což minimalizuje riziko chyb při správě dat. Díky svému ORM přístupu, EF zajišťuje, že všechny operace prováděné na databázi jsou bezpečné a efektivní. [6]

4.3 Windows Presentation Foundation (WPF)

Pro grafické uživatelské rozhraní byl využit Windows Presentation Foundation (WPF), což je framework umožňující vytváření vizuálně atraktivních a funkčně bohatých uživatelských rozhraní pro Windows aplikace.

WPF nabízí pokročilé možnosti pro práci s grafikou, animacemi a stylizací, což umožňuje vytvářet uživatelské rozhraní, které je nejen vizuálně přitažlivé, ale také intuitivní pro koncové uživatele. Jeho použití XAML (eXtensible Application Markup Language) pro deklarativní definici UI umožňuje oddělení designu a logiky aplikace. [7]

4.4 SQLite databáze

Pro ukládání dat v aplikaci pro oceňování vozidel byla zvolena SQLite databáze, která je ideální pro tento projekt z několika důvodů. SQLite je vestavěná databáze, což znamená, že nevyžaduje samostatný server pro správu databází a je přímo integrována do aplikace. Tato vlastnost významně zjednodušuje nasazení a distribuci aplikace, jelikož celá databáze je uložena v jednom souboru na disku uživatele.

SQLite je také vysoce spolehlivá a efektivně podporuje potřeby aplikace v oblasti transakcí, dotazů a konzistence dat. Je schopna zvládnout střední objemy dat (což je přesně případ této aplikace) bez nutnosti složitých konfigurací nebo pravidelné údržby. Díky těmto vlastnostem umožňuje SQLite rychlý přístup k datům a jednoduchost, což je důležité pro plynulý chod aplikace. [8]

4.5 Visual Studio

Visual Studio je integrované vývojové prostředí (IDE) od Microsoftu, které podporuje širokou škálu programovacích jazyků a nástrojů pro vývoj aplikací napříč různými platformami. Umožňuje vývojářům psát, ladit a testovat kód efektivně díky pokročilým funkcím, jako jsou inteligentní editace kódu, debugger, a integrovaná podpora pro správu verzí. Visual Studio se také vyznačuje svou adaptabilitou, umožňuje snadnou integraci s dalšími službami a nástroji. Podporuje také moderní vývojové metody, jako je cloudový vývoj a mobilní aplikace. Toto prostředí nabízí vývojářům komplexní řešení pro rychlý a efektivní vývoj software. [9]

5 Architektura aplikace

Architektura aplikace pro oceňování vozidel je postavena na modelu **MVVM** (Model-View-ViewModel), který poskytuje vysokou míru oddělení mezi logikou prezentace a logikou aplikace. [10]

Tento model umožňuje lepší správu a testovatelnost kódu. V následujících odstavcích je popsána struktura jednotlivých vrstev a jejich interakce.

5.1 MVVM

5.1.1 Model

Model reprezentuje doménovou logiku aplikace. Obsahuje data a logiku dat, které nejsou závislé na uživatelském rozhraní. V projektu je tento koncept rozšířen o "repositáře" – třídy, které zprostředkovávají komunikaci mezi databází a logikou aplikace (repozitáře nejsou běžně součástí MVVM architektury). Každý typ vozidla má svůj specifický repositář, který obsahuje metody pro získávání, aktualizaci a manipulaci s daty specifickými pro danou kategorii vozidel. Tyto repositáře jsou implementovány s použitím Entity Framework a SQLite databáze, což umožňuje efektivní a rychlé dotazy na databázi bez nutnosti psát složité SQL dotazy. [10]

5.1.2 View

View vrstva je zodpovědná za prezentaci dat a interakci s uživatelem. Aplikace je organizována do jednoho hlavního okna, které dynamicky zobrazuje různé UserControl okna v závislosti na uživatelských interakcích. Tento modulární přístup umožňuje snadnou správu různých pohledů a funkcí aplikace bez nutnosti přenášet uživatele mezi různými okny nebo aplikacemi.

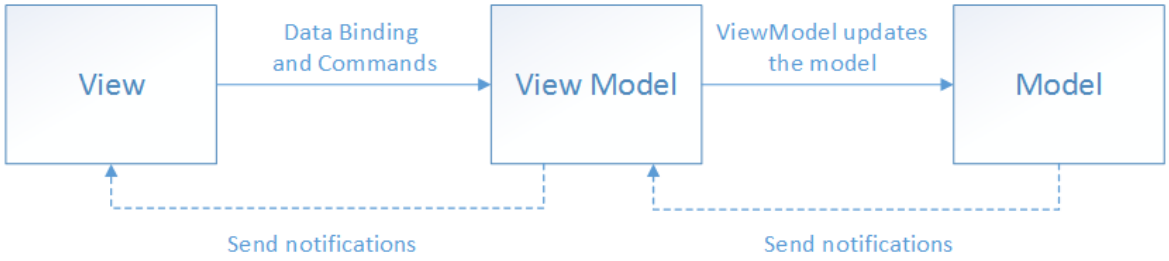
View vrstva je také odpovědná za definici struktury, rozložení a vzhledu toho, co uživatel vidí na obrazovce. Ideálně je každý pohled definován v XAML, s omezeným množstvím kódu v pozadí, který neobsahuje business logiku. Nicméně, v některých případech může obsahovat UI logiku v kódu v pozadí, která implementuje vizuální chování, které je obtížné vyjádřit v XAML, jako jsou animace. [10]

5.1.2.1 UserControl okna

UserControl okna jsou samostatné komponenty v rámci WPF, které slouží k modularizaci uživatelského rozhraní. Každé UserControl okno může obsahovat vlastní logiku a layout, což usnadňuje znovupoužití kódu a zjednodušuje údržbu aplikace. V tomto projektu každý UserControl reprezentuje specifický pohled nebo funkční sekci aplikace, jako jsou například formuláře pro vstup dat specifických podle uživatelem zadaných údajů, nebo zobrazení výsledků. Tyto UserControl okna jsou dynamicky načítána do hlavního okna aplikace podle toho, jaké úkoly nebo akce uživatel provádí, což zajišťuje, že uživatelské rozhraní zůstává čisté, organizované a přehledné.

5.1.3 ViewModel

ViewModel funguje jako spojnice mezi View a Model vrstvami. Ve ViewModelu jsou definovány vlastnosti a příkazy (commands), které reagují na uživatelské akce, zpracovávají data a aktualizují View. ViewModel také zpracovává veškerou logiku, která se týká prezentace dat, ale nemanipuluje přímo s UI prvky, což umožňuje oddělení prezentace od business logiky. [10]



Obrázek 2 - MVVM architektura, zdroj: [10]

5.2 Analýza požadavků

Aplikace pro oceňování motorových vozidel slouží jako komplexní nástroj, který umožňuje uživatelům zjistit tržní hodnotu vozidel na základě řady specifikovaných parametrů, jako je stáří vozidla, počet najetých kilometrů a kategorie vozidla.

Tato platforma v mnoha ohledech připomíná systémy používané pro hodnocení a oceňování v jiných oborech, například online katalogy ojetých vozidel, které nabízí podobné funkcionality s přidáním specifických vlastností pro jednodušší a přesnější vyhodnocení vozidel. Základní funkcionalita aplikace zahrnuje sofistikované výpočty amortizace, podílových skupin a zbytkové užitnosti vozidla, které společně tvoří komplexní hodnotící mechanismus nezbytný pro stanovení správné tržní ceny.

Aplikace přináší řadu užitečných funkcí, které usnadňují a zefektivňují proces ocenění, a jsou navrženy tak, aby poskytovaly výsledky v maximálně přehledné a srozumitelné formě.

5.2.1 Specifika uživatelského přístupu

Aplikace funguje na principu jednotného uživatelského rozhraní bez specifických úrovní oprávnění. To znamená, že neexistují žádní administrátoři nebo rozdílné úrovně uživatelů. Každý, kdo aplikaci používá, má přístup ke všem jejím funkcím a datům, podobně jako u běžné kalkulačky. Tento přístup zajišťuje, že všechny funkce jsou přístupné bez nutnosti zvláštního nastavování nebo konfigurace, což usnadňuje a zrychluje práci s aplikací.

5.2.2 Funkční požadavky

- Přijímat vstupní parametry vozidla – uživatelé budou moci zadávat klíčové informace o vozidle, jako jsou stáří vozidla, počet najetých kilometrů a kategorie vozidla. Tyto údaje jsou základem pro všechny následné výpočty a ocenění vozidla.

- Výpočet amortizace – na základě vstupních údajů aplikace automaticky vypočítá základní amortizaci, která zahrnuje srážky za dobu provozu (ZAD) a srážky za počet najetých kilometrů (ZAP).

- Výpočet podílových skupin – aplikace dále vypočítá hodnoty pro podílové skupiny, které určují finanční hodnotu jednotlivých částí vozidla ve vztahu k jeho celkové tržní hodnotě.

- Výpočet zbytkové užitnosti redukovaného vozidla (ZUrv) – na základě předchozích výpočtů a vstupů uživatele aplikace určí zbytkovou užitnost redukovaného vozidla, což představuje konečnou tržní hodnotu vozidla v současném stavu.

- Generování a export dat – aplikace umožní generování podrobné tabulky s výpočty, kterou může uživatel exportovat do formátu .docx pro další použití nebo archivaci.

5.2.3 Nefunkční požadavky

- Jednoduchost – aplikace je intuitivní a snadno použitelná i pro osoby bez technického vzdělání.

- Výkon – reakce aplikace na uživatelské vstupy a požadavky je rychlá, aby zajišťovala plynulý pracovní postup.

- Přehlednost – uživatelské rozhraní poskytuje jasné informace a vede uživatele krok za krokem skrze proces oceňování.

- Přenositelnost – aplikace je kompatibilní s různými operačními systémy a není vázána na konkrétní hardwarové prostředí.

- Rychlost – výpočty a databázové operace jsou optimalizovány pro rychlost a efektivitu.

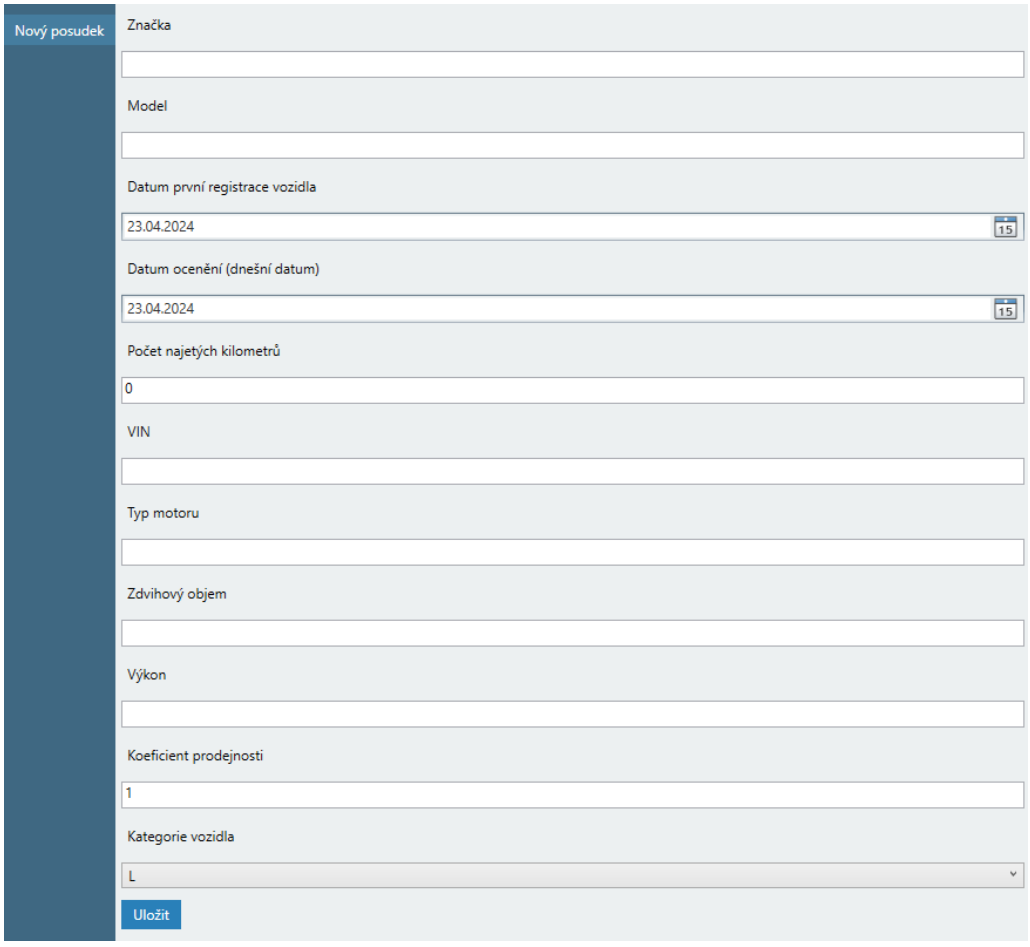
- Údržba – kód aplikace je dobře strukturovaný, což usnadňuje budoucí rozšíření a údržbu.

6 Aplikace

Desktopová aplikace slouží k sestavování znaleckých posudků vozidel. Po spuštění poskytuje přehledné uživatelské rozhraní, ve kterém je možné bezprostředně zahájit tvorbu nového posudku. Aplikace vede uživatele krokově přes proces sběru důležitých údajů o vozidle. Začíná se zadáním základních informací jako jsou značka, model, datum první registrace, a pokračuje dalšími specifickými parametry, které jsou nezbytné pro kvalifikované ocenění vozidla. Okamžitá interakce a intuitivní navigace umožňuje efektivní a přesnou práci při vytváření posudku.

6.1 Vzhled aplikace

Aplikace představuje jednoduchý design, jenž zaručuje snadnou orientaci v nabízených funkcích. V hlavním okně je dispozici strukturovaný formulář pro zadávání informací, který je logicky rozdělen podle kategorií vozidel a specifických atributů nutných k ocenění. Každý segment formuláře je jasně označen a připraven k přijetí příslušných údajů, což zjednodušuje proces zadávání dat a minimalizuje riziko chyb při vyplňování. Estetika aplikace je koncipována tak, aby neodváděla pozornost od hlavní funkce a současně poskytovala vizuálně příjemné pracovní prostředí.



Obrázek 3 - Úvodní okno aplikace, zdroj: vlastní

6.2 Nový posudek

V procesu vytváření nového znaleckého posudku aplikace umožňuje uživateli zadat řadu informací, z nichž některé jsou určeny jako povinné, aby bylo možné provést přesný výpočet hodnoty vozidla. Kritickými informacemi jsou „datum první registrace vozidla“, „datum ocenění“, „počet najetých kilometrů“, „koeficient prodejnosti“ a „kategorie vozidla“. Těmto datům je přikládán zvláštní význam, neboť hrají klíčovou roli v algoritmu pro výpočet amortizace vozidla.

Výběr kategorie vozidla je zásadním krokem, protože aplikace dále nastavuje formulář podle specifikované kategorie.

K dispozici jsou následující kategorie:

- L – Motorky
- M1 a N1 – Osobní automobily
- M2 a M3 – Autobusy
- N2 a N3 – Nákladní automobily
- T a C – Traktory
- O a R – Přípojná vozidla
- S – Pracovní stroje

Každá kategorie vozidla má předepsané další parametry nutné pro hodnocení, což znamená, že uživatel je při výběru přesměrován do odpovídající sekce aplikace. V této části pak uživatel doplní specifické informace, které jsou vyžadovány pro danou kategorii vozidla.

Po uložení aktuálního formuláře a ověření, že všechna potřebná data byla vyplněna správně a ve správném formátu, jsou informace o posudku uloženy do lokálního úložiště aplikace, známého jako *LocalStorage*. Objekt *ExpertReview*, který obsahuje kompletní sadu uživatelem poskytnutých údajů, je pak přesunut do této centrální úložné instance.

V *LocalStorage* jsou data uchovávána a jsou připravena k dalším operacím. Tento proces zajišťuje, že všechny informace jsou bezpečně uloženy a mohou být využity v dalších krocích aplikace.

Objekt *ExpertReview* představuje centrální datovou strukturu v aplikaci pro sestavování znaleckých posudků vozidel. Obsahuje základní charakteristiky vozidla, které jsou esenciální pro ocenění.

6.2.1 Zadávané hodnoty

Ve vstupním formuláři je potřeba vyplnit několik vstupních údajů. Některé jsou povinné, jelikož se z nich budou později počítat srážky na ceně vozidla.

Údaje jsou následující:

- Značka vozidla
- Model vozidla
- Datum první registrace, což je klíčový údaj pro určení stáří vozidla (povinné)
- Datum, od kterého se vozidlo začíná oceňovat (povinné)
- Počet najetých kilometrů, důležitý pro výpočet opotřebení a hodnoty vozidla (povinné)
- VIN – Výrobní číslo vozidla
- Typ motoru
- Zdvihový objem motoru.
- Výkon motoru.
- Typ paliva
- Koeficient prodejnosti (povinné)
- Kategorie vozidla (povinné)

Kromě následujících údajů, které všechny reprezentují údaje z formuláře, objekt obsahuje ještě 4 zásadní hodnoty ZAD, ZAP, ZUrv a COB. Jedná se o primární výpočty, ze kterých lze určit výslednou cenu vozidla. Výpočet těchto hodnot je jedna z hlavních problematik celého programu. Jedná se o následující hodnoty:

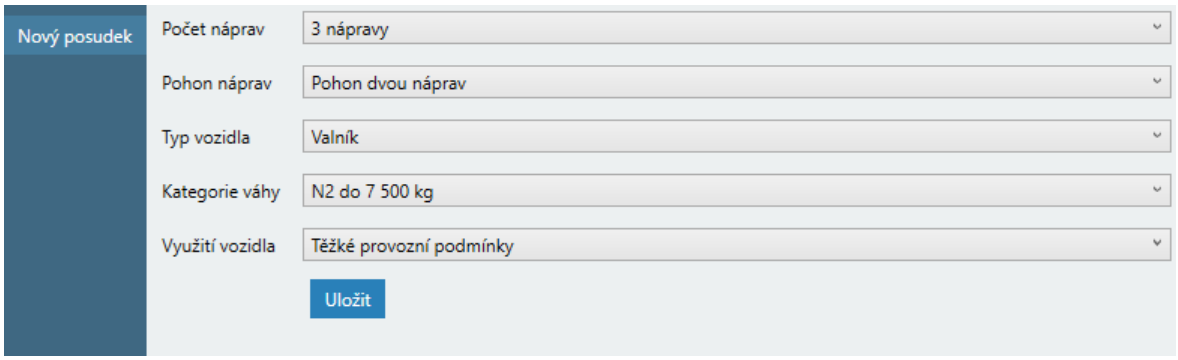
- ZAD: Základní srážky za dobu provozu
- ZAP: Základní srážky za počet ujetých kilometrů
- ZUrv: Zbytková užitnost redukovaného vozidla
- COB: Cena obvyklá, což reprezentuje tržní hodnotu vozidla.

6.3 Separátní typy vozidel

Aplikace rozlišuje vozidla do různých kategorií, přičemž každá kategorie má své specifické parametry, které je třeba vyplnit pro správné sestavení znaleckého posudku. Každé vozidlo má nějakou svoji specifickou kategorii, pod kterou je označováno. Například běžné automobily mají kategorii „M1 a N1“. Tyto kategorie jsou určeny regulacemi a mají přímý vliv na další hodnocení a klasifikaci vozidla:

- L – Pro kategorii motocyklů je v aplikaci potřeba specifikovat pouze konkrétní kategorii motorky.
- M1 N1 – U osobních automobilů jsou uživatelské formuláře rozšířeny o dodatečné informace jako Podkategorie vozidla, Typ vozidla, Podtyp vozidla a Obchodní třída, které pomáhají lépe charakterizovat vozidlo pro jeho ocenění.
- M2 M3 – U autobusů je definujícím parametrem třída autobusu, což reflektuje různé typy a velikosti autobusů a jejich potenciální využití.
- N2 N3 – U nákladních vozidel aplikace vyžaduje detailnější informace jako počet náprav, pohon náprav, typ vozidla, kategorie váhy a využití vozidla, aby mohla přesně stanovit hodnotu vozidla.
- T C – U traktorů je důležité specifikovat typ traktoru a počet náprav, což má značný vliv na možnosti využití a hodnotu traktoru.
- R – Přípojná vozidla jsou v aplikaci klasifikována především dle počtu náprav, který je důležitý pro určení kategorie vozidla.
- S – Pro výměnná tažná zařízení neboli pracovní stroje je klíčová pouze kategorie vozidla, což odráží jejich specifické vlastnosti a možnosti využití.

Formuláře aplikace jsou dynamické a pružně se přizpůsobují zvolené kategorii vozidla. Například v kategorii N2 N3, pokud uživatel vybere počet náprav, formulář se dynamicky aktualizuje a nabídne relevantní možnosti pro pohon náprav, které jsou vhodné pro zvolený počet náprav. Tento inteligentní design formulářů zajistí, že uživatelé budou vždy prezentováni s relevantními možnostmi a že data získaná pro znalecký posudek jsou co nejpřesnější a nejúplnější.



Obrázek 4 - Ukázka okna formuláře pro N2 a N3 vozidla, zdroj: vlastní

7 Základní amortizace (ZA)

Základní amortizace (ZA) je klíčový koncept v ocenění vozidel, který kvantifikuje průměrné snížení užitných vlastností vozidla v závislosti na době provozu a počtu ujetých kilometrů. Amortizace je vyjádřena procentuálně a vychází z normovaných hodnot. [2]

Výpočet ZA využívá dvou základních amortizačních hodnot:

- **Základní srážka za dobu provozu (ZAD)**

Hodnota ZAD je určena podle tabulky odpisových sazeb, které odpovídají jednotlivým rokům provozu vozidla. Pro kategorii osobních automobilů M1 a N1 může výpočet vypadat takto:

Doba provozu vozidla roky	Základní srážka za dobu provozu pro vozidla obchodní třídu A	Základní srážka za dobu provozu vozidla pro ostatní obchodní třídy
1. rok	25 %	33 %
2. rok	33 %	40 %
3. rok	40 %	45 %
4. rok	45 %	50 %
5. rok	50 %	55 %
6. rok	55 %	60 %
7. rok	60 %	65 %
8. rok	65 %	70 %
9. rok	70 %	74 %
10. rok	74 %	78 %
11. rok	78 %	82 %
12. rok	82 %	86 %
13. rok	86 %	90 %
14. rok a další	90 %	90 %

Tabulka 1 - Vozidla M1 a N1, základní srážky za dobu provozu, zdroj: [2]

Příklad: Pokud je automobil skupiny "Ostatní obchodní třídy" ve věku 4,5 roku, dle tabulky (Tabulka 1) odpovídá hodnota ZAD pro čtvrtý rok 50 %. Pro zohlednění půl roku, který zbývá do pátého roku provozu, je potřeba dopočítat hodnotu z příslušného rozdílu mezi sazbami pro čtvrtý a pátý rok. Ve výsledku lze tedy vypočítat, že ZAD by bylo 52,5 % pro daný vůz.

Vzorec pro výpočet rozdílu na základě doplňujících měsíců lze vyjádřit takto:

$$ZAD = Srážka\ pro\ daný\ rok + \left(\frac{Srážka\ pro\ následující\ rok - Srážka\ pro\ daný\ rok}{12}\right)$$

Příslušné tabulky pro výpočet ZAD se liší pro každou kategorii vozidla, ale samotný princip výpočtu srážky je pro všechny stejný.

- **Základní srážka za počet ujetých kilometrů (ZAP)**

Tato hodnota reflektuje celkový počet kilometrů, které vozidlo najelo, a odráží opotřebení související s tímto ujetým kilometrovým výkonem. Pro výpočet základní srážky za počet ujetých kilometrů (ZAP) je důležité zohlednit třídu vozidla a celkový počet najetých kilometrů. Srážka se aplikuje na každých 1000 km ujeté vzdálenosti, přičemž pro různé rozsahy ujetých kilometrů mohou být stanoveny odlišné procentuální sazby.

Obchodní třída	Rozsah ujetých km	% srážka za každých 1 000 km
A	do prvních 15 000	1,30 %
A	nad 15 000	0,65 %
B/B-BMPV/B-SUV/S	do prvních 20 000	0,90 %
B/B-BMPV/B-SUV/S	nad 20 000	0,40 %
C/C-MPV/C-SUV	do prvních 30 000	0,70 %
C/C-MPV/C-SUV	nad 30 000	0,30 %
D/D-MPV/D-SUV	do prvních 30 000	0,70 %
D/D-MPV/D-SUV	nad 30 000	0,30 %
E/E-SUV	do prvních 40 000	0,70 %
E/E-SUV	nad 40 000	0,30 %
E-MPV/F	do prvních 40 000	0,50 %
E-MPV/F	nad 40 000	0,25 %

Tabulka 2 - Vozidla M1 a N1, základní srážky za počet ujetých kilometrů, zdroj: [2]

Příklad: Máme-li vozidlo třídy B s počtem najetých kilometrů 100 000, pak podle tabulky pro třídu B (Tabulka 2) vypočítáme srážku následovně:

Pro prvních 20 000 km použijeme sazbu 0,9 %, což odpovídá výpočtu 20 × 0,9 % = 18 %. Pro zbytek, tedy pro 80 000 km, použijeme sazbu 0,4 %, což je výpočet 80 × 0,4 % = 32 %. Celková srážka za počet ujetých kilometrů (ZAP) pro vozidlo třídy B s 100 000 km tedy bude 18 % + 32 % = 50 %. Vzorec pro výpočet základní srážky za počet ujetých kilometrů lze vyjádřit takto:

$$ZAP = (km\ v\ prvním\ rozsahu * \ sazba\ pro\ první\ rozsah) + (km\ v\ dalším\ rozsahu * sazba\ pro\ další\ rozsah)$$

Je důležité zmínit, že pro vozidla kategorie "S", "O R" a "T C" se výpočty ZAP neprovádějí.

7.1 Základní amortizace (ZA)

Základní amortizace (ZA) je výsledným ukazatelem, který se používá pro ocenění vozidla. Je to průměr dvou dříve vypočítaných hodnot – základní srážka za dobu provozu (ZAD) a základní srážka za počet ujetých kilometrů (ZAP). Výpočet ZA odhaluje celkový stupeň opotřebení vozidla a je významným faktorem při určení jeho tržní hodnoty. Pro stanovení ZA se používá následující vzorec:

$$ZA = \frac{(ZAD + ZAP)}{2}\ [\%]$$

7.2 Poměrné díly skupiny (PDs)

Výpočet poměrných dílů skupiny (PDs) se opírá o informace specifikované uživatelem v druhé části aplikace, která se věnuje konkrétní kategorii vozidla. PDs představuje část hodnoty vozidla ve stavu nového vozidla, která je přisuzována jednotlivým skupinám součástí a komponentů bez započtení pneumatik a mimořádné výbavy. To znamená, že celková hodnota vozidla je distribuována mezi různé komponenty vozidla, což reflektuje jejich podíl na tržní ceně nového vozidla.

Typ motoru	Koncepce vozidla	Motor	Převodovka	Zadní náprava	Přední náprava vč. řízení	Rám	Karoserie	Výbava karoserie	Hnací hřídel + zadní náprava	Převodovka + hnací hřídel + zadní náprava	Převodovka + hnací hřídel
Motor zážehový	Motor vpředu, zadní pohon	20.0	7.0	0.0	10.0	2.0	23.0	30.0	8.0	0.0	0.0
Motor zážehový	Motor vzadu, zadní pohon	20.0	0.0	0.0	10.0	2.0	23.0	30.0	0.0	15.0	0.0
Motor zážehový	Motor vpředu, přední pohon	20.0	0.0	4.0	11.0	2.0	23.0	30.0	0.0	0.0	10.0
Motor vznětový	Motor vpředu, zadní pohon	25.0	7.0	0.0	10.0	2.0	21.0	27.0	8.0	0.0	0.0
Motor vznětový	Motor vzadu, zadní pohon	25.0	0.0	0.0	10.0	2.0	21.0	27.0	0.0	15.0	0.0
Motor vznětový	Motor vpředu, přední pohon	25.0	0.0	4.0	11.0	2.0	21.0	27.0	0.0	0.0	10.0

Tabulka 3 - Vozidla M1 a N1 s pohonem jedné nápravy, zdroj [2]

Pro praktickou demonstraci: v případě, že uživatel v aplikaci zvolí osobní automobil kategorie M1 N1 s motorovým uspořádáním 'Motor vzadu, zadní pohon', motorem zážehovým a obchodní třídou B, pak podle přiložené tabulky (Tabulka 3) lze určit, že distribuce hodnoty vozidla na jednotlivé skupiny je následující:

- Motor: 20%
- Převodovka, hnací hřídel a zadní náprava: 15%
- Přední náprava včetně řízení: 10%
- Rám: 2%
- Karoserie: 23%
- Výbava karoserie: 30%

Všechny tyto procentuální hodnoty dohromady musí vždy dosáhnout 100 %, což zajistí, že celková hodnota vozidla je rozdělena mezi všechny jeho části. Výpočet PDs je tedy základním kamenem pro správné ocenění vozidla a jeho jednotlivých komponent.

Tímto způsobem se hodnoty získávají v kódu:

```
        public async Task<TableModel>
        M1N1_GetTableData_DriveOfOneAxle(M1N1_DriveOfOneAxle_Type type,
        M1N1_DriveOfOneAxle_Subtype subtype)
        {
            var data = await _context.Set<M1N1_DriveOfOneAxle_Db>()
                .Where(m => m.Type == type && m.Subtype == subtype)
                .Select(m => new TableModel
                {
                    Engine = m.Engine,
                    Transmission = m.Transmission,
                    DriveShaft = m.DriveShaft,
                    RearAxle = m.RearAxle,
                    FrontAxle = m.FrontAxle,
                    Frame = m.Frame,
                    Body = m.Body,
                    BodyEquipment = m.BodyEquipment,
                    DriveShaft_RearAxle = m.DriveShaft_RearAxle,
                    Transmission_DriveShaft_RearAxle =
m.Transmission_DriveShaft_RearAxle,
                    Transmission_DriveShaft = m.Transmission_DriveShaft
                })
                .FirstOrDefaultAsync();
            return data;
        }
```

Tyto tabulky se liší v závislosti na kategorii vozidla, a dokonce uvnitř stejných kategorií se mohou specifikace pro různé třídy vozidel podstatně rozcházet. Z toho důvodu je v aplikaci implementována třída *TableModel*, která je navržena tak, aby flexibilně zachytila různorodé specifické atributy. Tato univerzální třída umožňuje shromažďovat data potřebná pro výpočty v konečné fázi sestavování znaleckého posudku, což zahrnuje kompilaci tabulky s úplným přehledem o všech relevantních výpočtech.

V aplikaci se pro práci s datovými modely a jejich zobrazení využívají slovníky, které mapují technické názvy na čitelné termíny. Třída DictionaryStorage obsahuje slovník, jenž překládá názvy vlastností z TableModel.

7.3 Zpracování

V aplikaci se po dokončení vyplňování údajů, zahrnujících jak základní, tak specifické parametry vozidla, aplikace přistupuje k výpočtům amortizace a poměrných dílů skupin (PDs). Pro tyto účely jsou vytvořeny speciální repositáře pro jednotlivé typy vozidel, které obsahují metody pro výpočet. Tyto metody interagují s databází SQLite, a to prostřednictvím databázového kontextu nazvaného "ExpertReviewContext", který je součástí Entity Frameworku.

	Id	BusinessClass	InitialMileageLimit	InitialDepreciationRate	SubsequentDepreciationRate
	Filtr	Filtr	Filtr	Filtr	Filtr
1	1	1	15000	1.3	0.65
2	2	2	20000	0.9	0.4
3	3	3	20000	0.9	0.4
4	4	4	20000	0.9	0.4
5	5	15	20000	0.9	0.4
6	6	5	30000	0.7	0.3
7	7	6	30000	0.7	0.3
8	8	7	30000	0.7	0.3
9	9	8	30000	0.7	0.3
10	10	9	30000	0.7	0.3
11	11	10	30000	0.7	0.3
12	12	11	30000	0.7	0.3
13	13	13	30000	0.7	0.3
14	14	12	40000	0.5	0.25
15	15	14	40000	0.5	0.25

Obrázek 5 - M1 N1 tabulka pro výpočet ZAD, zdroj vlastní

V aplikaci jsou data pro výpočty založena na tabulkách, které korespondují s tabulkami ve znaleckém standardu I/2022. Tyto tabulky, jak je vidět na přiloženém obrázku, obsahují různé hodnoty pro jednotlivé oceňovací kategorie. To znamená, že jak pro výpočet ZAD, ZAP tak pro PDs máme v tabulkách reprezentovány typy vozidel pomocí výčetních hodnot (enum), v tomto případě, jak je vidět na obrázku to je ‚BusinessClass'. Tyto enum hodnoty jsou v tabulce reprezentovány jako číselné hodnoty. Tabulky pak obsahují také ty samotná hlavní data, která jsou uložena v decimal formátu.

Příklad výčtové hodnoty pro tabulky PDs:

```
public enum M1N1_DriveOfOneAxle_Subtype
{
    [Description("Motor vpředu, zadní pohon")]
    FrontEngineRearDrive = 1,

    [Description("Motor vzadu, zadní pohon")]
    RearEngineRearDrive = 2,

    [Description("Motor vpředu, přední pohon")]
    FrontEngineFrontDrive = 3
}
```

Příklad výčtové hodnoty pro tabulky ZAD:

```
public class M1N1_MileageDepreciation
{
    [Key]
    public int Id { get; set; }
    public M1N1_BusinessClass BusinessClass { get; set; }
    public int InitialMileageLimit { get; set; }
    public decimal InitialDepreciationRate { get; set; }
    public decimal SubsequentDepreciationRate { get; set; }
}
```

Ukázka metody pro výpočet hodnoty ZAD:

```
public async Task<decimal?> GetDepreciationRateAsync(M1N1_BusinessClasses
businessClass, DateTime dateTimeCreated, DateTime awardDate)
        {
            int operationYears = awardDate.Year - dateTimeCreated.Year;
            int remainingMonths = awardDate.Month - dateTimeCreated.Month +
             (DateTime.Now.Day < dateTimeCreated.Day ? -1 : 0);

            if (operationYears == 0 && remainingMonths == 0)
            {
                return 0m;
            }

            if (remainingMonths < 0)
            {
                operationYears -= 1;
                remainingMonths += 12;
            }

            if (operationYears >= 14) { return 90m; }

            var currentYearRate = await _context.Set<M1N1_DepreciationRate>()
                .FirstOrDefaultAsync(r => r.OperationYear == operationYears);

            decimal rate = 0m;
            if (currentYearRate != null)
            {
                switch (businessClass)
                {
                    case M1N1_BusinessClasses.RateA:
                        rate = currentYearRate.RateA;
                        break;
                    case M1N1_BusinessClasses.RateOther:
                        rate = currentYearRate.RateOther;
                        break;
                    default:
                        throw new
ArgumentOutOfRangeException(nameof(businessClass), "Neplatná obchodní třída");
                }
            }

            if (remainingMonths > 0)
            {
                var nextYearRate = await _context.Set<M1N1_DepreciationRate>()
                    .FirstOrDefaultAsync(r => r.OperationYear ==
operationYears + 1);

                if (nextYearRate != null)
                {
                    decimal nextRate = businessClass ==
M1N1_BusinessClasses.RateA ? nextYearRate.RateA : nextYearRate.RateOther;
                    decimal monthlyRateDifference = (nextRate - rate) / 12;
                    rate += monthlyRateDifference * remainingMonths;
                }
            }

            return rate;
        }
```

Ukázka metody pro výpočet hodnoty ZAD:

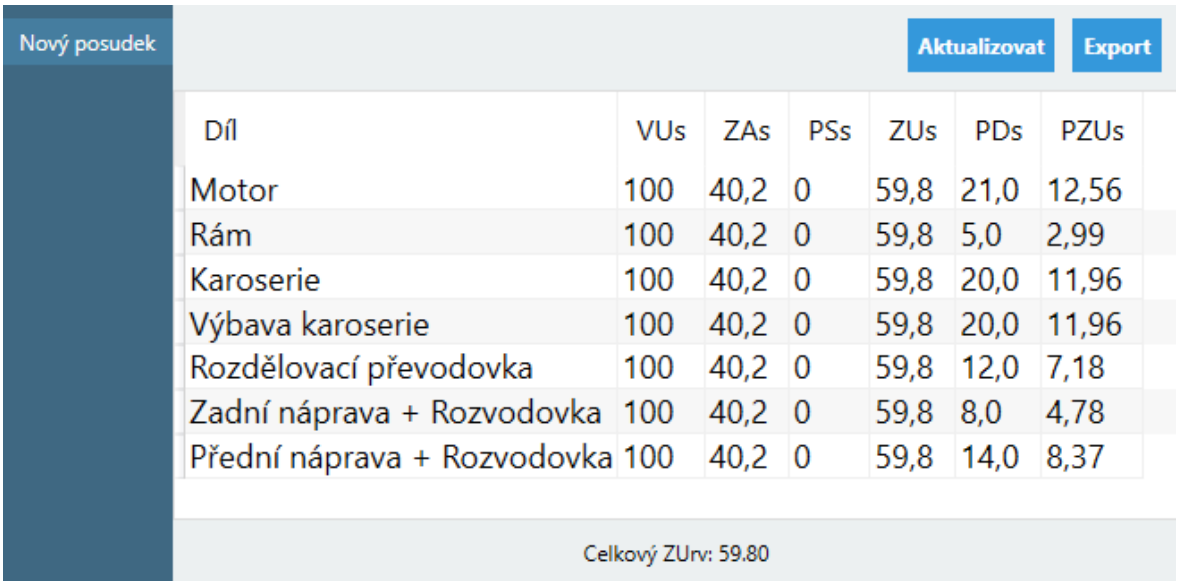
```
public async Task<decimal?>
CalculateTotalDepreciationRateAsync(M1N1_BusinessClass businessClass, int
totalKilometers)
{

// Zaokrouhlení nahoru na celé tisíce
totalKilometers = (int)Math.Ceiling(totalKilometers / 1000m) * 1000;

// Najde odpovídající sazbu odpisu pro zadanou třídu vozidla
var depreciationRateInfo = await
_context.Set<M1N1_MileageDepreciation>().AsNoTracking()
.FirstOrDefaultAsync(m => m.BusinessClass == businessClass);

if (depreciationRateInfo == null)
{
// Informace o sazbě odpisu pro danou třídu vozidla nebyla nalezena
      return null;
}

// Výpočet celkové sazby odpisu
int initialKilometers = Math.Min(depreciationRateInfo.InitialMileageLimit,
totalKilometers);
int subsequentKilometers = totalKilometers - initialKilometers;

decimal totalDepreciationRate = initialKilometers / 1000m *
depreciationRateInfo.InitialDepreciationRate +
subsequentKilometers / 1000m *
depreciationRateInfo.SubsequentDepreciationRate;

if(totalDepreciationRate >= 100)
{
      return 100;
}

      return totalDepreciationRate;
}
```

Databáze je vytvořena lokálně na počítači uživatele při inicializaci aplikace, kde cesta k databázi je dynamicky sestavena a nastavena. Při inicializaci se všechny potřebné tabulky vytvoří a data jsou do nich uložena. Díky využití SQLite nejsou třeba žádné externí databázové servery, což činí systém lehkým a snadno přenositelným.

Výběr dat z tabulek pro konkrétní výpočty se řídí enum hodnotami, které jsou v databázi reprezentovány odpovídajícími číselnými hodnotami. Takový přístup umožňuje snadnou aktualizaci sazeb nebo parametrů v případě změn ve standardu nebo legislativě bez nutnosti přepisování velké části kódu.

8 Výstup aplikace

Na konci procesu hodnocení aplikace kompiluje a zobrazuje všechny nezbytné údaje v podobě strukturované tabulky, která slouží jako finální výstupní rozhraní pro uživatele. Tato tabulka představuje agregovaný přehled výpočtů, které byly odvozeny z dříve zadaných dat a z příslušných podílových tabulek.



Obrázek 6 - Ukázka finální tabulky s výslednými daty, zdroj: vlastní

Struktura tabulky je následující:

- VUs (Výchozí užitnost): Tato hodnota vždy vychází ze 100 %, což je výchozí bod pro postupné snižování hodnoty vozidla v rámci oceňovacího procesu.
- ZAs (Základní amortizace): Již jsme si vysvětlili, jak se vypočítává – je to aritmetický průměr hodnot ZAD a ZAP.
- PSs (Přirážka/srážka): Toto pole je jediné, které může uživatel ovlivnit, a umožňuje nastavit buď přirážku nebo srážku. Tyto hodnoty ovlivňují konečný výpočet PZUs a ZUrv a jsou podloženy zdůvodněním znalce ve svém posudku.
- ZUs (Zbytková užitnost): Vypočítá se podle vzorce, který započítává VUs, ZAs a PSs a představuje zbývající hodnotu vozidla.

$$ZUs = \frac{(VUs * (100 - ZAs) * (100 + PSs))}{10^4} [\%]$$

- PDs (Poměrný díl): Tyto údaje, převzaté z tabulek, ukazují rozdělení celkové hodnoty vozidla mezi jeho jednotlivé části.

- PZUs (Poměrná zbytková užitnost): Tato hodnota udává poměrný podíl hodnoty každé komponenty vozidla vzhledem k celkové hodnotě vozidla, vypočítává se jako produkt VUs, ZAs, PSs a PDs.

$$PZUs = \frac{((VUs * (100 - ZAs) * (100 + PSs) * PDs))}{10^6} [\%]$$

- ZUrv (Zbytková užitnost redukovaného vozidla): Tento klíčový údaj, který je cílem celého hodnotícího procesu, odhaduje procentuální hodnotu vozidla v porovnání s jeho původní cenou. ZUrv je součtem všech hodnot PZUs a poskytuje ucelený pohled na aktuální hodnotu vozidla.

Celkový přehled těchto hodnot v tabulce umožňuje znalcům i dalším uživatelům programu provést komplexní hodnocení vozidla a stanovit jeho hodnotu s maximální možnou přesností.

Výpočet v kódu:

```
decimal ZUs = (VUs * (100 - ZAs) * (100 + PSs)) / 10000;
decimal PZUs = ((VUs * (100 - ZAs) * (100 + PSs) * PDs)) / 1000000;
```

Aplikace také umožňuje jednoduchý export dat do formátu .docx po dokončení výpočtů, což znalcům poskytuje možnost přenést vypočtené hodnoty spolu s kompletní tabulkou do jejich znaleckých posudků. Dokument .docx obsahuje všechny relevantní informace, včetně mezi výpočtových dat jako ZAD a ZAP, což znalci umožňuje tyto údaje explicitně zdokumentovat.

Díl	VUs	ZAs	PSs	ZUs	PDs	PZUs
Motor	100	40,2	0	59,80	21,0	12,56
Rám	100	40,2	0	59,80	5,0	2,99
Karoserie	100	40,2	0	59,80	20,0	11,96
Výbava karoserie	100	40,2	0	59,80	20,0	11,96
Rozdělovací převodovka	100	40,2	0	59,80	12,0	7,18
Zadní náprava + Rozvodovka	100	40,2	0	59,80	8,0	4,78
Přední náprava + Rozvodovka	100	40,2	0	59,80	14,0	8,37

Značka: Audi

Model: A4

Datum první registrace vozidla: 27.04.2018 0:00:00

Datum ocenění: 27.04.2024 20:39:32

Počet najetých kilometrů: 26000

VIN: F1RED5G1GR1

Typ motoru: Spalovací

Výkon: 180

ZAD: 60,0

ZAP: 20,4

ZUrv: 59,80

COB: 59,80

Obrázek 7 - Ukázka finálního exportu dat do .docx dokumentu, zdroj: vlastní

Exportovaný dokument zahrnuje nejen konečné hodnoty, ale i vstupní údaje z první fáze vyplňování aplikace, například značku a model vozidla. Dále obsahuje i hodnotu COB (Cena obvyklá), která je vypočítána ze ZUrv a předem zadaného koeficientu prodejnosti. Tento koeficient byl určen v počáteční fázi programu.

Kompletní přehled v tabulce, spolu s možností exportu dat, zaručuje, že znalci mohou provádět komplexní hodnocení vozidla a předkládat hodnoty s maximální přesností a důkladností v souladu s požadavky znalecké praxe.

8.1 Export dokumentu

Pro export dat do dokumentu formátu .docx v programu je využita třída *DataExporter*, která využívá knihovnu DocumentFormat.OpenXml. Tato knihovna poskytuje rozhraní pro manipulaci s Open XML dokumenty. Je to poměrně běžný standard používaný Microsoft Office dokumenty, jako jsou Word, Excel nebo PowerPoint. [11]

Třída *DataExporter* obsahuje metodu *ExportData*, která přijímá cestu k souboru, data z tabulky a aktuální objekt posudku. Metoda nejprve vytvoří nový dokument pomocí WordprocessingDocument.Create s cestou k souboru a typem dokumentu. Po vytvoření hlavní části dokumentu přidá tělo dokumentu (*Body*) a následně vytvoří tabulku s daty z tabulky (*CreateTable*) a tuto tabulku vloží do dokumentu.

```
using (WordprocessingDocument wordDocument =
WordprocessingDocument.Create(filePath,
DocumentFormat.OpenXml.WordprocessingDocumentType.Document))
{
// Přidání hlavní části dokumentu
MainDocumentPart mainPart = wordDocument.AddMainDocumentPart();
mainPart.Document = new Document();

Body body = mainPart.Document.AppendChild(new Body());

// Vytvoření tabulky
Table table = CreateTable(tableData);
body.Append(table);

// Přidání informací o posudku
AddExpertReviewInfo(currentExpertReview, body);

mainPart.Document.Save();
}
```

Dále metoda *AddExpertReviewInfo* přidává do dokumentu informace o expertním posudku. Pro každou vlastnost objektu *ExpertReview* se vytvoří nový odstavec v dokumentu. Pokud pro vlastnost existuje překlad ve slovníku DictionaryStorage.Translations, vytvoří se odstavec s tučně označeným překladem a hodnotou vlastnosti.

V případě výjimky, například pokud je soubor již otevřený a nelze ho přepsat, zobrazí se chybová zpráva.

Na závěr metoda otevře soubor ve výchozím programu pro .docx soubory, což typicky bude Microsoft Word, a uživatel může dokument zkontrolovat a dále s ním pracovat.

```
System.Diagnostics.Process.Start("explorer.exe", filePath);
```

9 Pomocné nástroje

9.1 Práce se slovníky

V předešlé části bylo zmíněno, že program využívá slovníky k zpřístupnění technické terminologie pro uživatele. Slovníky *PropertyDisplayNames* a *Translations* nalezneme ve statické třídě *DictionaryStorage*. *PropertyDisplayNames* slouží k zobrazování názvů vlastností vozidla v uživatelském rozhraní, zatímco *Translations* zajišťuje překlad obecných termínů vyskytujících se v aplikaci. Díky tomuto přístupu aplikace umožňuje uživatelům snadné porozumění technickým aspektům a podporuje její lokalizaci.

Slovník ‚*Translations*‘:

```
public static readonly Dictionary<string, string> Translations = new
Dictionary<string, string>
       {
         { "Brand", "Značka" },
         { "Model", "Model" },
         { "FirstRegistrationDate", "Datum první registrace vozidla" },
         { "AwardDate", "Datum ocenění" },
         { "Mileage", "Počet najetých kilometrů" },
         { "VIN", "VIN" },
         { "EngineType", "Typ motoru" },
         { "Displacement", "Zdvihový objem" },
         { "Power", "Výkon" },
         { "Fuel", "Palivo" },
         { "ZAD", "ZAD" },
         { "ZAP", "ZAP" },
         { "ZUrv", "ZUrv" },
         { "COB", "COB" }
        };
```

Ukázka slovníku „*PropertyDisplayNames*“:

```
public static readonly Dictionary<string, string> PropertyDisplayNames = new
Dictionary<string, string>
        {
         {"Engine", "Motor"},
         {"Transmission", "Převodovka"},
         {"DriveShaft", "Kardan"},
         {"RearAxle", "Zadní náprava"},
         {"FrontAxle", "Přední náprava"},
         {"Frame", "Rám"},
         {"Body", "Karoserie"},
         {"BodyEquipment", "Výbava karoserie"},
         {"DriveShaft_RearAxle", "Kardan a zadní náprava"},
         {"Transmission_DriveShaft_RearAxle", "Převodovka, kardan a zadní
         náprava"},
         {"Transmission_DriveShaft", "Převodovka a kardan"}, // pokračuje dále
```

9.2 Konvertory

Aplikace zahrnuje soubor konvertorů, což jsou drobnější pomocné nástroje pro překlad mezi různými datovými typy a formáty. *VehicleCategoryResolver* se zaměřuje na odvození typu vozidla z jeho kategorie, což umožňuje lepší manipulaci s daty vozidel. *EnumToVisibilityConverter* a *BooleanToVisibilityConverter* řeší viditelnost prvků uživatelského rozhraní v závislosti na stavech a hodnotách, což je užitečné pro dynamické UI (User Interface – uživatelské rozhraní). Nakonec *EnumDescriptionConverter* poskytuje čitelné popisy pro výčtové hodnoty, které zlepšují srozumitelnost výstupů v aplikaci.

EnumDescriptionConverter:

```
    public class EnumDescriptionConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            Enum enumValue = value as Enum;
            if (enumValue == null)
                return string.Empty;

            return GetEnumDescription(enumValue);
        }

        private string GetEnumDescription(Enum enumValue)
        {
            FieldInfo fi = enumValue.GetType().GetField(enumValue.ToString());
            DescriptionAttribute[] attributes =
(DescriptionAttribute[])fi.GetCustomAttributes(typeof(DescriptionAttribute),
false);
            if (attributes != null && attributes.Length > 0)
                return attributes[0].Description;
            else
                return enumValue.ToString();
        }
    }
```

BooleanToVisibilityConverter:

```
    public class BooleanToVisibilityConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            bool isVisible = (bool)value;
            return isVisible ? Visibility.Visible : Visibility.Collapsed;
        }

        public object ConvertBack(object value, Type targetType, object
parameter, CultureInfo culture)
        {
            Visibility visibility = (Visibility)value;
            return visibility == Visibility.Visible;
        }
    }
```

10 Testování

Testování je důležitým prvkem v procesu vývoje softwaru, který umožňuje ověřit funkčnost aplikace a zabezpečit, že splňuje stanovené požadavky a očekávání uživatelů. Bez efektivního testování by mohly vzniknout závažné chyby, které by mohly vést k selhání celého systému. Tím jsou zde hlavně myšlené špatné výpočty.

10.1 Testování v praxi

Aplikace již prošla praktickými testy přímo v nasazení. Testování v reálných pracovních podmínkách bylo důležité pro zajištění spolehlivosti a přesnosti softwaru. Během této fáze byl program aktivně používán k provádění skutečných výpočtů, které měly praktické využití ve znaleckých posudcích.

Důležitým aspektem tohoto testování bylo manuální ověřování výsledků, kde byly porovnány s očekávanými hodnotami. Tato manuální kontrola umožnila odhalit potenciální chyby a zajistit, že program funguje přesně tak, jak by měl. Zpětná vazba z této fáze testování byla klíčová pro identifikaci oblastí, které vyžadovaly další vylepšení a úpravy.

10.2 Možná implementace testů

Implementace automatizovaných testů představuje další úroveň zabezpečení a zajištění kvality softwaru. Tyto testy mohou simulovat různé scénáře použití a prověřit aplikaci v širokém spektru podmínek a situací.

Automatizované testy jsou efektivní v identifikaci chyb a nedostatků v programu a umožňují opakované testování bez nutnosti manuální intervence. Tímto způsobem lze snížit riziko lidské chyby a zajistit konzistentní a spolehlivé výsledky. Tyto testy nebyly prozatím do projektu implementovány, ale do budoucího vývoje aplikace se takové testy plánují.

10.3 Údržba softwaru

Součástí procesu údržby softwaru bude pravidelné opravování nalezených chyb, aby se zajistila dlouhodobá funkčnost aplikace. Dále bude provedena aktualizace softwaru tak, aby odpovídala novým standardům a požadavkům v oblasti znaleckých posudků. Tento proces zajišťuje, že aplikace zůstane aktuální vzhledem k neustále se vyvíjejícím technologickým a průmyslovým trendům.

Závěr

Při pohledu zpět na dobu, kterou jsem věnoval práci na této bakalářské práci, si uvědomuji, jak cenné zkušenosti jsem získal. Bylo to období plné výzev, během kterých jsem se učil zvládat komplexní úkoly a rozvíjet své dovednosti v praktickém programování.

Tato práce mi umožnila hlouběji proniknout do problematiky vývoje software a přinutila mě přemýšlet nad každým rozhodnutím při návrhu a budování infrastruktury programu. Bylo to období intenzivního učení, kdy každý problém představoval novou příležitost k učení a každé dosažené řešení bylo krokem vpřed ve vývoji aplikace.

Ve výsledku se ale díky tomu povedlo vytvořit funkční aplikaci, která dokáže na základě přesně daných dat uživateli vyhotovit cenu vozidla podle znaleckého standardu I/2022. Tento software je krokem vpřed ve vývoji aplikací pro odborné ocenění vozidel a otevírá dveře k dalším inovacím v této oblasti.

Program byl také napsán tak, aby případné úpravy ve znaleckých tabulkách (například příchod nového standardu) nebylo těžké implementovat. Architektura aplikace je navržena s důrazem na modularitu a flexibilitu, což umožňuje snadné začlenění změn bez nutnosti zásadních zásahů do již existujícího kódu.

Největší problematikou bylo určitě pochopit postupy, jak se motorová vozidla oceňují, zjistit co vše se musí vypočítat a jaké tabulky použít, ale hlavně jak. Další výzva pak byla přenést tyto teoretické znalosti do kódu.

Jedním z nejvýznamnějších a nejnáročnějších úkolů, které jsem v průběhu práce musel řešit, bylo získání hlubokého porozumění metodikám ocenění motorových vozidel. Vyžadovalo to důkladný průzkum a studium odborné literatury a znaleckých standardů, aby bylo možné identifikovat všechny relevantní proměnné, které ovlivňují hodnotu vozidla. S tím souvisel i úkol pochopit, jaké výpočty je třeba provést, jaké tabulky jsou pro tento účel nezbytné a jakým způsobem je interpretovat a aplikovat při hodnocení.

Za těmito teoretickými znalostmi následovala snaha o ztvárnění získaného poznání do fungujícího softwaru. To zahrnovalo proces převodu abstraktních pravidel a výpočtových modelů do konkrétních algoritmů a kódů v jazyce C# a frameworku .NET, což byla sama o sobě výzva.

Je důležité zmínit, že práce na programu nekončí s odevzdáním této bakalářské práce. Vzhledem k omezenému počtu dostupných podobných nástrojů na trhu existuje velký potenciál pro další vývoj a vylepšení. Plánuji proto v programu pokračovat a rozvíjet jeho funkcionalitu, aby se mohl stát ještě užitečnějším nástrojem pro znalce a odborníky v oblasti ocenění vozidel. Těším se na další fáze vývoje a na příležitosti, které mi práce na tomto projektu přinese.

Citovaná literatura

[1] InsureDaily. *Car Valuation: Definition, Necessity and How to Compute It.* [online] 2024 [cit. 2024-21-04] Dostupné z: https://www.insuredaily.co.uk/blog/car-insurance/car-valuation

[2] KLEDUS, Robert; SEMELA, Marek; BELÁK, Michal a MAREŠ, Petr. Znalecký standard č. I/2022: oceňování silničních a zvláštních vozidel. Brno: Akademické nakladatelství CERM, 2021. ISBN 978-80-7623-076-7.

[3] KAPEŠ, Martin, soudní znalec [ústní sdělení]. Pardubice, 2024-21-04. Dostupné z: https://seznat.justice.cz/znalec/detail/2576?query=%7B"filter"%3A%7B"druhyOsoby"%3A%5B%5D%2C"fullText"%3A"kapeš"%2C"vcetnePozastavenych"%3Afalse%2C"vcetneZaniklych"%3Afalse%7D%2C"strankovani"%3A%7B"pocetNaStranku"%3A50%7D%2C"razeni"%3A%7B"hodnota"%3A"jmeno"%2C"smer"%3A0%7D%7D

[4] Knihobot. *Znalecký standard č. I/2022*. [online] 2021 [cit. 2024-21-04] Dostupné z: https://knihobot.cz/g/3626466

[5] Microsoft. *A Tour of C#.* [online] 2024 [cit. 2024-21-04] Dostupné z: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[6] Microsoft. *Entity Framework Core.* [online] 2021 [cit. 2024-21-04] Dostupné z: https://learn.microsoft.com/en-us/ef/core/

[7] Microsoft. *Desktop Guide (WPF .NET).* [online] 2023 [cit. 2024-21-04] Dostupné z: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-6.0

[8] SQLite. *Appropriate Uses For SQLite.* [online] 2024 [cit. 2024-21-04] Dostupné z: https://sqlite.org/whentouse.html

[9] Microsoft. *Co je Visual Studio.* [online] 2023 [cit. 2024-21-04] Dostupné z: https://learn.microsoft.com/cs-cz/visualstudio/get-started/visual-studio-ide?view=vs-2022

[10] Microsoft. *Model-View-ViewModel (MVVM).* [online] 2022 [cit. 2024-22-04] Dostupné z: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

[11] NuGet. *DocumentFormat.OpenXml*. [online] 2024 [cit. 2024-28-04] Dostupné z: https://www.nuget.org/packages/DocumentFormat.OpenXml