MASARYKOVA
UNIVERZITA

FAKULTA INFORMATIKY

CRM webová aplikace pro malou firmu

Bakalářská práce

JAN GUČA

Brno, jaro 2023

MASARYKOVA UNIVERZITA

FAKULTA INFORMATIKY

CRM webová aplikace pro malou firmu

Bakalářská práce

JAN GUČA

Vedoucí práce: doc. Mgr. Jan Obdržálek, PhD.

Katedra teorie programování

Brno, jaro 2023

FACULTAS ARTIS INFORMATICÆ • UNIVERSITAS MASARYKIANA •

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Jan Guča

Vedoucí práce: doc. Mgr. Jan Obdržálek, PhD.

Poděkování

Rád bych poděkoval vedoucímu této práce doc. Mgr. Janu Obdržálkovi PhD. za cenné rady a konstruktivní kritiku během psaní a za jeho ochotu a čas, který mi věnoval. Také bych chtěl poděkovat mým kolegům z Pinya s.r.o. za jejich důvěru, podporu a pomoc.

Shrnutí

Tato práce se věnuje vývoji CRM webové aplikace. Cílem je vytvořit nový produkt pro softwarovou společnost Pinya s.r.o., který bude určen pro menší firmy. Aplikace by měla sloužit jako základ pro možné zájemce, kterým by byla vytvořena vlastní instance této aplikace s úpravami a přidanou funkcionalitou dle jejich potřeb.

Klíčová slova

CRM, C#, .NET, .NET 6, ASP.NET Core, MVC, Kendo UI, webová aplikace

Obsah

Úvod **1**

1 CRM **2**
1.1 Analýza dostupných aplikací 2
1.1.1 Salesforce CRM 3
1.1.2 HubSpot CRM 3
1.1.3 Shrnutí 4

2 Analýza a návrh **5**
2.1 Specifikace požadavků 5
2.1.1 Funkční požadavky 5
2.1.2 Nefunkční požadavky 6
2.2 Diagram případů užití 7
2.2.1 Rozdělení do modulů 7
2.2.2 Aktéři 7
2.2.3 Případy užití 8
2.3 Datový model 9
2.3.1 Datový model aplikace 9

3 Použité technologie **14**

4 Implementace **18**
4.1 Architektonické a návrhové vzory 18
4.1.1 Model-View-Controller architektura 18
4.1.2 Repositář (Repository pattern) 19
4.1.3 Jednotka práce (Unit of Work) 21
4.2 Struktura aplikace 22
4.2.1 CRMBase.Shared 23
4.2.2 CRMBase.Model 25
4.2.3 CRMBase.Data 26
4.2.4 CRMBase 27

5 Testování **32**

6 Závěr **34**
6.1 Možná rozšíření 34

Bibliografie **36**

A Ukázky uživatelského rozhraní aplikace **38**

B Spuštění aplikace **42**

C Přílohy **44**

Seznam obrázků

2.1 Diagram případů užití 9
2.2 Entitně-relační diagram navrhované aplikace 13

4.1 Diagram MVC architektury 19
4.2 Diagram repositáře 21
4.3 Diagram jednotky práce 22
4.4 Diagram závislostí mezi projekty 23
4.5 Diagram rodičovských tříd a rozhraní entit 25
4.6 Nedostatečná oprávnění 31

A.1 Přihlašování 38
A.2 Domovksá stránka 38
A.3 Správa uživatelů 39
A.4 Role a oprávnění 39
A.5 Formulář pro editaci šablony zpráv 40
A.6 Formulář pro vytvoření obchodní příležitosti 40
A.7 Karta zákazníka 41

Úvod

CRM (Customer Relationship Management) systémy jsou softwarové nástroje určené pro správu vztahů se zákazníky. Tyto systémy slouží k organizaci a centralizaci informací o zákaznících, interakcích s nimi a jejich nákupních preferencí. Tyto informace pomáhají společnostem lépe porozumět potřebám zákazníků a lépe poskytovat služby, což může vést ke zlepšení spokojenosti zákazníků [1].

Cílem této bakalářské práce je navrhnout a vytvořit vlastní CRM aplikaci, která se stane nově nabízeným produktem společnosti Pinya s.r.o.[1]. Tato aplikace bude primárně sloužit jako základ pro úpravu a rozšíření dle konkrétních potřeb možných zákazníků.

V první kapitole je popsán koncept CRM systémů a jejich základní funkcionality. Součástí této kapitoly je také představení některých existujících CRM systémů, z nichž jsou dva blíže analyzovány. Třetí kapitola se věnuje analýze a návrhu, kde jsou s využitím diagramů popsány specifikace požadavků a datový model. Ve čtvrté kapitole jsou popsány použité technologie. Pátá kapitola je věnována implementaci aplikace, konkrétně jsou zde popsány použité architektonické a návrhové vzory a struktura aplikace, včetně popisu některých částí kódu. Poslední kapitola je věnována testování a jeho průběhu.

Výsledkem práce je funkční webová aplikace, která obsahuje základní funkcionalitu pro řízení vztahů se zákazníky. Aplikace umožňuje správu zákazníků, potenciálních zákazníků, kontaktních osob a komunikace s nimi, produktů a obchodních příležitostí. Součástí aplikace je také správa zaměstnanců, úkolů a aktivit.

1. Pinya s.r.o. je softwarová společnost se specializací na tvorbu softwarových řešení jako Pinya HR, Pinya SharePoint a další. `https://www.pinya.cz`

1 CRM

CRM (Customer Relationship Managemenet) je strategie a metoda, která se zaměřuje na budování a udržování vztahů se zákazníky a potenciálními zákazníky. To zahrnuje všechny interakce mezi organizací a zákazníky, včetně prodeje.

CRM systémy jsou softwarové nástroje, které organizacím pomáhají spravovat všechny tyto interakce a informace o zákaznících. Základní funkcionalita CRM systémů zahrnuje:

- Správu zákazníků
- Správu potenciálních zákazníků
- Správu kontaktních osob
- Správu interakcí se zákazníky/kontaktními osobami
- Správu nabízených produktů
- Správu prodeje, obchodních příležitostí, smluv apod.

Součástí funkcionality CRM systémů bývá často i podpora integrace s jinými aplikacemi a marketingových funkcí.

CRM systémy jsou používány v různých odvětvích, včetně maloobchodu, výroby, služeb, lékařství a dalších. Hlavním cílem CRM systémů je pomoci organizacím lépe porozumět zákaznickým potřebám a vytvářet dlouhodobé a profitabilní vztahy se zákazníky [1].

1.1 Analýza dostupných aplikací

Existuje mnoho domácích i zahraničních CRM systémů. Příkladem můžou být Salesforce CRM [2], HubSpot CRM [3], Zoho CRM [4], eBrána [5], Atollon [6] a mnoho dalších. Tyto CRM systémy poskytují podobnou funkcionalitu jakou by měla mít tato aplikace, a proto bylo potřeba zaměřit se na detailnější analýzu některých z uvedených systémů. Z těchto systémů byly vybrány Salesforce CRM a HubSpot CRM, které jsou v následujících podkapitolách blíže popsány a analyzovány.

1.1.1 Salesforce CRM

Salesforce CRM je jedním z největších a nejkomplexnějších CRM řešení na trhu. Jeho hlavní výhodou je rozsáhlá komunita uživatelů a široká nabídka integrací s dalšími aplikacemi. Nicméně, velkou nevýhodou tohoto systému je, že je placený a nemá žádnou základní verzi, která by byla zdarma. Pro možnost vyzkoušení nabízí třicetidenní zkušební verzi.

Systém poskytuje základní funkcionalitu CRM systémů pro správu zákazníků, potenciálních zákazníků, kontaktních osob, interakcí se zákazníky a obchodních příležitostí, ale chybí zde správa nabízených produktů. Salesforce CRM umožňuje editaci datového modelu, což umožňuje upravovat existující datové objekty nebo vytvářet nové. Nabízí také širokou škálu marketingových funkcí. Dále poskytuje možnost integrace s mnoha aplikacemi do systému a také umožňuje integrovat tento systém do jiných aplikací pomocí vlastního API. V systému lze také nastavit vlastní oprávnění a uživatelské role, rozložení všech stránek a pohledů.

Vzhledem k množství možností, informací a funkcí tohoto systému působí uživatelské rozhraní velmi komplikovaně. Mnoho funkcí, zejména různá nastavení, je potřeba poměrně složitě vyhledávat.

Celkově lze říci, že Salesforce CRM je velmi výkonný a rozsáhlý systém, ale může být náročný pro uživatele s menší zkušeností nebo ty, kteří potřebují jednodušší CRM řešení.

1.1.2 HubSpot CRM

HubSpot CRM patří mezi nejvíce používané CRM systémy pro menší firmy. Jeho základní verze je zdarma, což ho činí velmi atraktivním řešením pro malé firmy s omezeným rozpočtem. Stejně jako Salesforce CRM nabízí zkušební verzi, ale pouze po dobu dvou týdnů.

Tento systém poskytuje základní funkcionalitu CRM systémů. Správa zákazníků, potenciálních zákazníků a kontaktních osob je v tomto systému nahrazena správami společností a kontaktů. Na rozdíl od Salesforce CRM zahrnuje také správu nabízených produktů. Umožňuje také upravovat a přidávat vlastní pole do existujících objektů. Existuje také možnost přidávat vlastní objekty, avšak tato funkcionalita je dostupná pouze v placených licencích. Hubspot CRM nabízí

několik fmarketingových funkcí, včetně odesílání emailových kampaní. Podporuje integraci s různými aplikacemi a má svou vlastní API pro integraci HubSpot CRM do jiných systémů. Systém umožňuje nastavení oprávnění uživatelů.

Stejně jako Salesforce CRM, i tento systém nabízí mnoho funkcí. Uživatelské rozhraní tohoto systému může působit velmi komplikovaně vzhledem k množství zobrazovaných informací. Výhodou je možnost nastavení všech pohledů, a tedy dovoluje upravit si uživatelské rozhraní dle svých potřeb.

1.1.3 Shrnutí

Funkcionalita analyzovaných systémů si je velmi podobná. Oba tyto systémy obsahují většinu funkcionality jako aplikace této bakalářské práce, ale obsahují také další funkcionalitu jako je marketing, integrace s dalšími aplikacemi a možnost vytváření vlastních datových objektů. Tyto funkce přidávají na složitosti práce s těmito systémy.

Uživatelské rozhraní obou analyzovaných systémů působí poněkud komplikovaně, což je způsobeno množstvím nabízené funkcionality. Plusem je možnost nastavení množství zobrazených informací na všech kartách. V obou systémech chybí samostatná evidence zaměstnanců. Uživatelé a zaměstnanci v těchto systémech představují stejný objekt, nejsou tedy rozděleni zaměstnanci firmy používající tyto systémy od systémových administrátorů a případných dalších vnějších uživatelů. V systému Salesforce CRM chybí evidence konkrétních nabízených produktů/služeb.

S ohledem na uvedené nedostatky, jako je chybějící rozdělení zaměstnanců a uživatelů a evidence nabízených produktů/služeb a složité uživatelské rozhraní, bude cílem této bakalářské práce navrhnout a naimplementovat aplikaci, která bude obsahovat základní funkcionalitu CRM systémů popsanou v kapitole 1 a která bude mít přívětivé a jednoduché uživatelské rozhraní.

2 Analýza a návrh

Tato kapitola je věnována analýze a návrhu. První část je věnována specifikaci funkčních a nefunkčních požadavků. Další část se věnuje diagramu případů užití, kde jsou funkční požadavky rozděleny do jednotlivých logických celků (modulů). V poslední části je popsán datový model vyvíjené aplikace.

2.1 Specifikace požadavků

Součástí analýzy je specifikace požadavků. V této kapitole jsou definovány funkční i nefunkční požadavky. Sběr těchto požadavků probíhal ve spolupráci s kolegy ze společnosti Pinya s.r.o.

2.1.1 Funkční požadavky

Funkční požadavky na aplikaci vychází z běžné funkcionality CRM systémů popsánu v kapitole 1 a z konkrétních požadavků daných během analýzy. Během analýzy požadavků byly určeny následující funkční požadavky:

- Správa uživatelů – přidávání, úprava a odstraňování uživatelů. Vygenerování nového hesla pro uživatele.
- Správa uživatelských rolí a oprávnění – přidávání, úprava a odstraňování uživatelských rolí. Úprava oprávnění jednotlivých rolí pro všechny moduly aplikace.
- Správa číselníků – přidávání, úprava a odstraňování položek číselníků u jednotlivých typů.
- Správa šablon zpráv – úprava existujících šablon. Šablony zpráv budou určeny pro systémové a informační zprávy pro uživatele, jako je vytvoření účtu, změna hesla a vytvoření nového úkolu. Proto se nedají vytvářet nové.
- Správa zaměstnanců – přidávání, úprava a odstraňování zaměstnanců.

- Správa úkolů – přidávání a úprava úkolů. Úkoly se nedají odstranit, dají se pouze zrušit.
- Správa aktivit – přidávání, úprava a odstraňování aktivit.
- Kalendář aktivit.
- Správa produktů/služeb – přidávání, úprava a odstraňování produktů/služeb.
- Správa leadů (potenciálních zákazníků) – přidávání a úprava.
- Správa zákazníků – přidávání a úprava.
- Správa kontaktních osob zákazníků – přidávání, úprava a odstraňování kontaktních osob u jednotlivých zákazníků.
- Správa komunikace – přidávání, úprava a odstraňování komunikace se zákazníky.
- Správa obchodních příležitostí – přidávání a úprava.
- Exporty a reporty - možnost exportu některých tabulek do excelu a možnost generování reportů/hlášení.

2.1.2 Nefunkční požadavky

Během analýzy požadavků byly určeny následující nefunkční požadavky:

- Uživatelská přívětivost – uživatelské rozhraní bude jednoduché a přehledné.
- Rozšiřitelnost – aplikace bude připravena na možná rozšíření. Přidání nové funkcionality do systému bude vyžadovat minimální zásah do již existující funkcionality systému.
- Bezpečnost – do aplikace budou mít přístup pouze vytvoření uživatelé.

- Technologie - aplikace bude vyvíjena na platformě .NET 6 v programovacím jazyce C#. Pro ukládání dat bude použita databáze Microsoft SQL Server. Při vývoji uživatelského rozhraní bude využita knihovna Kendo UI.

Požadavek na přívětivé uživatelské rozhraní vyplývá z analýzy dostupných CRM systémů v kapitole 1.1. Použité technologie jsou odvozeny od technologií používaných při vývoji všech aplikací společnosti Pinya s.r.o.

2.2 Diagram případů užití

Tato podkapitola popisuje rozdělení funkcionality aplikace do jednotlivých modulů, návrh možných aktérů a diagram případů užití.

Diagram případů užití je jedním z typů diagramů v oblasti softwarového inženýrství, který slouží k popisu interakcí mezi aktéry a systémem. Diagram je grafickým zobrazením funkčních požadavků na systém.

Na diagram 2.1 je znázorněno rozdělení funkcionality aplikace do jednotlivých modulů. Tento diagram také zobrazuje ilustrativní aktéry.

2.2.1 Rozdělení do modulů

Některá funkcionalita aplikace nemusí být přístupná pro všechny uživatele. Z tohoto důvodu bylo rozhodnuto rozdělit tuto aplikaci do jednotlivých modulů, které budou představovat určitou logickou část funkcionality aplikace. Uživatelé budou mít pro každý z modulů nastavená oprávnění, která budou určovat, jaké akce bude mít uživatel povoleny nebo jestli bude mít uživatel k danému modulu přístup. Na diagramu 2.1 jsou zobrazeny moduly aplikace a jejich funkcionalita.

2.2.2 Aktéři

Aktéři jsou osoby, skupiny lidí nebo jiné systémy, kteří komunikují s tímto systémem.

Aplikace bude umožňovat vlastní nastavení uživatelských rolí a jejich oprávnění. Díky této funkcionalitě bude možné si libovolně přidávat aktéry do systému. Pro ilustraci byli definováni následující aktéři:

- Řadový zaměstnanec - tento aktér představuje všechny zaměstnance. Má přístup do systému a má přístup k modulům *Aktivity* a *Úkoly*. Může tedy spravovat aktivity a úkoly a má přístup ke kalendáři.

- HR Manager - tento aktér představuje personální tým zaměstnanců. Dědí oprávnění od řadového zaměstnance a navíc má přístup k modulu *Zaměstnanci*. Může tedy spravovat informace o zaměstnancích.

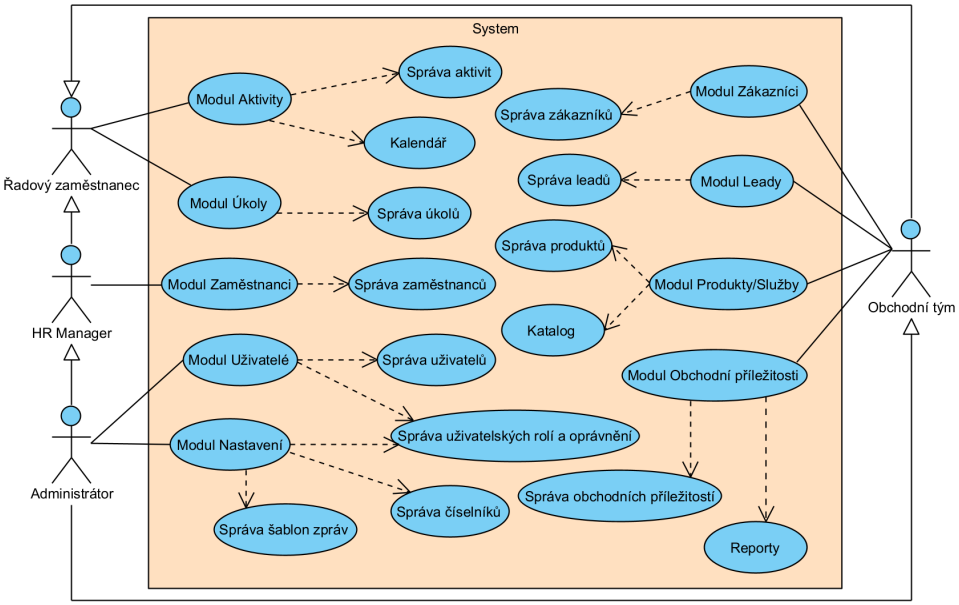
- Obchodní tým - tento aktér představuje obchodní tým. Dědí oprávnění od řadového zaměstnance a navíc má přístup k modulům *Zákazníci*, *Leady*, *Produkty/Služby* a *Obchodní příležitosti*. Může tedy spravovat zákazníky a poteciální zákazníky (leady), nabízené produkty/služby a obchodní příležitosti. Navíc může vytvářet reporty o vývoji obchodních příležitostí.

- Administrátor - tento aktér představuje administrátory systému. Dědí oprávnění od katérů *HR Manager* a *Obchodní tým* a navíc má přístup k modulům *Uživatelé* a *Nastavení*. Může tedy spravovat uživatele, uživatelské role, oprávnění, číselníky a šablony zpráv. Tento aktér má přístup ke všem funkcím aplikace.

2.2.3 Případy užití

Diagram případů užití 2.1 obsahuje požadovanou funkcionalitu z podkapitoly 2.1.1 rozdělenou do jednotlivých modulů.

Případy užití popisují interakce mezi aktéry a systémem a slouží k pochopení požadavků uživatelů a specifikaci funkcí aplikace. Každý případ užití reprezentuje konkrétní cíl, kterého uživatel chce dosáhnout a obsahuje popis kroků nebo činností vykonávaných uživatelem. Při vývoji aplikace mohou být případy užití použity k ověření, zda aplikace splňuje požadovanou funkcionalitu a zda je v souladu s požadavky uživatelů.



Obrázek 2.1: Diagram případů užití

2.3 Datový model

Návrh datového modelu aplikace je klíčovým krokem při vývoji systému, protože definuje strukturu a vztahy mezi jednotlivými datovými entitami. Pro reprezentaci datového modelu aplikace byl zvolen entitně-relační diagram (ERD), který vizualizuje jednotlivé entity a vztahy mezi nimi. V této kapitole jsou popsány jednotlivé datové entity, které jsou poté zobrazeny v ERD 2.2.

Navržený datový model byl implementován jako databáze, která slouží k ukládání a správě dat v rámci aplikace.

2.3.1 Datový model aplikace

Pro tuto aplikaci bylo definováno patnáct entit, které jsou zobrazeny v entitně-relačním diagramu 2.2. Všechny entity obsahují unikátní identifikátor *Id*, který je zároveň jejich primárním klíčem. Jednotlivé entity aplikace jsou popsány níže:

Nastavení

- *Role* - entita, která reprezentuje uživatelskou roli. Tato entita obsahuje informace o názvu role. Obsahuje také informaci, zda se jedná o roli vlastníka systému. Uživatelé s rolí vlastníka mají vždy přístup ke všemu, bez ohledu na nastavená oprávnění. Role vlastníka je vytvořena automaticky během vytvoření databáze. V aplikaci není možné vytvořit novou roli vlastníka.

- *Permission* - entita reprezentující oprávnění pro modul. Tato entita obsahuje informace o modulu, oprávnění a příslušné uživatelské roli. Oprávnění je řešeno jako bitové pole pěti bitů, kde každý bit reprezentuje jeden typ oprávnění. Typy oprávnění jsou: zobrazit mé, zobrazit vše, přidat, upravit a odstranit. Jelikož každé oprávnění patří určité uživatelské roli a každá role má oprávnění pro různé moduly, je mezi entitami *Role* a *Permission* relační vztah 1:N.

- *Field* - entita, která reprezentuje položky číselníků. Obsahuje informace o typu číselníku, jeho názvu, kódu, pořadí v rozbalovacích seznamech (komboboxech) a poznámce. Tato entita má 1:N relační vztah sama se sebou. Tento vztah určuje rodiče položky, a tak lze definovat hierarchii položek.

- *MessageTemplate* - entita, která reprezentuje šablony emailových zpráv. Tato entita obsahuje informace jako typ šablony, předmět zprávy, odesílatel, příjemci a obsah zprávy.

Uživatelé a zaměstnanci

- *User* - entita reprezentující uživatele aplikace. *User* obsahuje informace o uživatelském jménu, heslu a zda je nebo není uživatel aktivní. Dále také obsahuje informace o e-mailové adrese, uživatelské roli a příslušném zaměstnanci. Entita *User* má s entitou *Employee* 1:1 relační vztah. Mezi entitami *User* a *Role* je relační vztah 1:N.

- *Employee* - entita pro zaměstnance obsahující informace o jménu, příjmení, čísle, telefonním čísle, pracovní pozici, oddělení, stře-

disku a nadřízeném. Číslo zaměstnance musí být v rámci aplikace unikátní. Jelikož jsou informace o pracovní pozici, oddělení a středisku definovány v číselníku, jsou mezi entitami *Field* a *Employee* tři různé 1:N relační vztahy.

- *Activity* - entita reprezentující plánovanou aktivitu uživatele. Obsahuje informace o názvu, začátku a konci aktivity. Každá aktivita má svůj typ, uživatele, kterému je tato aktivita naplánována, a kontaktní osobu, které se může tato aktivita týkat. Proto má tato entita tři 1:N relační vztahy, konkrétně s entitami *Field*, *User* a *Contact*.

- *Assignment* - entita reprezentující úkoly. Každý úkol obsahuje informace o názvu, typu, datu odevzdání, stavu a popisu. Tato entita má také tři 1:N relační vztahy s entitou *User*, které reprezentují zadavatele, řešitele a schvalovatele úkolu. Typ úkolu je řešen 1:N relačním vztahem s entitou *Field*.

Produkty

- *Product* - entita reprezentující nabízené produkty/služby. Tato entita obsahuje název, obrázek, cenu, popis a typ produktu/služby. Typ produktu je řešen relačním vztahem 1:N mezi entitami *Field* a *Product*.

- *PriceHistory* - entita, která reprezentuje vývoj ceny produktu v čase. Obsahuje informace o datu změny a ceně. Relační vztah 1:N s entitou *Product* definuje, ke kterému produktu záznam změny ceny patří.

Zákazníci a potenicální zákazníci

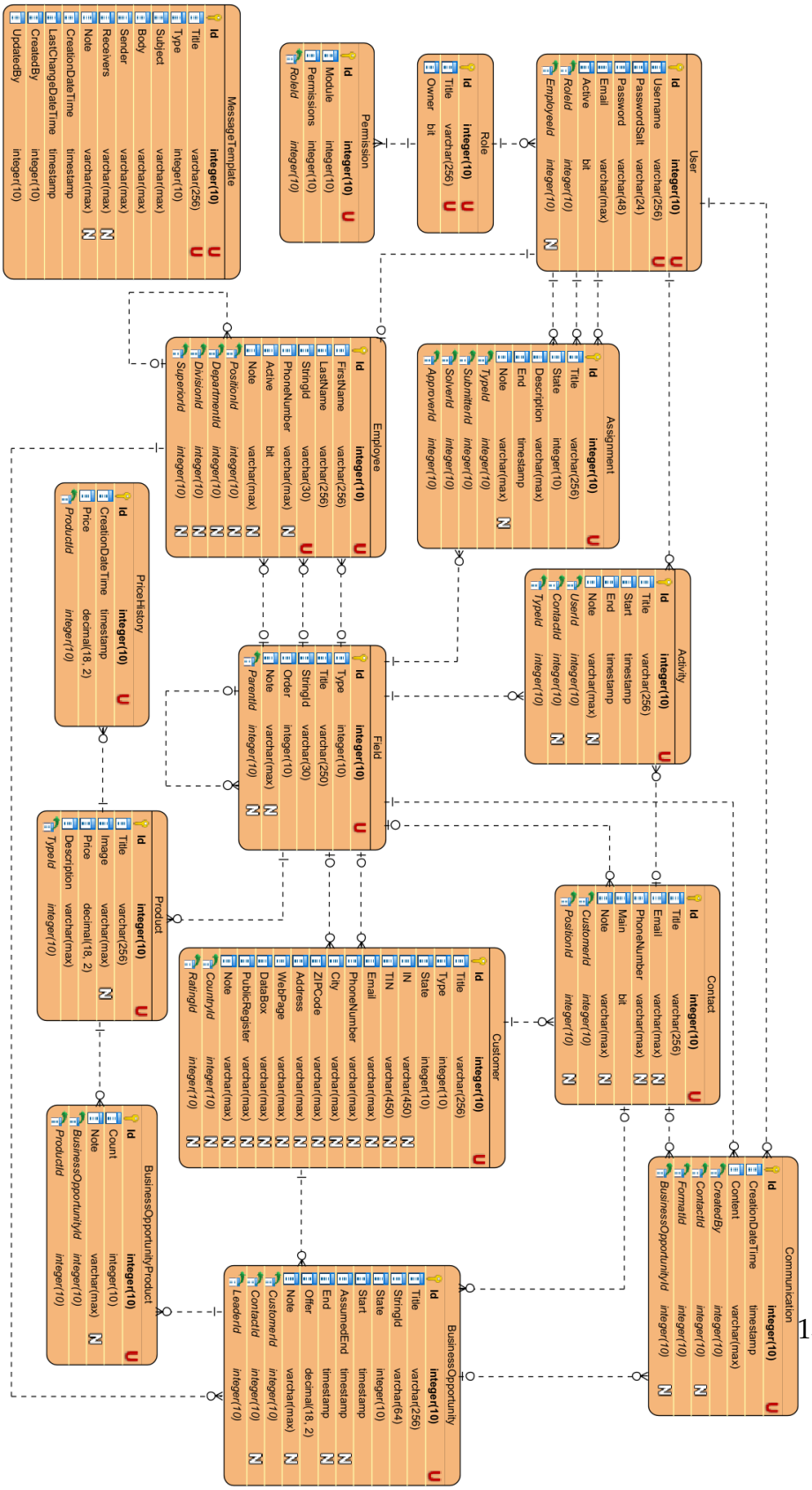
- *Customer* - entita reprezentující zákazníky a potenciální zákazníky. Tato entita obsahuje informaci o typu, která udává, jestli se jedná pouze o potenciálního nebo úplného zákazníka. Dalšími informacemi obsaženými v této entitě jsou název, IČO, DIČ, e-mailová adresa, telefonní číslo, země, město, PSČ, adresa, webová stránka, datová schránka a veřejný rejstřík. Informace

o stavu definuje stav potenciálního zákazníka. Informace o zemi je definována pomocí relačního vztahu 1:N s entitou *Field*. U potenciálních zákazníků se eviduje navíc hodnocení, které je řešeno 1:N relačním vztahem opět s entitou *Field*.

- *Contact* - entita reprezentující kontaktní osobu. Každá kontaktní osoba má k dispozici informace o svém jménu, e-mailové adrese, telefonním číslu, pracovní pozici a jestli se jedná o hlavní kontaktní osobu u zákazníka. Relační vztah 1:N mezi entitami *Customer* a *Contact* definuje, u kterého zákazníka daná kontaktní osoba pracuje, a relační vztah 1:N s entitou *Field* definuje pracovní pozici kontaktní osoby.
- *Communication* - entita reprezentující komunikaci s kontaktními osobami. Tato entita obsahuje informace ohledně obsahu, formátu, který je určen relačním vztahem 1:N s entitou *Field*, a datum vzniku. Dalšími informacemi, které tato entita obsahuje, jsou řešeny třemi relačními vztahy 1:N s entitami, konkrétně s entitami *Cotnact* (kontaktní osoba, se kterou se komunikovalo), *BusinessOpportunity* (obchodní příležitost, které se komunikace mohla týkat) a *User* (uživatel, který tuto komunikaci vytvořil).

Obchodní příležitosti

- *BusinessOpportunity* - entita reprezentující obchodní příležitosti. Každá obchodní příležitost má svůj název, kód a stav. Dalšími informacemi, které tato entita obsahuje, jsou cenová nabídka, datum impulsu, odhadované datum ukončení a skutečné datum ukončení. Zákazník, kterého se daná obchodní příležitost týká, je definován pomocí relačního vztahu 1:N s entitou *Customer*. Druhý relační vztah 1:N s entitou *Cotnact* definuje konkrétní kontaktní osobu, se kterou se daná obchodní příležitost řeší. Další relační vztah 1:N s entitou *Employee* udává vedoucího zaměstnance.
- *BusinessOpportunityProduct* - entita reprezentující položku obchodní příležitosti. Tato entita obsahuje informace o počtu kusů a produktu. Entita *BusinessOpportunityProduct* má dva 1:N relační vztahy, konkrétně s entitami *BusinessOpportunity* a *Product*.



Obrázek 2.2: Entitně-relační diagram navrhované aplikace

3 Použité technologie

Aplikace je vytvořena na platformě .NET v programovacím jazyku C# a byla vyvíjena ve vývojovém prostředí Visual Studio. Aplikace je postavena na frameworku pro tvorbu webových aplikací ASP.NET Core. Pro ukládání dat byla zvolena relační databáze Microsoft SQL Server 19 a pro práci s touto databází byl zvolen Entity Framework Core.

Uživatelské rozhraní je postaveno na technologiích CSHTML a SCSS. Pro manipulaci s uživatelským rozhraním je použita javascriptová knihovna jQuery. Pro tvorbu uživatelského rozhraní byla použita komponentová knihovna Kendo UI.

Tyto technologie byly vybrány pro svou robustnost, spolehlivost a uživatelsky přívětivé rozhraní, které umožňuje snadný vývoj a údržbu. V následující části této kapitoly jsou blíže popsány některé z použitých technologií.

.NET

.NET je open-source[1] vývojová platforma od firmy Microsoft, která podporuje vývoj webových, mobilních i desktopových aplikací. Podporovanými programovacími jazyky na této platformě jsou C#, F#, Visual Basic a IronPython. Nabízí výhody jako jsou vysoká výkonost, bezpečnost a škálovatelnost [8].

Pro vývoj této aplikace byl vybrán .NET 6, který je aktuálně poslední verzí z řady LTS[2] verzí. Nejnovější .NET 7 je STS[3] verzí.

ASP.NET Core

ASP.NET Core je open-source framework pro vývoj webových aplikací a služeb. Byl vytvořen společností Microsoft jako nástupce původní verze frameworku ASP.NET. ASP.NET Core nabízí mnoho výhod, včetně zvýšené rychlosti, vysoké flexibility, snadného používání a lehkosti.

---

1. Software s volně dostupným zdrojovým kódem [7].
2. LTS (long-term support) verze .NET jsou podporovány po dobu tří let.
3. STS (standard-term support) verze .NET jsou podporovány pouze osmnáct měsíců.

Jedním z hlavních cílů tohoto frameworku je multiplatformovost. Běží na všech hlavních operačních systémech, jako Windows, Linux a macOS. To znamená, že kód psaný pro jednu platformu je spustitelný i na ostatních platformách bez nutnosti úprav.

ASP.NET Core je postaven na základě modulární architektury, což umožňuje použití pouze potřebných funkcí bez zátěže zbytečných balíčků. Díky této modularitě lze používat například různé ORM[4] knihovny, jako je Entity Framework Core nebo Dapper.

ASP.NET Core podporuje také architekturu MVC (Model-View-Controller), která umožňuje oddělit logiku aplikace od prezentace dat. MVC lze použít pro vytvoření strukturovaného a snadno udržovatelného kódu.

Kromě toho ASP.NET Core nabízí mnoho dalších funkcí, například integrované řešení pro testování, vysokou rychlost zpracování požadavků a širokou podporu pro bezpečnostní funkce [9].

Entity Framework Core

Entity Framework Core (EF Core) je open-source ORM framework pro platformu .NET, který umožňuje práci s databázemi jako s objekty. Umožňuje definování modelu dat pomocí C# nebo VB.NET a provádění CRUD operací s těmito daty bez použití SQL příkazů.

EF Core je multiplatformní a podporuje různé databázové platformy, včetně Microsoft SQL Server, MySQL, PostgreSQL a dalších. Podporuje také použití *in-memory* databáze pro vývoj a testování aplikací, což umožňuje práci s databází bez nutnosti připojení k reálné databázi.

Je navržen tak, aby byl co nejvíce modulární a flexibilní. Obsahuje nástroje pro tvorbu databázových schémat, migrace dat a správu změn v databázi. Umožňuje také použití vlastních SQL dotazů.

Podporuje také asynchronní operace, což umožňuje aplikacím pracovat s databází efektivněji a rychleji. EF Core poskytuje také možnost cachování dat, díky čemuž mají opakované dotazy na stejná data rychlejší přístup [10].

4. Objektově-relační mapování.

Kendo UI

Kendo UI je komerční knihovna UI komponent pro webové, desktopové i mobilní aplikace, která nabízí mnoho funkcí pro tvorbu moderních uživatelských rozhraní. Nabízí různé sady komponent, včetně grafů, tabulek, formulářů, kalendářů, okének a dalších. Tyto komponenty jsou postaveny na technologiích HTML5, CSS a JavaScript [11].

Kendo UI má také rozšíření pro ASP.NET Core, které nabízí kompletní sadu UI komponent speciálně navržených pro vývoj ASP.NET Core webových aplikací. Poskytuje sadu rozhraní API pro snadné použití komponent. Tyto rozhraní API poskytují širokou škálu funkcí pro práci s daty, jako je řazení, filtrování a stránkování. Kromě toho nabízí také funkce pro snadné propojení s backendovými službami pomocí REST API [12].

Mezi další výhody Kendo UI pro ASP.NET Core patří:

- Plná podpora Bootstrapu pro snadné použití v rámci projektů, které používají Bootstrap.
- Přizpůsobitelné styly pro každou komponentu, takže vývojáři mohou snadno upravovat vzhled a chování komponent podle svých potřeb.
- Integrace s Visual Studio pro snadnou správu a vývoj aplikací.

SCSS

SCSS (Sassy CSS) je preprocesor pro CSS, který umožňuje psát styly s využitím programovacích konstrukcí, jako jsou proměnné, podmínky a funkce. SCSS je součástí rodiny preprocesorů Sass (Syntactically Awesome Style Sheets), avšak používá syntaxi podobnou klasickému CSS [13].

Hlavní výhodou SCSS je možnost použití proměnných, což zvyšuje efektivitu a snižuje pravděpodobnost chyb. Další výhodou SCSS je možnost používání funkcí, které lze použít k výpočtu hodnot a ke změně vlastností na základě různých podmínek.

Vnořené selektory jsou další výhodou SCSS. Tato funkcionalita umožňuje vnořování selektorů do jiných selektorů, což usnadňuje strukturování kódu a zlepšuje čitelnost.

Mixinování je také podporováno v SCSS a umožňuje vytváření opakujícího se kódu, který lze poté použít v různých částech kódu. To značně snižuje množství kódu, které je nutné napsat, a zvyšuje efektivitu vývoje. Dědičnost v SCSS umožňuje vytváření stylů, které dědí vlastnosti z jiných stylů.

CSHTML

CSHTML (C# Razor Syntax) je značkovací jazyk používaný v kombinaci s ASP.NET Core pro tvorbu webových stránek a aplikací. Jedná se o součást frameworku ASP.NET Core a slouží k vytváření šablon stránek.

Soubory s příponou *.cshtml* obsahují HTML kód s vloženým kódem napsaným v jazyce C#. Syntaxe jazyka CSHTML je založena na Razor Syntax, který umožňuje snadnou integraci C# kódu do HTML kódu. Kód v CSHTML lze tedy používat k dynamickému generování stránek a přizpůsobení zobrazení dat podle potřeby.

Jazyk CSHTML také umožňuje používání dalších funkcí, jako jsou direktivy, které slouží k importování jiných souborů, vkládání kódu z jiných jazyků jako například JavaScript nebo CSS a další. V rámci ASP.NET Core jsou soubory s příponou *.cshtml* používány pro tvorbu šablon pro různé části webové aplikace, jako jsou například layouty, stránky, částečné pohledy (*partial views*), komponenty a další [14].

4 Implementace

V této kapitole jsou popsány vybrané architektonické a návrhové vzory a struktura aplikace. V první části jsou popsány vybrané architektonické a návrhové vzory. Další část se věnuje struktuře aplikace, rozdělení do jednotlivých projektů a popisu některých řešených problémů, tříd nebo rozhraní.

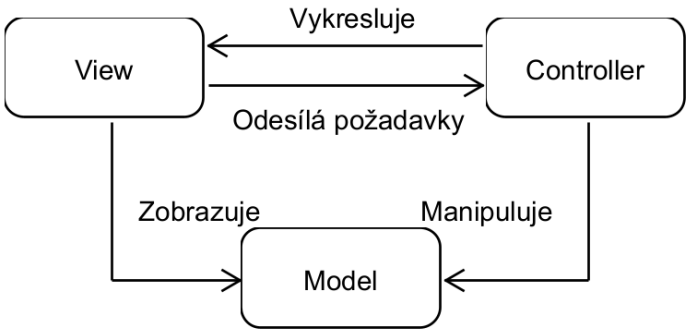
4.1 Architektonické a návrhové vzory

Tato podkapitola popisuje vybrané architektonické a návrhové vzory.

4.1.1 Model-View-Controller architektura

MVC (Model-View-Controller) je architektura softwarového návrhu, která rozděluje aplikaci do tří základních komponent: *Model*, *View* a *Controller*. Vztahy mezi těmito komponentami jsou zobrazeny na obrázku 4.1.

Model je komponenta, která reprezentuje data. Tyto data jsou poté vykreslována uživatelským rozhraním. *View* představuje uživatelské rozhraní, které zobrazuje data uživateli. *Controller* je komponenta, která se stará o komunikaci mezi ostatními dvěma komponentami. *Controller* zpracovává uživatelské požadavky, podle kterých manipuleje s daty a výsledek uloží do *Model*u. Poté předává data *View*, který je zobrazí [15].



Obrázek 4.1: Diagram MVC architektury

4.1.2 Repositář (Repository pattern)

Ukázka kódu 4.1 zobrazuje rozhraní *IRepository*, které definuje základní metody, které by měl implementovat repositář:

Ukázka kódu 4.1: Generické rozhraní repozitáře

```
public interface IRepository<TEntity> : IDisposable where TEntity :
    class, IEntity
{
    IQueryable<TEntity> All { get; }

    IQueryable<TEntity> AllAsNoTracking { get; }

    IQueryable<TEntity> AllIncluding(params Expression<Func<TEntity,
        object?>>[] includeProperties);

    IQueryable<TEntity> AllIncludingAsNoTracking(params Expression<
        Func<TEntity, object?>>[] includeProperties);

    TEntity? Find(params object[] keyValues);

    Task<TEntity?> FindAsync(params object[] keyValues);

    void Add(TEntity entity);

    Task AddAsync(TEntity entity);

    void AddRange(IEnumerable<TEntity> entities);

    Task AddRangeAsync(IEnumerable<TEntity> entities);

    void Delete(TEntity entity);

    void DeleteById(int id);
```

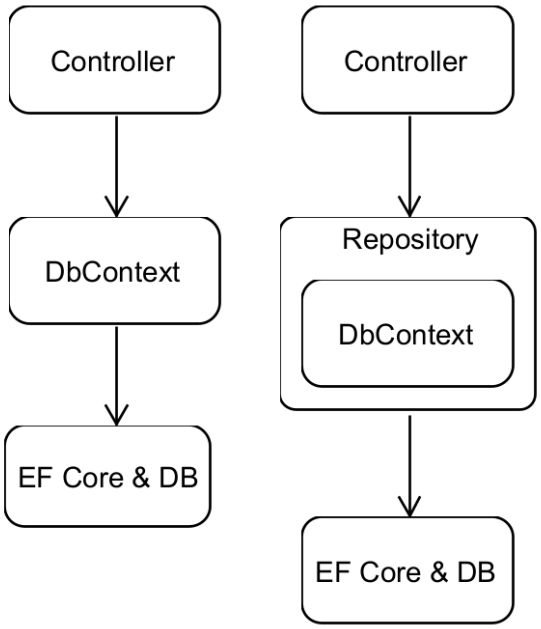
```
    void Edit(TEntity entity);

    void RollBack(TEntity entity);
}
```

Rozhraní *IRepository* obsahuje metody pro práci s daty, jako je získání entity podle primárního klíče *Find*, získání všech entit *All*, přidání entity *Add*, aktualizace entity *Edit*, odstranění entity *Delete* a další. Toto rozhraní je generické, kde typ *TEntity* představuje typ entity, se kterým repositář pracuje.

Implementace repositáře (třída *Repository*) obsahuje konkrétní implementace těchto metod, které zajišťují přístup k datům. Při implementaci byl využít ORM framework EF Core.

Využití repositáře umožňuje oddělení logiky přístupu k datům od zbytku aplikace. To zlepšuje modularitu, testovatelnost a údržbu aplikace, protože změny v přístupu k datům lze provést pouze v repositáři, aniž by bylo nutné upravovat zbytek kódu aplikace.

Repositář se často používá v kombinaci s dalšími návrhovými vzory, jako například *Unit of Work*. Na obrázku 4.2 je zobrazen rozdíl v přístupu k databázi bez použití repositáře (vlevo) a s použitím repositáře (vpravo).



Obrázek 4.2: Diagram repositáře

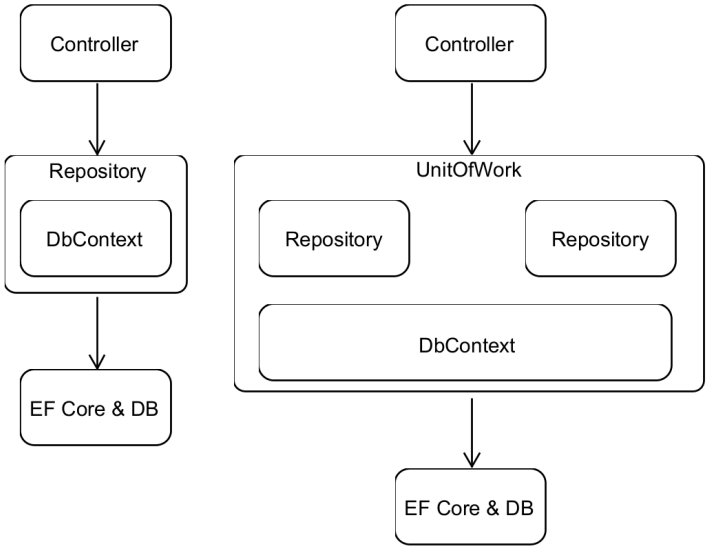
4.1.3 Jednotka práce (Unit of Work)

Implementace třídy *UnitOfWork* je zodpovědná za správu jednotného databázového kontextu a repositářů. Tato třída je rozšířením třídy *BaseUnit*, která obsahuje implementaci metod pro uložení změn, asynchronní uložení změn a uvolnění prostředků.

Třída *UnitOfWork* přijímá v konstruktoru instanci *CRMBaseDbContext* a další potřebné objekty. Obsahuje také vlastnosti pro jednotlivé repositáře, které mohou být použity pro provádění operací s daty pro danou entitu.

Použití třídy *UnitOfWork* zajišťuje, že všechny operace nad databází budou probíhat v rámci jedné transakce a že všechny změny budou uloženy až po zavolání metody *Save* nebo *SaveAsync*. To zajišťuje konzistenci dat a umožňuje provádět větší logické jednotky práce nad daty.

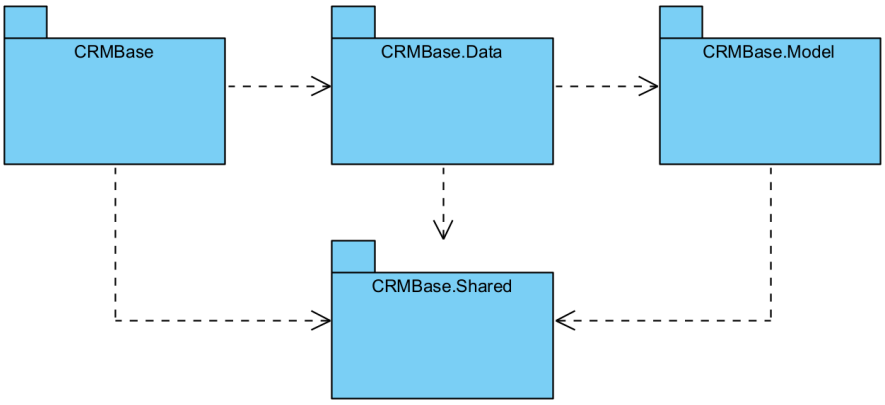
Obrázek 4.3 zobrazuje rozdíl v přístupu k databázi pouze s použitím repositářů (vlevo) a s použitím jednotky práce (vpravo).



Obrázek 4.3: Diagram jednotky práce

4.2 Struktura aplikace

Aplikace je rozdělena do čtyř projektů: *CRMBase*, *CRMBase.Data*, *CRMBase.Model* a *CRMBase.Shared*. Na diagramu 4.4 jsou zobrazeny závislosti mezi jednotlivými projekty.



Obrázek 4.4: Diagram závislostí mezi projekty

4.2.1 CRMBase.Shared

Projekt *CRMBase.Shared* je klíčovým projektem, na kterém jsou závislé všechny ostatní projekty. Obsahuje implementaci tříd, které jsou používány napříč celou aplikací. Mezi tyto třídy patří:

- *EnumExtensions*: Tato třída rozšiřuje výčtové typy a poskytuje dodatečné metody pro práci s nimi.
- *PasswordStrategy*: Metoda *GetEncryptedPassword* v této třídě slouží k získání zahashovaného hesla.
- *PasswordHelper*: Tato třída obsahuje metodu pro generování náhodného hesla, které je užitečné při vytváření nových účtů nebo při obnově hesel.
- *DropDownItem*, *Token* a *ValidationItem*: Tyto záznamy (records) slouží k reprezentaci různých datových struktur v aplikaci, například položek v rozevíracích nabídkách, validace, atd.

Tyto třídy a záznamy v projektu *CRMBase.Shared* poskytují společné funkcionality a datové struktury, které jsou potřebné v rámci celé aplikace.

EnumExtensions

Statická třída *EnumExtensions* implementuje dvě rozšiřující metody pro výčtové typy: *DisplayNameOrEnumName* a *DisplayNameOrDefault*. Tyto metody slouží k zobrazení hodnoty výčtového typu na uživatelském rozhraní. Pokud daná hodnota výčtového typu obsahuje atribut *Display*, tyto metody vrátí hodnotu vlastnosti *Name*, která je definována v tomto atributu. V případě, že hodnota výčtového typu nepoužívá tento atribut, tyto metody vrátí výchozí hodnotu. Výchozí hodnoty se liší pro každou z těchto metod. Metoda *DisplayNameOrEnumName* používá název hodnoty výčtového typu jako výchozí hodnotu, zatímco metoda *DisplayNameOrDefault* používá prázdný řetězec.

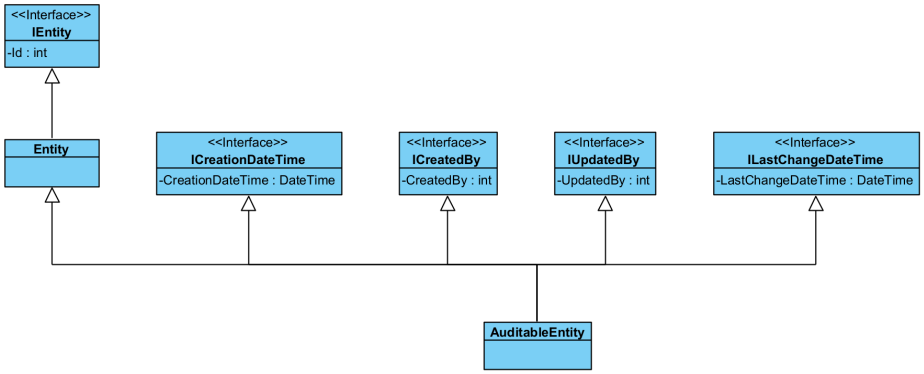
Tato třída obsahuje také metodu *ToDropDownItems*, která pro výčtový typ *type* vrací seznam položek pro rozbalovací seznamy. Záznam (record) *DropDownItem* reprezentuje jednu položku rozbalovacího seznamu a obsahuje dvě neměnné vlastnosti: *Text* a *Value*. Vlastnost *Text* určuje zobrazovaný text položky, zatímco vlastnost *Value* určuje hodnotu, která je položce přiřazena.

RequestValidation

Třída *RequestValidation* implementuje rozhraní *IRequestValidation*. Tato třída udržuje informace o průběhu validace v seznamech s prvky typu *ValidationItem*. *ValidationItem* je záznam (record), implementovaný v projektu *CRMBase.Shared*, s dvěma vlastnostmi: *Property* a *Message*. Třída *RequestValidation* obsahuje seznamy pro validační chyby, upozornění a úspěchy, ke kterým je možné přistupovat zvenku pomocí veřejných vlastností *Errors*, *Warnings* a *Successes*. Tato třída také obsahuje metody pro přidání nových položek do těchto seznamů. Konkrétně se jedná o metody *AddSuccess*, *AddWarning* a *AddError*. Tyto metody mají dvě varianty: buď přijímají dva argumenty - validační zprávu a název vlastnosti, nebo přijímají pouze jeden argument - validační zprávu. V této třídě se nachází také speciální metoda *AddInsufficientPermissionsError*, která nebere žádné argumenty a přídá pouze validační chybu pro nedostatečná oprávnění. Nakonec tato třída implementuje metodu *Clear*, která vyprázdní všechny seznamy.

4.2.2 CRMBase.Model

V projektu *CRMBase.Model* se nachází implementace výčtových typů a entit, které byly implementovány podle navrženého datového modelu popsaného v podkapitole 2.3.1. Na diagramu tříd 4.5 je zobrazena hierarchie základních entitních tříd a rozhraní.



Obrázek 4.5: Diagram rodičovských tříd a rozhraní entit

Entity

Abstraktní třída *Entity* implementuje rozhraní *IEntity* a je rodičovskou třídou všech entit. Jak je vidět na ukázce 4.2, obsahuje pouze vlastnost *Id* s atributem *Key*. Vlastnosti s atributem *Key* jsou součástí primárního klíče.

Ukázka kódu 4.2: Abstraktní třída Entity

```
public abstract class Entity : IEntity
{
    [Key]
    public virtual int Id { get; set; }
}
```

AuditableEntity

Abstraktní třída *AuditableEntity* rozšiřuje třídu *Entity* a implementuje rozhraní *ICreationDateTime*, *ILastChangeDateTime*, *ICreatedBy* a

IUpdatedBy. Tato třída obsahuje čtyři vlastnosti: *CreationDateTime*, *LastChangeDateTime*, *CreatedBy* a *UpdatedBy*. Vlastnosti *CreationDateTime* a *LastChangeDateTime* jsou typu *DateTime* a popisují, kdy byla entita vytvořena a upravena. Vlastnosti *CreatedBy* a *UpdatedBy* jsou typu *int* a popisují, kdo entitu vytvořil a upravil.

4.2.3 CRMBase.Data

Projekt *CRMBase.Data* představuje datovou vrstvu a obsahuje třídy pro manipulaci s databází. V podkapitole 3 je popsán Entity Framework Core, který zjednodušuje práci s daty. Pro vytvoření databáze pomocí tohoto frameworku je nutné implementovat třídu rozšiřující *DbContext*. V této implementaci je touto třídou *CRMBaseDbContext*.

Důležitou součástí tohoto projektu je infrastruktura využívající repositáře a jednotku práce, které byly popsány v podkapitolách 4.1.2 a 4.1.3. Nachází se zde také třída *BaseUnit*, jejíž implementace je uvedena na ukázce 4.3. Tato třída implementuje metody *Save*, *SaveAsync* a *Dispose* a slouží jako základ pro všechny pracovní jednotky. Je připravena pro možné rozšíření o logování, které by využívalo novou třídu *UnitOfLog* a vlastní DbContext.

Ukázka kódu 4.3: Abstraktní třída BaseUnit

```
public abstract class BaseUnit<TDbContext> : IBaseUnit<TDbContext>
    where TDbContext : DbContext
{
    protected readonly TDbContext _context;
    private bool _disposed;

    public TDbContext Context => _context;

    protected BaseUnit(TDbContext context)
    {
        _context = context;
    }

    public int Save()
    {
        return _context.SaveChanges();
    }

    public async Task<int> SaveAsync()
    {
        return await _context.SaveChangesAsync();
    }

    public void Dispose()
    {
```

```
        Dispose(true);
        GC.SuppressFinalize(this);
    }

    private void Dispose(bool disposing)
    {
        if (!_disposed)
        {
            if (disposing)
            {
                _context.Dispose();
            }
        }
        _disposed = true;
    }
}
```

4.2.4 CRMBase

Tento projekt využívá architektonický vzor MVC implementovaný pomocí frameworku ASP.NET Core MVC. Obsahuje implementaci kontrolerů a uživatelského rozhraní. Projekt také obsahuje implementaci rozšíření pro Kendo UI, pomocníků značek[1] a atributů.

Přidání vzdálené validace do kendo.ui.validator

Podpora vzdálené validace u vstupních polí je běžně podporována, jelikož se nachází uvnitř formuláře a využívají jQuery validátor. Nicméně editování v okně u Kendo UI tabulek nepoužívá formulář ani jQuery validátor, má svůj vlastní validátor, který vzdálenou validaci nepodporuje. V C# se vzdálená validace přidává pomocí atributu *Remote*, který se přiřadí vstupnímu poli a automaticky připraví všechny potřebné informace pro vzdálenou validaci, jako je URL adresa a další pole.

Vzdálená validace je dostupná pouze pro vstupní pole s atributem *data-val-remote-url*, ve kterém je uložena URL adresa pro vzdálenou validaci. Proto se nejprve zjišťuje, zda vstupní pole obsahuje tento atribut, a zároveň se ověřuje, jestli má vstupní pole nějakou hodnotu. Poté se z hodnoty atributu *data-val-remote-additionalfields* získají názvy dalších vstupních polí, jejichž hodnoty jsou také potřeba pro validaci. Pomocí těchto názvů se najdou potřebná vstupní pole a jejich hodnoty

1. Pomocník značek (tag helper), umožňuje pomocí C# kódu vytvořit novou HTML značku.

se přidají do objektu *data*. Následně se odešle požadavek na příslušnou URL adresu, kterým se ověřuje, zda je hodnota požadovaného vstupního pole platná. Požadavek vrací hodnotu *true*, pokud je hodnota platná, jinak vrací chybovou zprávu. V případě, že hodnota není platná, uloží se do atributu *data-val-remote* chybová zpráva; v opačném případě je hodnota tohoto atributu prázdná.

Ukázka kódu 4.4: Rozšíření kendo validátoru o remote validaci

```
$.extend(true, kendo.ui.validator, {
   rules: {
      remote: function (input) {
         if(!input.attr("data-val-remote-url") || input.val() == "") {
            return true;
         }

         var url = input.attr("data-val-remote-url");
         var container = input.closest(".k-edit-form-container");
         var data = {};
         var fields = input.attr("data-val-remote-additionalfields").
             replace(/\*./g, "").split(",");
         fields.forEach(function (field) {
            var value = container.find("input[name=" + field + "],␣
                textarea[name=" + field + "]").val();
            data[field] = value;
         });
         var valid = true;
         $.ajax({
            url: url,
            type: "GET",
            async: false,
            data: data,
            success: function (res) {
               if (res == true) {
                  input.attr("data-val-remote", "");
               } else {
                  input.attr("data-val-remote", res);
                  valid = false;
               }
            }
         });
         return valid;
      }
   },
   messages: {
      remote: function (input) {
         return input.attr("data-val-remote");
      }
   }
});
```

Oprávnění

Aplikace definuje několik typů oprávnění, která jsou implementována pomocí výčtového typu *Permissions* s atributem *Flags*, což umožňuje používat tento výčotvý typ jako bitové pole. V aplikaci jsou definována následující oprávnění:

- Zobrazit - určuje, jestli má uživatel přístup k modulu. Toto oprávnění je dále rozděleno:
  - Zobrazit moje - uživatel má přístup k modulu, ale vidí pouze ta data, která se ho týkají. Toto oprávnění je reprezentováno hodnotou *00001* (1).
  - Zobrazit vše - uživatel má přístup k modulu a vidí všechna data. Toto oprávnění je reprezentováno hodnotou *00010* (2).
- Přidat - určuje, jestli uživatel může přidávat nová data. Toto oprávnění je reprezentováno hodnotou *00100* (4).
- Upravit - určuje, jestli může uživatel upravovat existující data. Toto oprávnění je reprezentováno hodnotou *01000* (8).
- Odstranit - určuje, jestli uživatel může odstraňovat existující data. Toto oprávnění je reprezentováno hodnotou *10000* (16).

Konkrétní oprávnění je definováno kombinací (logickým součtem) výše uvedených hodnot. Například uživatel s oprávněním *00101* má oprávnění *zobrazit moje* a *přidat*.

Kontrola oprávnění

V aplikaci se každá uživatelská role řídí svými oprávněními, která určují, co uživatel vidí a jaké akce může provádět. Kontrola těchto oprávnění je nezbytná jak na úrovni pohledů, tak při volání akcí kontrolerů. K tomuto účelu byly implementovány třídy *PermissionsHelper* a *PermissionsAttribute*.

Třída *PermissionsHelper* obsahuje metodu *HasPermission*, která slouží k ověření, zda uživatel disponuje dostatečným oprávněním. Tuto třídu využívá také *PermissionsAttribute* pro kontrolu oprávnění. Kromě toho

je tato třída používána v pohledech, které se liší podle oprávnění uživatelů. Tato třídá je také používána pro kontrolu oprávnění přímo v pohledech.

Třída *PermissionsAttribute* je implementována jako atribut a používá se pro označení akcí kontrolerů, které vyžadují specifická oprávnění. Příklady použití tohoto atributu jsou uvedeny v ukázce 4.5. Pro akci *Create* je vyžadováno oprávnění pro přídávání, pro akci *Detail* je vyžadováno buď opravnění *Zobrazit moje* nebo *Zobrazit vše*, a pro akci *Index* jsou vyžadována obě oprávnění pro zobrazení. Pokud uživatel nedisponuje dostatečnými oprávněními, je mu zabráněno provést požadovanou akci a je přesměrován na pohled s omezenými oprávněními, jak je zobrazeno na obrázku 4.6.

Ukázka kódu 4.5: Příklady použití PermissionsAttribute

```
[Permissions(Permissions.Add)]
public IActionResult Create()
{
    return View();
}

[Permissions(Permissions.Overview | Permissions.View, true)]
public IActionResult Index()
{
    return View();
}

[Permissions(Permissions.Overview | Permissions.View, false)]
public IActionResult Detail(int id)
{
    var model = _manager.Detail(id);
    if (model == null) return RedirectToAction("Index");
    return View(model);
}
```

Obrázek 4.6: Nedostatečná oprávnění

5 Testování

Po dokončení implementace následovalo testování aplikace s cílem odhalit případné chyby a nedostatky. Pro účely testování a dalšího přístupu k aplikaci byla tato aplikace nasazena na server a je přístupná na adrese `https://crm-demo.pinya.cz`. Testování probíhalo s účastí několika kolegů ze společnosti Pinya s.r.o., kteří měli za úkol provést náseldující akce:

1. Přihlásit se jako automaticky vytvořený uživatel s neomezenými oprávněními.
2. V nastavení přidat libovolnou položku do číselníku, tuto položku následně upravit a poté odstranit.
3. Vytvořit nového poteciálního zákazníka (lead).
4. Přidat komunikaci s tímto leadem.
5. Po vytvoření leadu upravit jeho stav na *Kvalifikováno*.
6. Konvertovat tohoto leada na plnohodnotného zákazníka.
7. Vytvořit nového zákazníka a přidat hlavní kontaktní osobu.
8. Vytvořit nového zaměstnance se svým jménem a emailem. Nejprve vyplnit do pole čísla zaměstnance již existující hodnotu, která by neměla být povolena, a poté vyplnit nové ještě nepoužité číslo. Nastavit roli zaměstnance na *Řadový zaměstnanec*.
9. Přidat nový produkt a po přidání přejít na kartu produktu a nahrát obrázek.
10. Vytvořit novou obchodní příležitost pro jednoho z vytvořených zákazníků a jako vedoucího nastavit vytvořeného zaměstnance.
11. Přihlásit se do aplikace jako vytvořený zaměstnanec. Potřebné informace k přihlášení byly zaslány na zadaný email.
12. Změnit si heslo, odhlásit se, zkusit se přihlásit starým heslem a poté se přihlásit s novým heslem.

13. Zkusit přistoupit do správy zákazníků na adrese `https://crm-demo.pinya.cz/Customers`. Role *Řadový zaměstnanec* by k tomuto modulu neměla mít oprávnění.

14. Vytvořit novou aktivitu.

15. Vytvořit nový úkol a jako řešitele nastavit jednoho z jiných zaměstnanců a jako schvalovatele nastavit sebe. Společně s tímto řešitelem provést následující akce: předat k ověření, vrátit k dořešení, předat k ověření a buď schválit nebo zrušit.

Během testování bylo nalezeno několik chyb a problémů. Jedním z nich bylo zobrazování identifikátoru *Id* místo jména kontaktních osob v tabulkách s komunikací.

6 Závěr

Cílem této bakalářské práce bylo vytvořit funkční CRM webovou aplikaci, která má sloužit jako nově nabízený produkt společnosti Pinya s.r.o. Nejprve byly popsány CRM systémy a jejich základní funkcionalita. Poté byly analyzovány dva existující CRM systémy.

Součástí analytické fáze byly definovány funkční a nefunkční požadavky. Během návrhové fáze byla funkcionalita aplikace rozdělena do jednotlivých modulů, také byl navrhnut diagram případů užití. V této části byl také navrhnut datový model aplikace.

Dalším krokem byl výběr vhodných technologií a architektonických a návrhových vzorů pro vývoj aplikace. Poté již následovala samotná implementace aplikace. Během implementace byl brán zřetel na vykonanou analýzu a návrh. Součástí bylo rozdělení aplikace do jednotlivých projektů.

Výsledná aplikace umožňuje uživatelům se přihlásit a odhlásit. V sekci nastavení lze spravovat číselníky, šablony zpráv, uživatelské role a jejich oprávnění. Další funkcionalitou jsou správa uživatelů, aktivit, úkolů, zaměstnanců, zákazníků, potenciálních zákazníků (leadů), kontaktních osob, produktů/služeb a obchodních příležitostí. Součástí výsledné aplikace je také kalendář aktivit, evidence komunikace se zákazníky, možnost exportu některých tabulek do excelu a report o mezikvartálním a meziročním vývoji obchodních příležitostí ve formě grafů.

6.1 Možná rozšíření

Návrhy na možná budoucí rozšíření aplikace:

- Lokalizace – aplikace by mohla podporovat přepínání mezi různými jazyky, např. českým, slovenským, anglickým.

- Upozornění – aplikace by mohla poskytovat upozornění pro uživatele ohledně blížících se termínů dokončení úkolu. Taková upozornění by se dala realizovat vícero možnostmi: upozornění přímo v aplikaci, odesílání upozornění na email uživatele nebo kombinací obou předchozích možností.

- Přílohy – aplikace by mohla podporovat přidávání příloh u šablon zpráv a obchodních příležitostí.
- Logování akcí – aplikace by mohla podporovat logování proběhlých akcí, kde by se dalo vyčíst, kdo, co a kdy dělal. Tato funkcionalita by mohla pomoct při vzniku nějaké chyby, ať už na straně aplikace (výjimky), tak na straně uživatelů, tj. nechtěná změna některých údajů. Logování by se dalo implementovat více způsoby, např. jednoduché logování do souboru, nebo ukládání těchto logů do databáze a možnosti přístupu ke čtení přímo v aplikaci, např. v sekci nastavení.
- Import – aplikace by mohla podporovat import zaměstnanců, leadů, zákazníků, kontaktů i obchodních příležitostí. Pro import by mohl sloužit například formát souboru .xlsx, tedy Excel.

Bibliografie

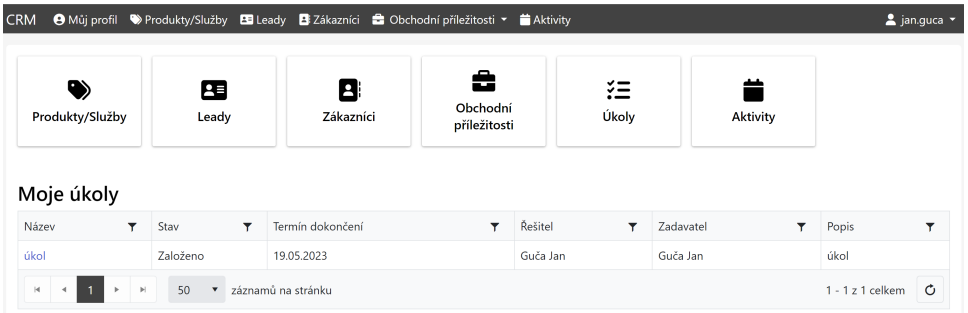
1. *CRM 101: What is CRM?* [online]. [cit. 2023-05-09]. Dostupné z: https://www.salesforce.com/crm/what-is-crm/.
2. *Salesforce CRM* [online]. [cit. 2023-04-18]. Dostupné z: https://www.salesforce.com/crm.
3. *HubSpot CRM* [online]. [cit. 2023-04-18]. Dostupné z: https://www.hubspot.com.
4. *Zoho CRM* [online]. [cit. 2023-04-18]. Dostupné z: https://www.zoho.com/crm.
5. *eBrána* [online]. [cit. 2023-04-18]. Dostupné z: https://system.ebrana.cz.
6. *Atollon* [online]. [cit. 2023-04-18]. Dostupné z: https://www.atollon.com.
7. *What is open source software?* [online]. [cit. 2023-04-21]. Dostupné z: https://opensource.com/resources/what-open-source.
8. *.NET introduction and overview* [online]. [cit. 2023-04-19]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/core/introduction.
9. *Overview of ASP.NET Core* [online]. [cit. 2023-04-19]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0.
10. *Entity Framework Core* [online]. [cit. 2023-04-19]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0.
11. *Welcome to Kendo UI for jQuery* [online]. [cit. 2023-04-21]. Dostupné z: https://docs.telerik.com/kendo-ui/introduction.
12. *Welcome to Telerik UI for ASP.NET Core Components* [online]. [cit. 2023-04-21]. Dostupné z: https://docs.telerik.com/aspnet-core/introduction.
13. *Official SCSS documentation* [online]. [cit. 2023-04-21]. Dostupné z: https://sass-lang.com/documentation/.

14. *Razor syntax reference for ASP.NET Core* [online]. [cit. 2023-04-21]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-7.0.

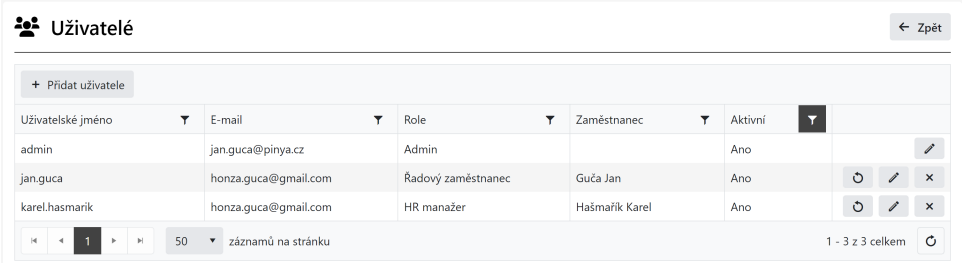
15. *5 essential patterns of software architecture* [online]. [cit. 2023-04-24]. Dostupné z: https://www.redhat.com/architect/5-essential-patterns-software-architecture.

A Ukázky uživatelského rozhraní aplikace

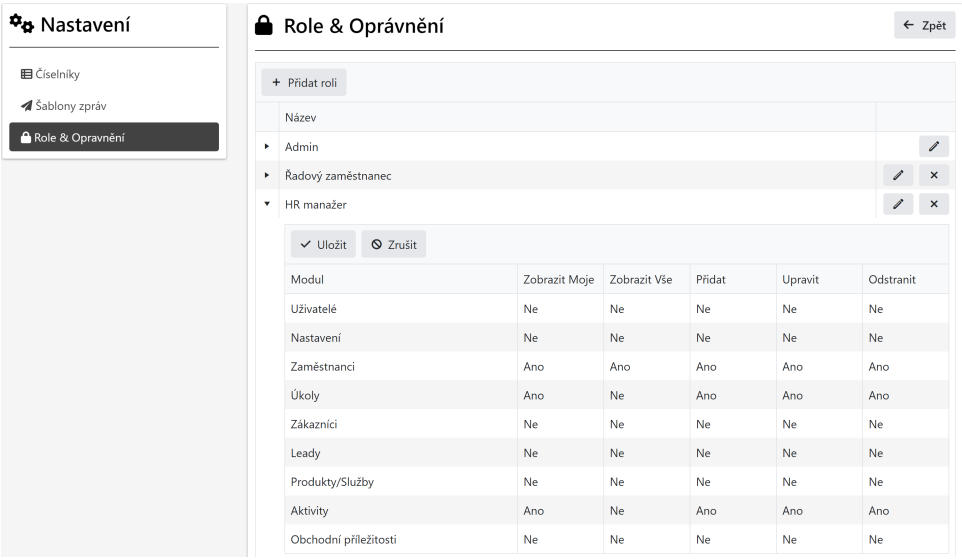
Obrázek A.1: Přihlašování



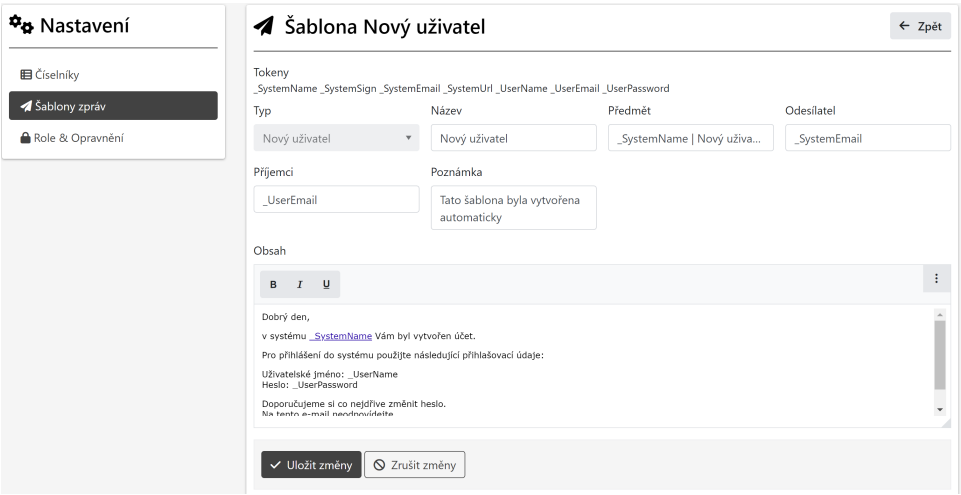
Obrázek A.2: Domovksá stránka



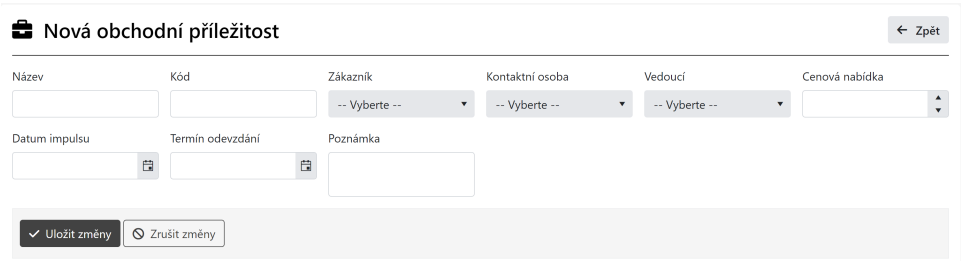
Obrázek A.3: Správa uživatelů



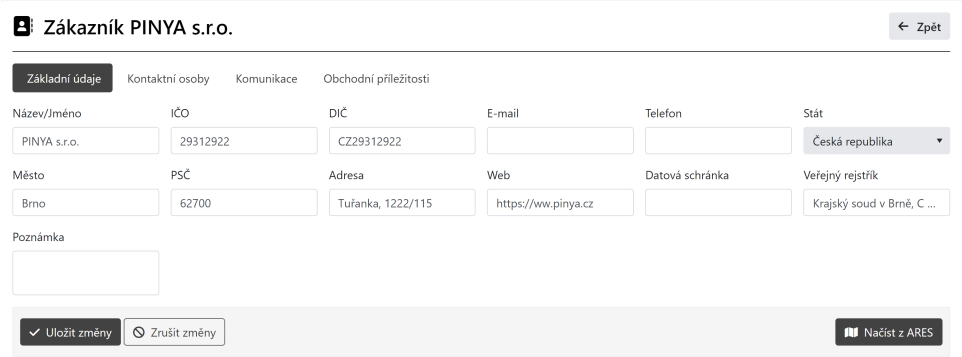
Obrázek A.4: Role a oprávnění



Obrázek A.5: Formulář pro editaci šablony zpráv



Obrázek A.6: Formulář pro vytvoření obchodní příležitosti



Obrázek A.7: Karta zákazníka

B Spuštění aplikace

Pro účely testování i pro budoucí účely je aplikace nasazena na adrese: `https://crm-demo.pinya.cz`. V případě zájmu o nahlédnutí do této aplikace na zmíněné adrese, je nutné předem kontaktovat pro vytvoření účtu.

Pro lokální spuštění aplikace je potřeba mít nainstalované následující nástroje:

- .NET 6[1]
- IDE na spuštění aplikace, například Visual Studio[2], Visual Studio Code[3] nebo Rider[4].
- Microsoft SQL Server 19[5] (použití jiné verze může vést ke komplikacím), pro lokální spuštění bude postačovat *Express* edice

Po otevření *.sln* souboru v IDE je potřeba provést následující kroky:

- V souboru *appsettings.json* v projektu *CRMBase* upravit připojovací řetězec (connection string) *"DefaultConnection"* a ujistit se, že je hodnota *"ForceDefaultDbConnection"* nastavena na *false*.
- V souboru *appsettings.json* doplnit hodnoty u *"Email"* (emailová adresa odesílající emaily) a *"Smtp"* (údaje SMTP serveru).
- Do složek *wwwroot/lib/kendo-ui/dist/js* a *wwwroot/lib/kendo-ui/dist/css* nahrát javascriptové a css soubory. Při vývoji byla použita verze *Kendo UI v2022.1.301* pro jQuery.
- V projektu *CRMBase* přidat/upravit referenci na *Kendo.Mvc.dll* (Kendo UI pro ASP.NET Core) na vašem počítači.

Poté už jen stačí aplikaci spustit. Po spuštění se zobrazí přihlašovací okno, kde je potřeba zadat údaje automaticky vytvořeného uživatele

1. `https://dotnet.microsoft.com/en-us/download/dotnet/6.0`
2. `https://visualstudio.microsoft.com/downloads/`
3. `https://code.visualstudio.com/download`
4. `https://www.jetbrains.com/rider/download/`
5. https://www.microsoft.com/en-us/sql-server/sql-server-2019-pricing

s přihlašovacím jménem *admin* a heslem *pass123*. Pro používání aplikace je také potřeba inicializovat šablony zpráv (stačí přejít do sekce *Nastavení -> Šablony zpráv*, poté se tyto šablony sami vytvoří) a naplnit hodnoty do číselníků.

C Přílohy

CRMBase.zip - zdrojový kód