UNIVERZITA PARDUBICE  
Fakulta elektrotechniky a informatiky

*Informační sytém Pojišťovny*

Seminární práce z předmětu Databázové systémy II

Pracovní tým: Jiří Král + Michal Jelínek

V Pardubicích dne 12.12.2024

Obsah

[1 Úvod 11](#_Toc184907389)

[2 Zadání 11](#_Toc184907390)

[2.1 Scénář: 14](#_Toc184907391)

[2.2 Scénář 14](#_Toc184907392)

[2.3 Procedurálná pravidla 15](#_Toc184907393)

[2.4 Uživatelské role 16](#_Toc184907394)

[2.5 CRUD analýza 16](#_Toc184907395)

[2.6 Integritní omezení 17](#_Toc184907396)

[2.7 ERD Diagram 18](#_Toc184907397)

[2.7.1 ERD Diagram BDAS1 18](#_Toc184907398)

[2.7.2 ERD Diagram BDAS2 18](#_Toc184907399)

[2.7.3 Rozdíly ERD 19](#_Toc184907400)

[2.8 Relační model dat: 19](#_Toc184907401)

[2.8.1 Relační model dat z BDAS1 19](#_Toc184907402)

[2.8.2 Relační model dat z BDAS2 20](#_Toc184907403)

[2.8.3 Rozdíly relačního modelu dat 20](#_Toc184907404)

[2.9 Fyzický model dat 21](#_Toc184907405)

[2.9.1 Adresa 21](#_Toc184907406)

[2.9.2 Banka 21](#_Toc184907407)

[2.9.3 Fotodokumentace 22](#_Toc184907408)

[2.9.4 Kategorie\_Majetku 22](#_Toc184907409)

[2.9.5 Klient 23](#_Toc184907410)

[2.9.6 Kontakt 23](#_Toc184907411)

[2.9.7 Majetkove\_pojisteni 24](#_Toc184907412)

[2.9.8 Pobocky 24](#_Toc184907413)

[2.9.9 Permise 25](#_Toc184907414)

[2.9.10 Pojistka 25](#_Toc184907415)

[2.9.11 Pojistka\_klient 26](#_Toc184907416)

[2.9.12 Povinosti 26](#_Toc184907417)

[2.9.13 Pozice 27](#_Toc184907418)

[2.9.14 Pripojisteni 27](#_Toc184907419)

[2.9.15 Secret 28](#_Toc184907420)

[2.9.16 Urazove\_pojisteni 28](#_Toc184907421)

[2.9.17 Zaměstnanec 29](#_Toc184907422)

[2.9.18 Zamestnanec\_pobocka 29](#_Toc184907423)

[2.9.19 Zamestnanec\_pojistka 30](#_Toc184907424)

[2.9.20 Zavazek 30](#_Toc184907425)

[2.10 Číselníky 30](#_Toc184907426)

[2.10.1 Tabulka – kraj 30](#_Toc184907427)

[2.10.2 Tabulka permision 31](#_Toc184907428)

[2.11 Skripty v Databázi 31](#_Toc184907429)

[2.11.1 Kurzory 31](#_Toc184907430)

[2.11.2 Procedury 32](#_Toc184907431)

[2.11.3 Funkce 34](#_Toc184907432)

[2.11.4 Trigery 36](#_Toc184907433)

[2.11.5 Pohledy 38](#_Toc184907434)

[2.11.6 Sekvence 39](#_Toc184907435)

[3 Aplikace 40](#_Toc184907436)

[4 Uživatelská dokumentace 41](#_Toc184907437)

[4.1 Základní popis používané aplikace 42](#_Toc184907438)

[4.2 Instalace 42](#_Toc184907439)

[4.3 Přístupová oprávnění 42](#_Toc184907440)

[4.3.1 Přihlašovací testovací údaje 42](#_Toc184907441)

[4.4 Použití aplikace 43](#_Toc184907442)

[4.4.1 Vstup do OASIS 44](#_Toc184907443)

[5 Závěr 52](#_Toc184907444)

# Úvod

Cílem seminární práce je vytvoření informačního systému pro pojišťovnu na platformě C# .NET6 s využitím Oracle databáze, který umožňuje správu klientů, smluv a jejich plnění. Projekt navazuje na předmět BDAS1 a dále rozvíjí znalosti o databázových systémech.

# Zadání

Zadání projektu BDAS2

Cílem projektu je prokázat, že si student osvojil prezentované teoretické znalosti a je tyto znalosti schopen aplikovat v praxi. Studenti budou vytvářet databázovou aplikaci v libovolném vyšším programovacím jazyku - C++, C#, PHP (ideálně objektový přístup), JAVA nebo je možné využít ASP.NET nebo ASP.NET MVC případně jiný framework dle domluvy se cvičícím. ORM je možné využívat pouze jako prostředek pro spouštění vlastních SQL skriptů. Jejich práce bude zahrnovat interakci se skutečným uživatelem. V rámci projektu musí projít všemi částmi návrhu databázové aplikace - analýzou, modelováním, logickým i fyzickým návrhem. Součástí projektu je tedy i vytvoření funkční aplikace, nestačí implementace požadavků pouze do databáze.

Rozsah projektu  
Cílem projektu je vytvoření databázové aplikace, ideálně webové aplikace.

*Předpokládá se znalost návrhu a tvorby aplikací studentem dle již absolvovaných předmětů. Aplikace bude využívat min. tři různé role – přičemž povinný je administrátor, registrovaný uživatel, uživatel (bez přihlášení), dále to může být zaměstnanec libovolného oddělení, vývojář, školitel, vyučující apod.*

Každá dvojice (případně jednotlivec) musí pro získání hodnocení ze zápočtového projektu:

provést stručnou prezentaci svého projektu (v rozsahu 10 - 15 minut), prezentaci demonstrujete na své vlastní aplikaci (nevytvářejte žádné powerpointové prezentace) – každý z dvojice by měl prezentovat tu část, kterou sám vytvářel;

odevzdat zprávu o projektu– odevzdává se do moodlu a postačí, aby odevzdal jeden z dvojice;

odevzdat všechny zdrojové kódy (!okomentované!), skripty a databázové skripty – postačí odevzdává jeden z dvojice;

mít implementované použité tabulky v databázovém schématu studenta, toto platí pro každého z dvojice!

Zpráva o projektu

Zpráva o projektu zahrnuje uživatelskou a programovou dokumentaci viz šablona projektu, kde naleznete i doprovodný text k jednotlivým položkám.

*Uživatelská dokumentace*  
Obsahuje informace o instalaci aplikace a jejím užití.  
*Programová dokumentace*  
Datová část:  
Popis zadavatelské dokumentace (zadání projektu), analýza, konceptuální datový model (ERD), logický datový model, datový slovník (fyzický model dat). Zkopírujte strukturální a procedurální pravidla z BDAS1. Zároveň uveďte, jak se liší vaše databáze od BDAS1, např. co jste změnili, jaká jste provedli rozšíření, apod.  
Aplikace  
Použité prostředí, instalace, popis celé aplikace, popis jednotlivých modulů. Připraveno tak, aby mohl další vývojář pokračovat ve vaší práci.

!!! Jakékoliv nedodržení zadání vede k neuznání projektu !!!

Databázová aplikace (DA) - funkční požadavky k databázové aplikaci BDAS2 (projekt navazuje na projekt z BDAS1)

Výsledná DA z BDAS2 musí splňovat MINIMÁLNĚ následující:

Orientačně je možné říci, že výsledný projekt (databáze + aplikace) bude zahrnovat následující, vše bude vytvořeno řádně v databázi a zobrazeno a využito v aplikaci!:

1. min. 10 tabulek navrženého datového modelu, včetně 2 číselníků. V dokumentaci bude uvedeno, proč byly tabulky vybrány jako číselníky;2. Každý umělý primární klíč bude mít vytvořenou sekvenci;

3. min. 3 pohledy – logicky využité, různého typu; pohledy je nutné využít pro výpis hodnot v DA;

4. min. 3 funkce různého typu s odpovídající složitostí, triviální a podobné funkce nebudou uznány, každá z funkcí musí mít odlišný výstup, tedy pracovat s různými operacemi. Do těchto funkcí nebudou počítány funkce převzaté ze cvičení;

5. min. 4 uložené procedury různého typu s odpovídající složitostí, triviální a podobné procedury nebudou uznány, každá z procedur musí mít odlišný výstup, tedy pracovat s různými operacemi, procedura může data zpracovávat i dávkově;

6. min. 2 triggery různého typu opět odpovídající složitostí, triviální a podobné spouště nebudou uznány;

7. Bude umožňovat uložit vybraný binární obsah do databáze a následně jej i z databáze získat (a pokud se bude jednat o obrázek, tak i v rámci aplikace zobrazit). Binární obsah bude možné skrz DA vložit, změnit či odstranit. Pro tento úkol vytvořte ve svém schématu speciální tabulku. Tabulku navrhněte tak, aby kromě samotného binární obsahu umožnila uložit doplňkové informace, jako např.: název souboru, typ souboru, přípona souboru, datum nahrání, datum modifikace, kdo provedl jakou operaci. Binárním obsahem může být kromě obrázku i jakýkoliv soubor např. PDF či DOCX apod.

8. Bude využívat minimálně 3 plnohodnotné formuláře (ošetření vstupních polí, apod.) pro vytváření nebo modifikaci dat v tabulkách, ostatní potřebné formuláře jsou samozřejmostí.

9. Datová vrstva aplikace bude v rámci vybraných PL/SQL bloků pracovat minimálně s jedním implicitním kurzorem a jedním explicitním kurzorem.

10. DA mohou plnohodnotně využívat pouze registrovaní uživatelé, neregistrovaný uživatel má velmi omezený výpis obsahu.

11. DA umožňuje vyhledávat a zobrazovat výsledky o všech přístupných datech v jednotlivých tabulkách dle svého oprávnění. V případě tabulky obsahující BLOB pak zobrazí název dokumentu/obrázku/jiného binárního souboru dle zvoleného tématu a návazné informace alespoň ze dvou tabulek.

12. DA umožňuje vkládání či modifikaci dat skrz uložené procedury! Tyto procedury na vkládání/modifikaci nelze započítat do nutných 4 procedur.

13. V DA nejsou viditelné ID a ani nelze vyhledávat a ani vyplňovat jakékoliv ID, aplikace je uživatelsky přívětivá. Ani pro roli administrátora není nutné ve výpisu tabulek mít ID jednotlivých záznamů.

14. Aplikace využívá triggery k logování, zasílání zpráv mezi uživateli, apod. Trigger k logování nelze započítat mezi 2 povinné triggery.

15. Grafické rozhraní DA bude funkční a bude umožňovat editovat jakýkoliv záznam, který je načtený z databáze.

16. DA řádně pracuje s transakcemi a má ošetřenou práci tak, aby nedošlo k nepořádku v datech.

17. DA využívá z datové vrstvy vlastní hierarchický dotaz, který je vhodně využit dle tématu semestrální práce.

18. DA eviduje a spravuje údaje o všech uživatelích, kteří mají do aplikace přístup.

19 Všechny číselníky se v DA chovají jako číselníky, tzn. že bude využit např. combobox, apod. Data z tabulky označená jako číselník nebude uživatel ručně zapisovat.

20. DA má implementované veškerá pravidla, omezení, apod., která byla popsána v projektu z předmětu BDAS1 a nebyla řešena na datové vrstvě.

21. DA je navržena tak, aby uchovávala historii o vkládání či úpravách jednotlivých záznamů, toto je zobrazeno pouze uživatelům s rolí Administrátor.

22. V DA existuje funkcionalita, které umožňuje nezobrazovat osobní údaje jiným uživatelům jako například rodné číslo, telefon, číslo účtu, apod. Toto neplatí pro uživatele v roli Administrátor, ty mají plný přístup všude.

23. DA umožňuje přidávat, modifikovat a mazat záznamy ve všech tabulkách dle oprávnění uživatele.

24. Aplikace bude mít menu nastaveno tak, že je možné z jedné karty přepnout na všechny ostatní, tak aby byla zaručena příjemná uživatelská správa.

25. Všechny tabulky musí být naplněny řádnými daty, nikoliv zkušebními.

26. Aplikace se skládá z hlavního okna, kde má možnost neregistrovaný uživatel procházet povolené položky menu. Hlavní okno aplikace také umožňuje přihlásit registrovaného uživatele.

27. Administrátor může spravovat jakákoliv data a zároveň se může přepnout (emulovat) na jakéhokoliv jiného uživatele.

29. Uživatel si nemůže sám zvolit při registraci svoji roli, vždy obdrží roli s nejnižšími právy a poté jej může změnit administrátor. Neregistrovaný uživatel musí vždy mít možnost registrace.

30. Databázová aplikace bude umožňovat výpis všech použitých databázových objektů  v semestrální práci (využijte systémový katalog).

## Scénář:

Středně velká pojišťovna z České republiky si přeje po mnoha letech provizorií přejít na nový informační systém. S ohledem na několika na sobě nezávislých důvodů si pojišťovna nemůže dovolit přejít na zcela nové in-house řešení, kvůli tomu vzniká toto technologické demo, aby si pojišťovna dokázala určit, které procesy lze zachovat, které odstranit a jakým stylem bude muset předělat postupy, smlouvy a jiné dokumenty. Systém proto bude v některých částech používat kombinaci nových a starých standardů zákazníka a počítat s tím, že během následujících 5 letech bude nahrazen finálním řešením, které plně nahradí jak toto technologické demo, tak hlavně dosavadní procesy v pojišťovně.

Celý proces, jakkoliv je dočasný, musí přes to podléhat nařízení o ohraně osobních údajů, neboli GDPR - (*Nařízení Evropského parlamentu a Rady (EU) č. 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES obecné nařízení o ochraně osobních údajů*)

## Scénář

Atributy jsou označeny kurzívou, **Entity** jsou označeny tučným písmem.

**Pojistka** je jedna z nejdůležitějších tabulek. **Pojistka** musí identifikovat kdy byla sjednána, do které hodnoty je kryto pojistkou a o jaký druh pojistky se jedná. A to konkrétně: **Připojištění**, **Úrazové pojištění** anebo **Majetkové pojištění**. Kromě toho je u každé pojistky evidováno, co smí a musí dělat v rámci smlouvy, díky **Závazkům** a **Pohledávkám**. Ke každé **Pojistce** také může být přidána **Fotodokumentace**. Co si také klient přál bylo ukládání dat o tom, který zaměstnanec vykonal danou pojistku. Pro další evidenci zaměstnanců slouží i tabulka **Zaměstnanec**. O **Zaměstnanci** je mimo jména nutné ukládat i co vykonávají za **Pozici**, pod kterou **Pobočku** patří. **Klient** a **Banka** ukládají svoje standartní identifikátory jako například název, oproti tomu bude **Kontakt** jako svá vlastní tabulka pouze v pozici odkazu – neboli cizího klíče. Na ni je navázaná i tabulka **Secret**, která uchovává údaje pro spojení s dvojúrovňovému ověřování a také heslo. Stejnou implementací je připojen **Kontakt** k **Zaměstnancovi**. K budoucímu spojení bude docházet mezi **Klientem** a **Bankou**. Ta momentálně není přidělená, protože jednotlivé Banky nedali integrovanému spojeni **KLIENT x POJISTOVNA x BANKA** zelenou. **Adresa** bude sloužit na uložení standartních údajů a hlavně města. To je kvůli následnému filtrování pomocí **Kraj** umožní dávat požadovaná anonymizovaná data analytickému a marketingovému týmu, či jiné časti **Zaměstnanců** pojišťovny.

## Procedurálná pravidla

* Adresa musí mít Kraj
* ID\_kraj nemůže být vyšší než 14
* Zaměstnanec musí mít Pozici
* V Adresa, Patra se počítají od 0 = přízemí, 1 = první podlaží
* Datum ukončení smlouvy nesmí být před uzavřením smlouvy
* Hodnota platba\_mesicne nesmí přesahovat 100% pojištěné hodnoty

## Uživatelské role

Databáze počítá že v rámci pojišťovny je nespočetné množství pozic, nicméně pro práci v aplikaci se počítá pouze s 5 rolemy. Jak se role přidávají jednotlivím zaměstnancům, záleží na daném postopu pojišťovny a teoreticky lze být časem modifikováno.

| **Id** | **Název role** | **Základní popis funkcí** |
| --- | --- | --- |
| 1 | Neregistrovaný uživatel (klient) | Může vyplnit základní údaje o sobě a tím si vytvořit účet v aplikaci |
| 2 | Registrovaný uživatel (klient) | Má přiděleného zaměstnance, který se o něj "stará". Může prohlížet své pojistky. |
| 3 | Zaměstnanec | Spravuje obsah pojistek. Může si přepínat mezi svými klienty a spravuje jodnotlivé pojistky. |
| 4 | Analytik | Může sledovat anonymizované statistiky a výstupy ze systému. |
| 5 | Supervisor | Má práva všech rolí (2 - 5). Přiděluje jednotlivé zaměstnance ke klientům. |
| 6 | Admin | Může modifikovat obsah všech tabulek (včetně číselníků), může sledovat logy v aplikaci a má funkce Analytika |

## CRUD analýza

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabulka** | **Neregistrovaný uživatel** | **Registrovaný uživatel** | **Zaměstnanec** | **Analytik** | **Supervisor** | **Admin** |
| **Kategorie\_Majetek** | - | - | R | R | R | CRUD |
| **KONTAKT** | C | R | CRUD | R | CRUD | CRUD |
| **Majetkove\_Pojisteni** | - | R | CRUD | R | CRUD | CRUD |
| **Permise** | - | - | - | R | CRUD | CRUD |
| **Pohledavka** | - | - | CRUD | R | CRUD | CRUD |
| **Pojistka** | - | R | CRUD | R | CRUD | CRUD |
| **Povinnosti** | - | - | CRUD | R | CRUD | CRUD |
| **Pozice** | - | - | R | R | CRUD | CRUD |
| **Pripojisteni** | - | R | CRUD | R | CRUD | CRUD |
| **Secret** | - | - | - | - | - | CRUD |
| **Urazove\_Pojisteni** | - | R | CRUD | R | CRUD | CRUD |
| **Zamestnanec\_Pobocka** | - | - | R | R | CRUD | CRUD |
| **Zamestnanec\_Pojistka** | - | - | CRUD | R | CRUD | CRUD |
| **Zavazek** | - | R | CRUD | R | CRUD | CRUD |

## Integritní omezení

IO1: Klient musí mít svoji Adresu.

IO2: Plat zaměstnance musí být minimálně ve výši minimální mzdy.

IO3: Pojištění Majetek musí spadat do nějaké kategorie

IO4: Adresa nerozlišuje městské části Prahy

IO5: Pojistka musí mít přiřazené Povinnosti a kategorii pojistky.

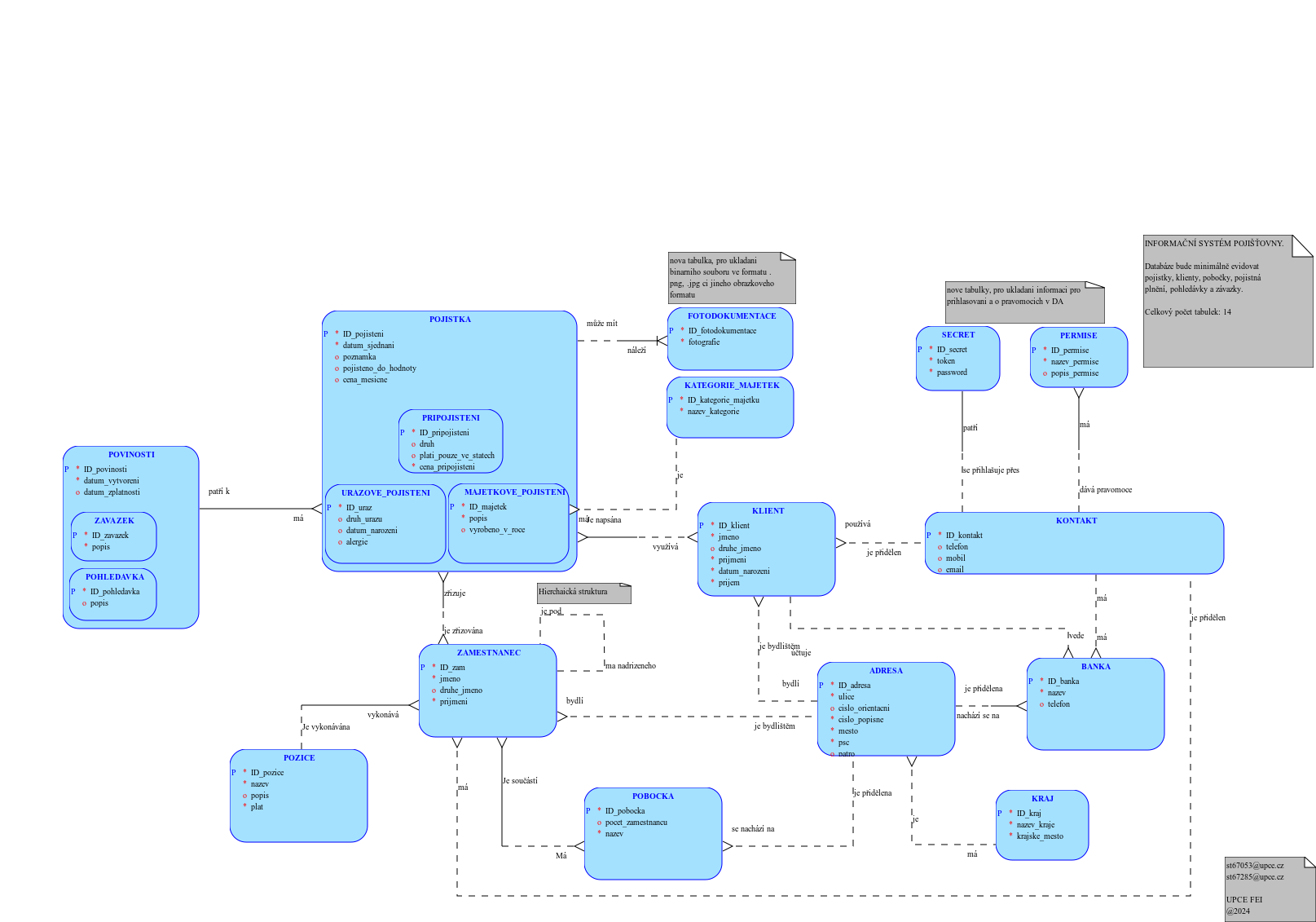
## ERD Diagram

### ERD Diagram BDAS1

Vytvořeno pomocí Oracle SQL Developer Data Modeler

Obsah obrázku text, diagram, snímek obrazovky, Plán

Popis byl vytvořen automaticky



### ERD Diagram BDAS2

Vytvořeno pomocí Oracle SQL Developer Data Modeler

### Rozdíly ERD

Oproti modelu v BDAS1, přibyli tabulky Secret - pro informace a ukládání dat nutné pro 2fa (dvojúrovňové ověřování) a tabulka Fotodokumentace, pro ukládání binárních souborů s fotografiemi k pojistkám. Zároveň přibyla i tabulka Permise, která uchovává informaci o úrovni přístupu do DA (podrobně popsáno u relačního modelu)

## Relační model dat:

Vytvořeno pomocí Oracle SQL Developer Data Modeler

### Relační model dat z BDAS1

Obsah obrázku text, diagram, Plán, Obdélník

Popis byl vytvořen automaticky

### Relační model dat z BDAS2

Obsah obrázku text, snímek obrazovky, Nalepovací papírek, Obdélník

Popis byl vytvořen automaticky

### Rozdíly relačního modelu dat

Práce z BDAS2 stojí na základech modelu z BDAS1, který byl vytvořen v roce 2022. Oproti pohledu, že práce v BDAS1 je jen teoretický scénář, jsme museli model trochu upravit a opravit, tak, aby se dal aplikovat v nějakém programu. Po analýze popisu z BDAS1 a podmínek BDAS2, jsme se rozhodli, že přihlašování bude probíhat přes emailovou adresu, která již v řešení byla a byl to nejlogičtější údaj pro přihlašování a z logiky uživatelského přístupu. Na tuto tabulku jsme pak připojili 2 nové tabulky – tabulku Secret a Permise.

Tabulka Secret obsahuje veškeré informace o heslech, tokenů pro 2fa a id, se kterým kontaktem se páruje – tak aby tato informace nebyla u Kontaktu a potencionálně hrozila jako bezpečností hrozba.

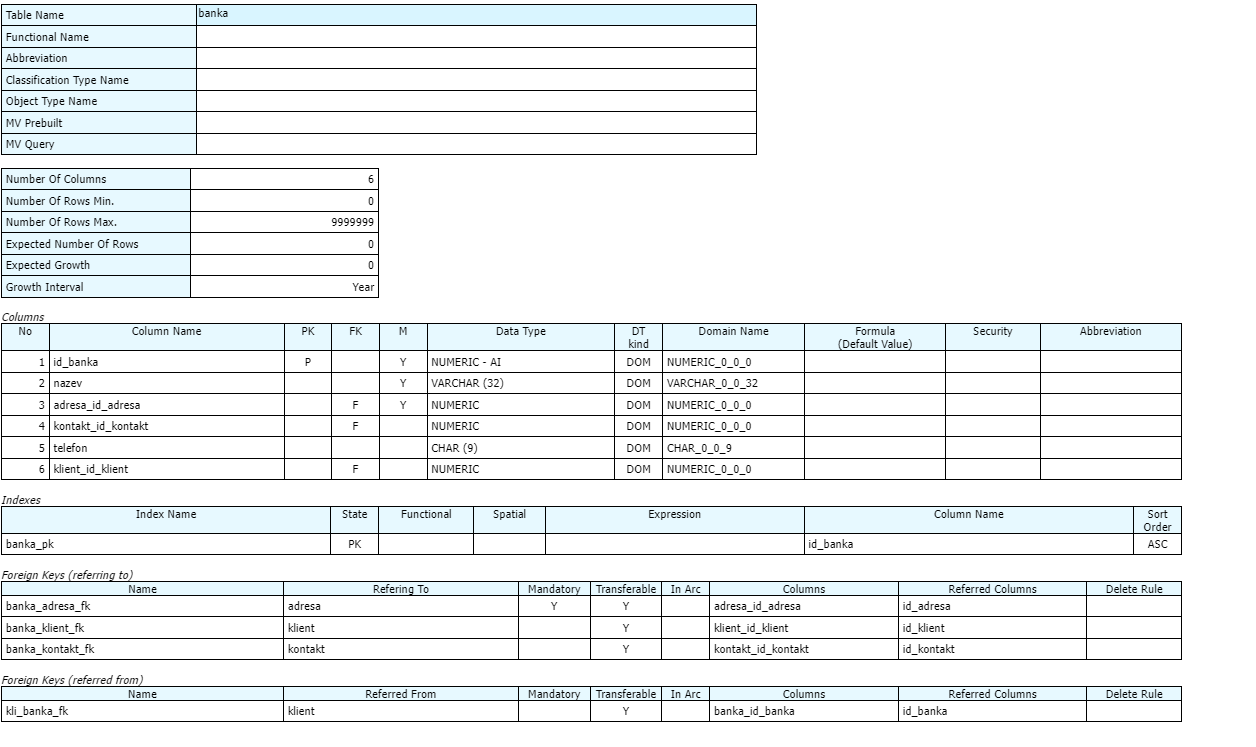
Tabulka Permise jen značí, jaké možnosti v ovládání uživatel bude mít (viz role). Další nová tabulka je navázaná na Pojistku, kdy přidává k již tak detailnímu popisu a detailu pojistek možnost k tomu všemu přiložit jakýkoliv binární soubor – ať již PDF či jakýkoliv obrázkový soubor.

Tabulka zaměstnance byla obohacena o hieratickou strukturu, která uvádí, kdo komu šéfuje a kolik podřízených má každý zaměstnanec.

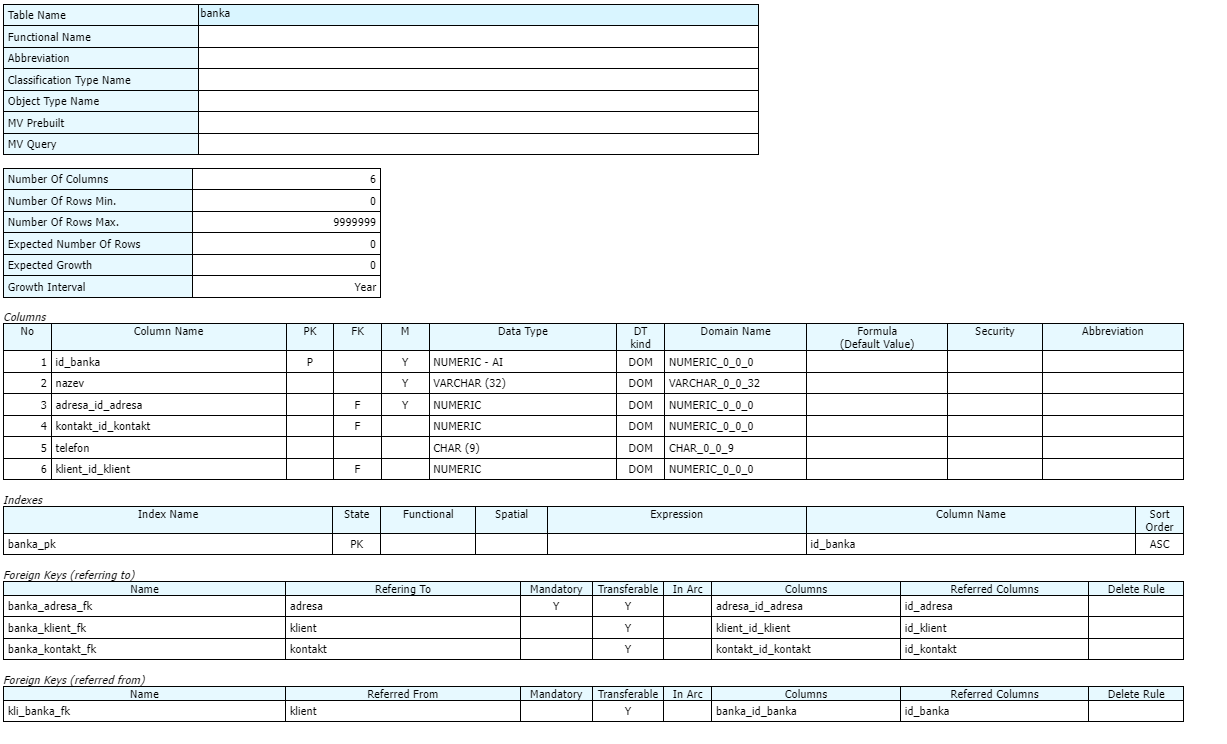
Pak již jsou v modelu jen vizuální změny pro lepší čitelnost modelu a sjednocení se stylem který je již ve fyzickém modelu, některé drobné změny proběhly i ve jménech atributů a přidání nových komentářů.

## Fyzický model dat

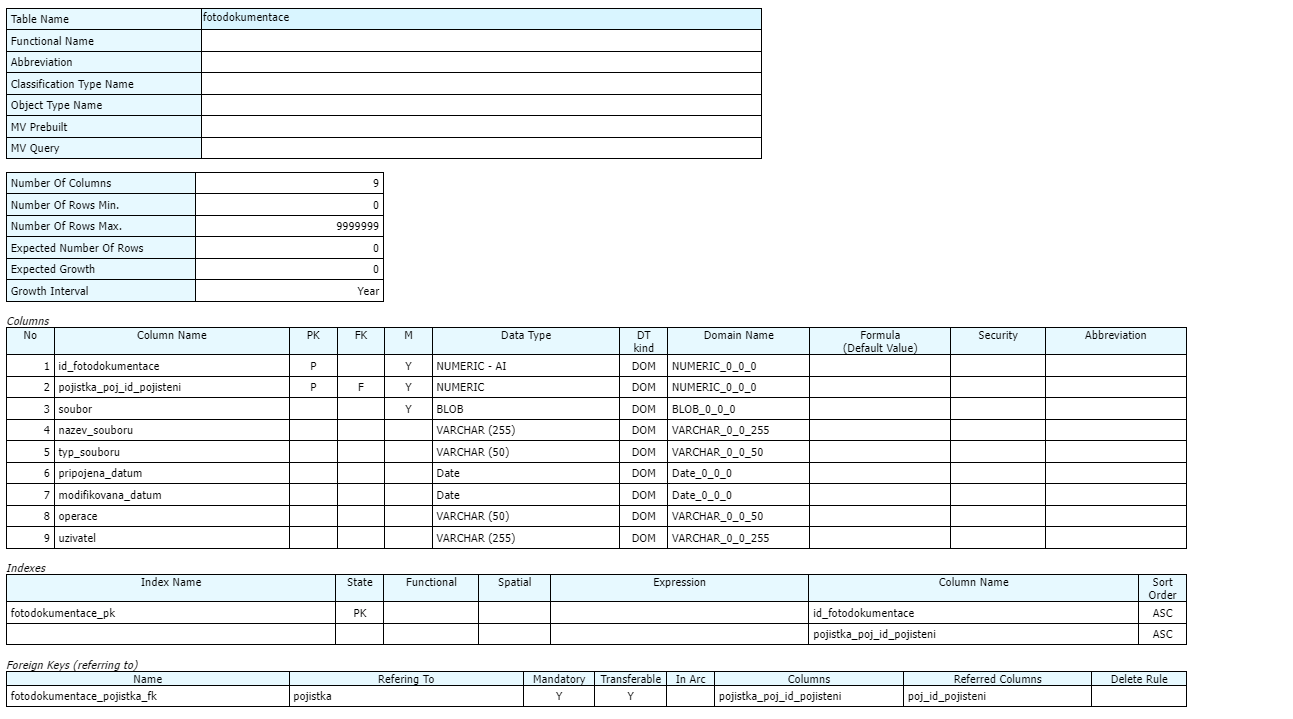
### Adresa



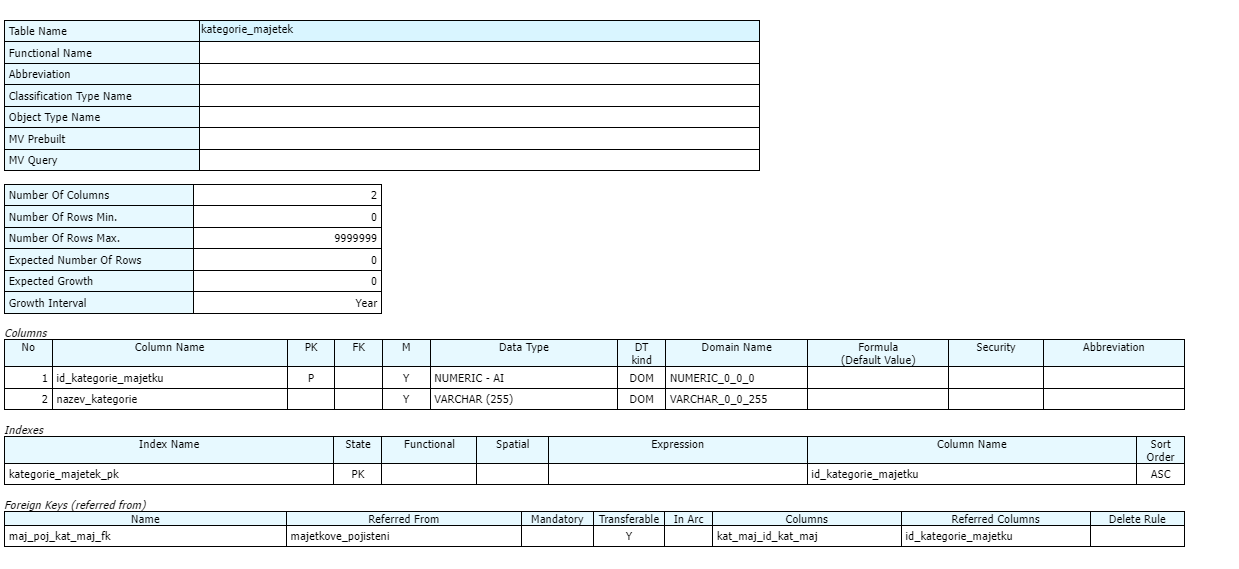
### Banka



### Fotodokumentace



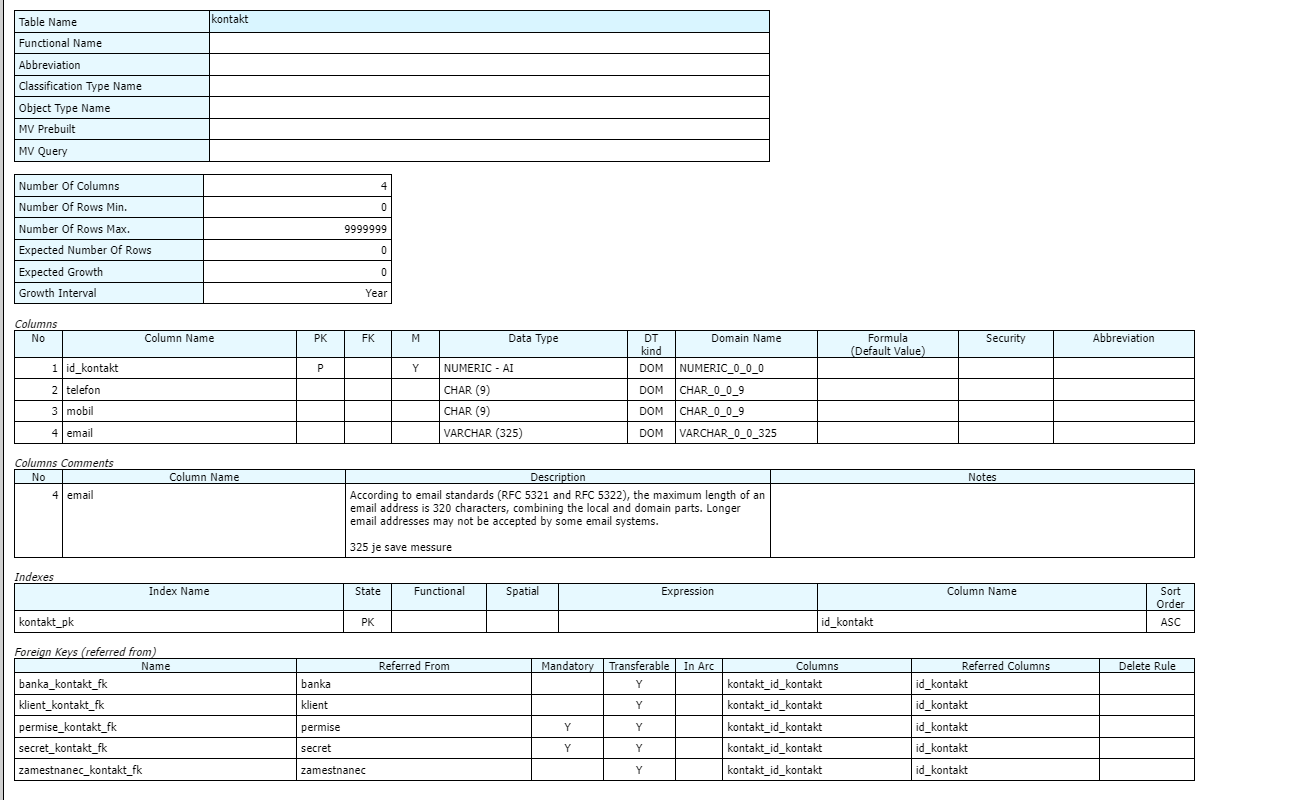
### Kategorie\_Majetku



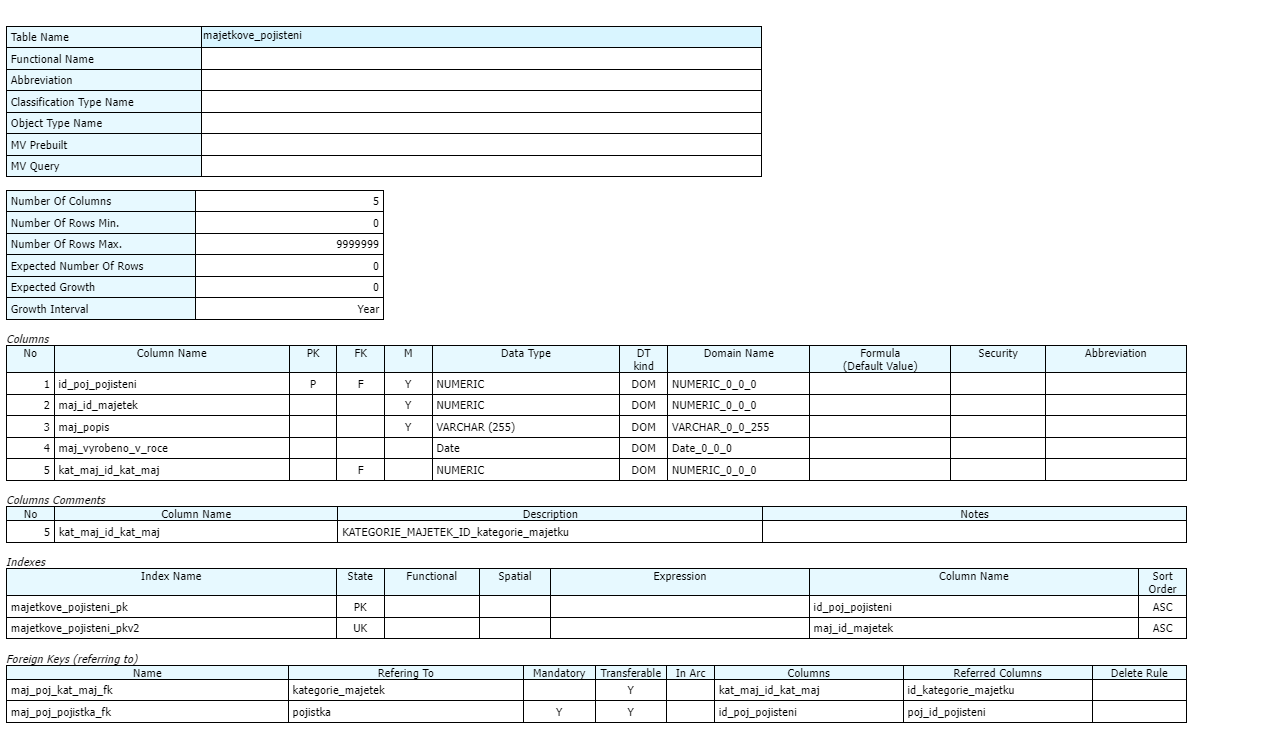
### Klient



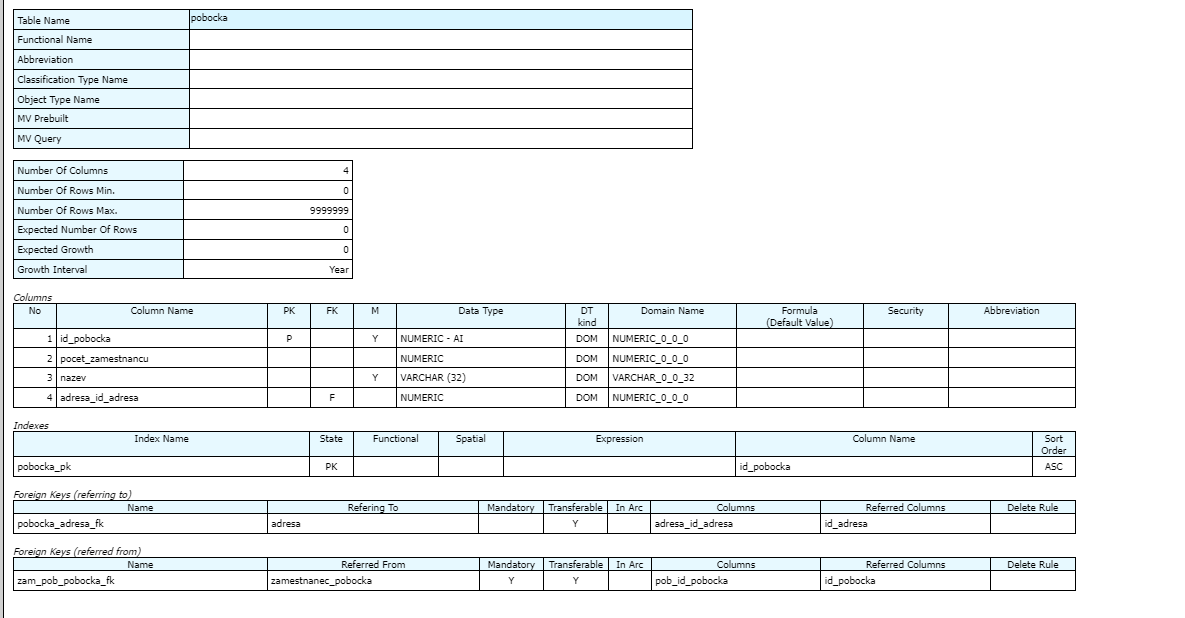
### Kontakt



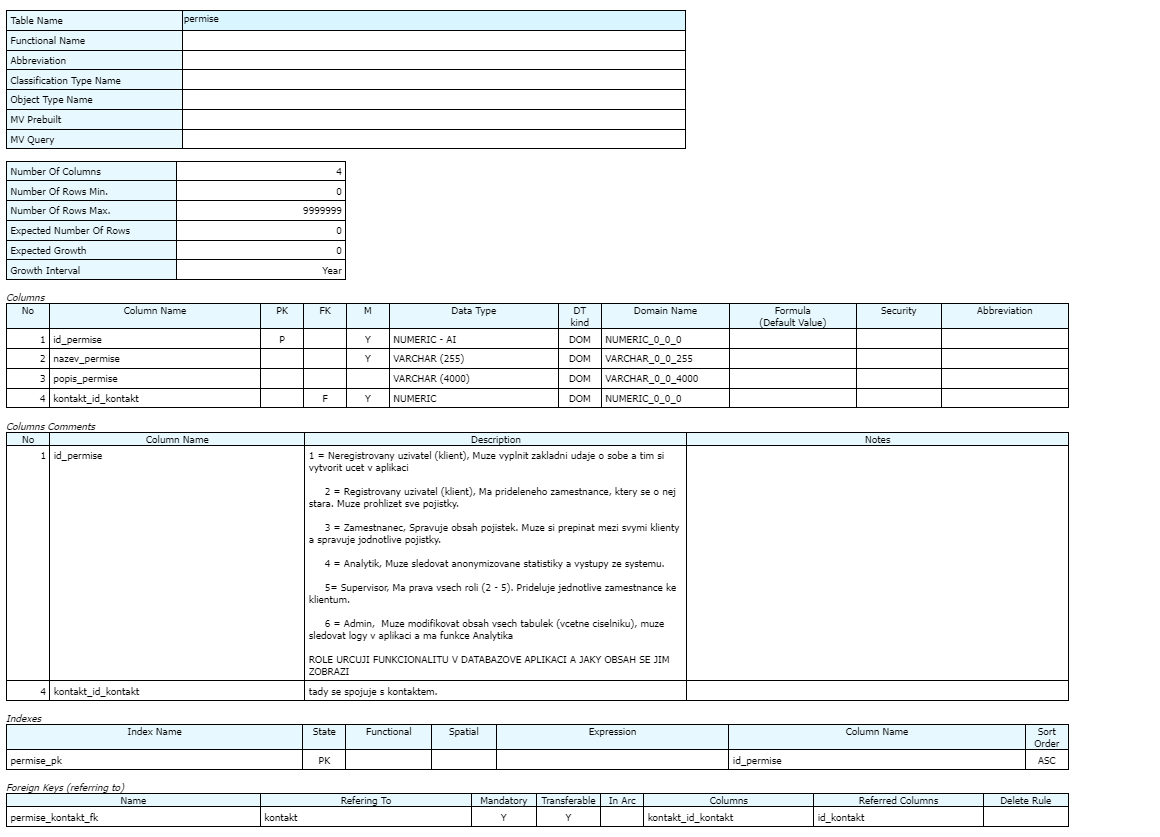
### Majetkove\_pojisteni



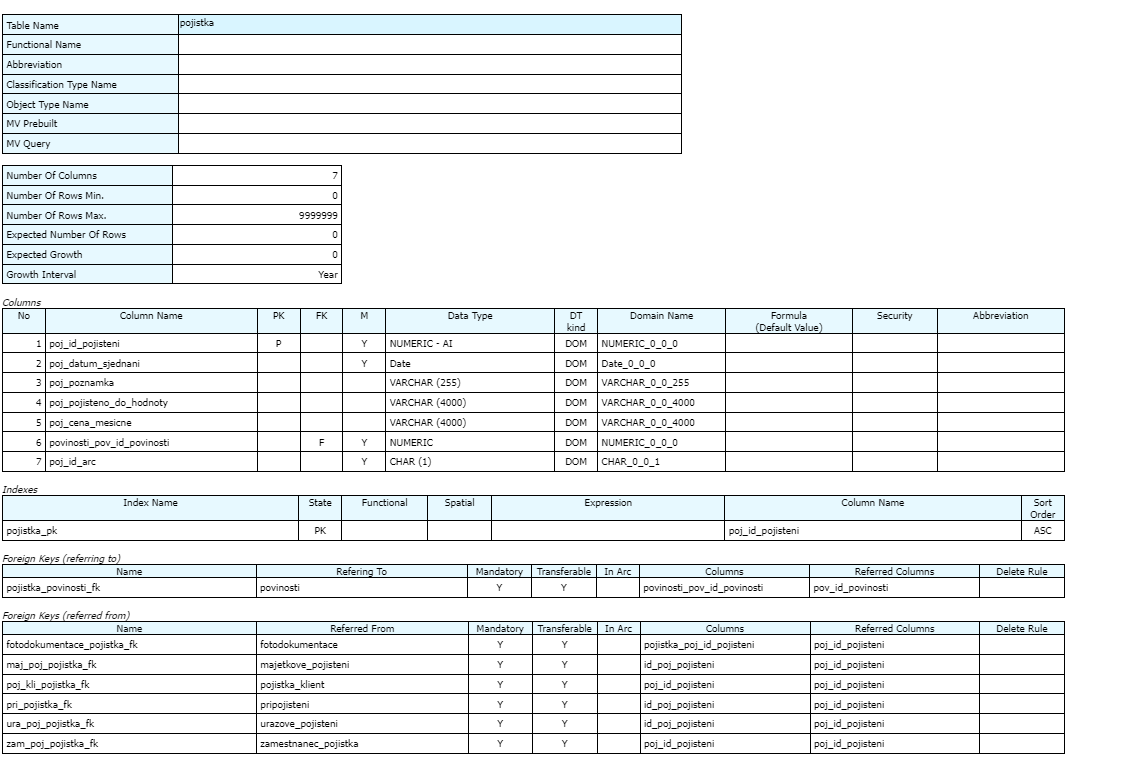
### Pobocky



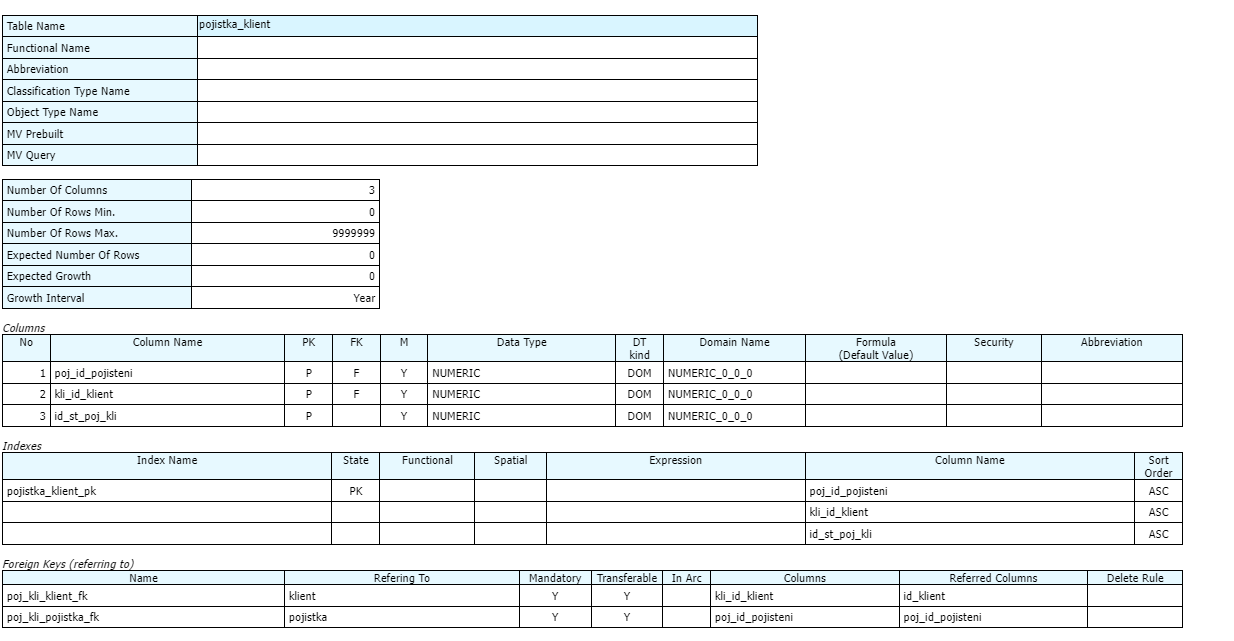
### Permise



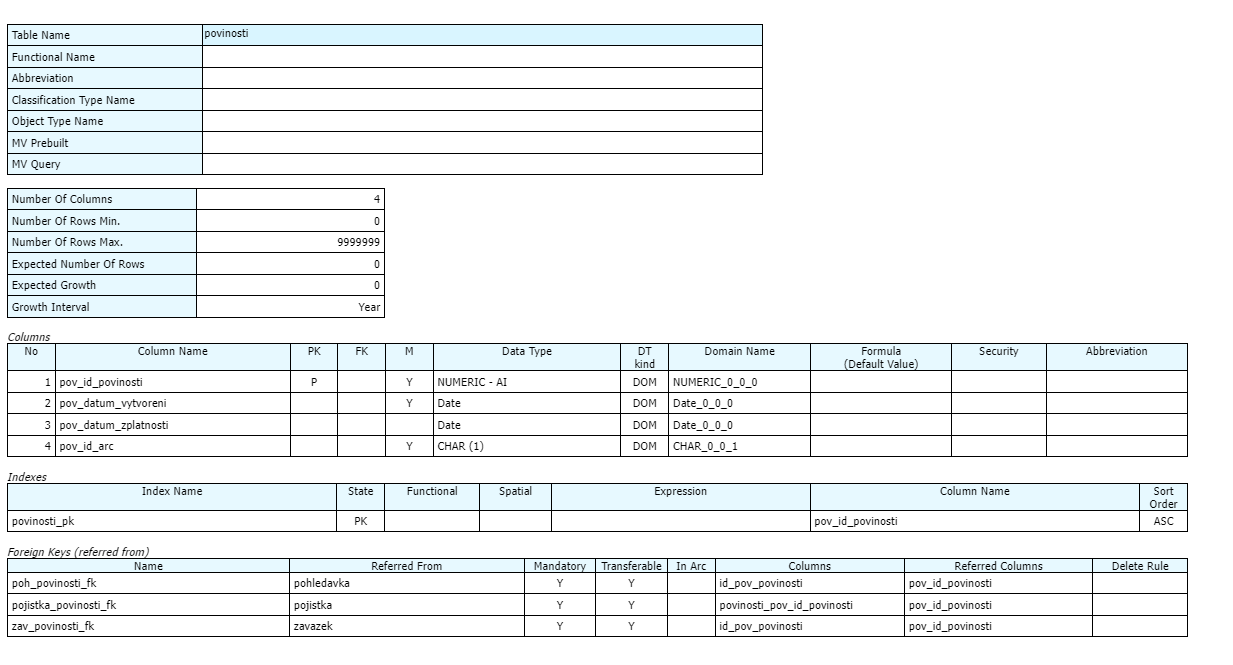
### Pojistka



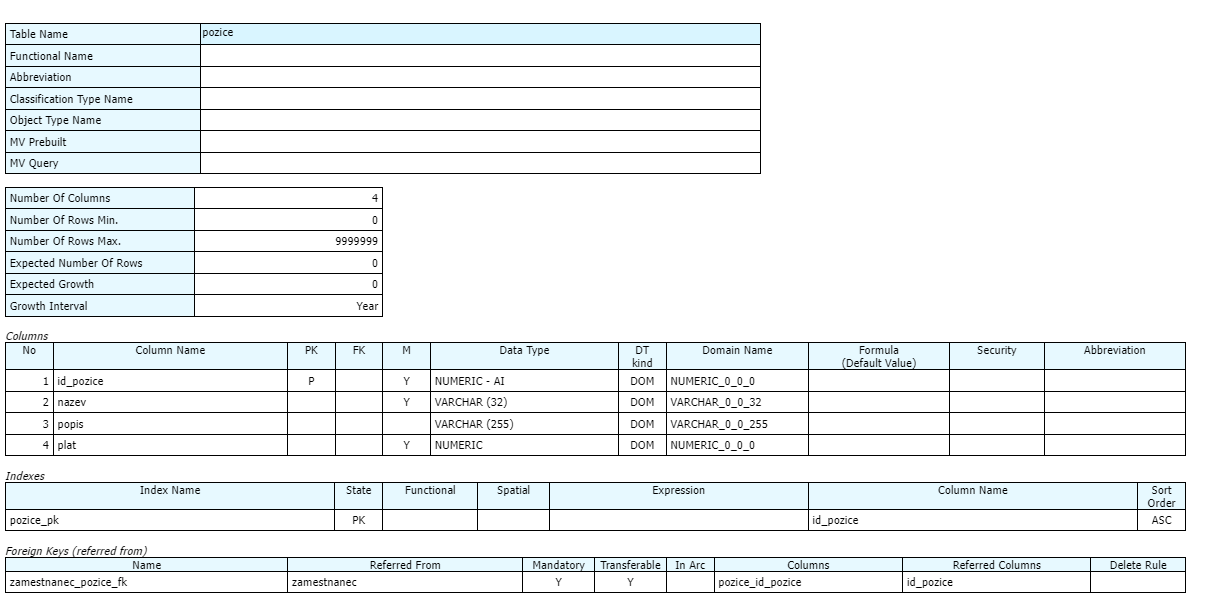
### Pojistka\_klient



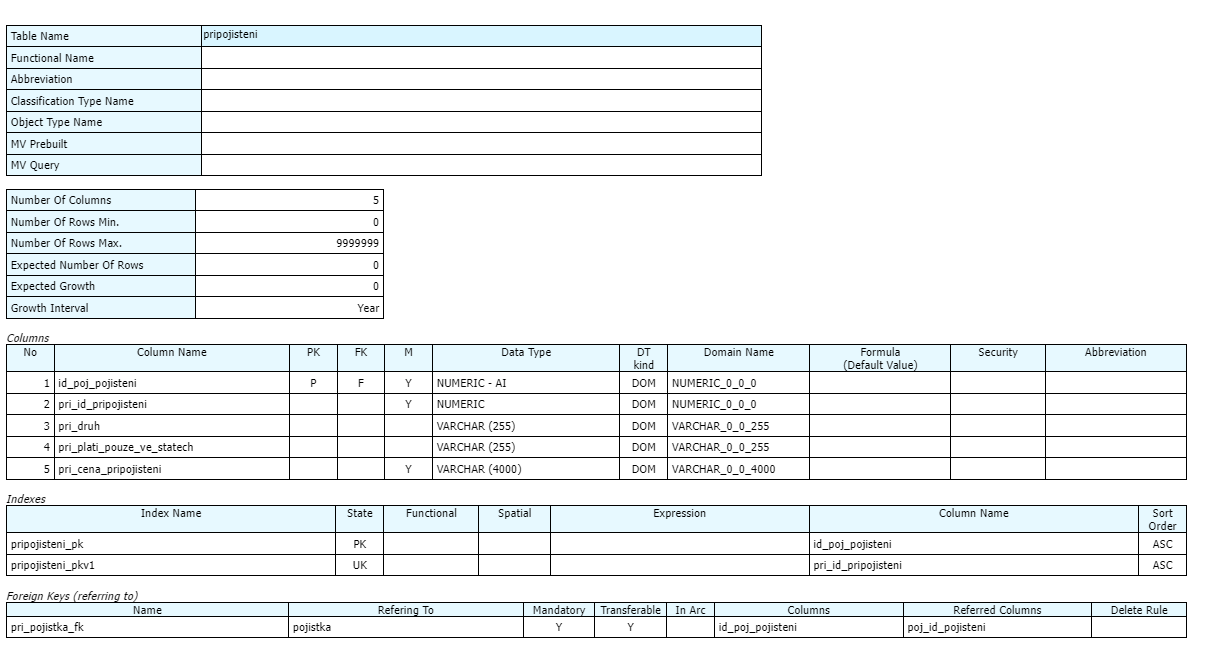
### Povinosti



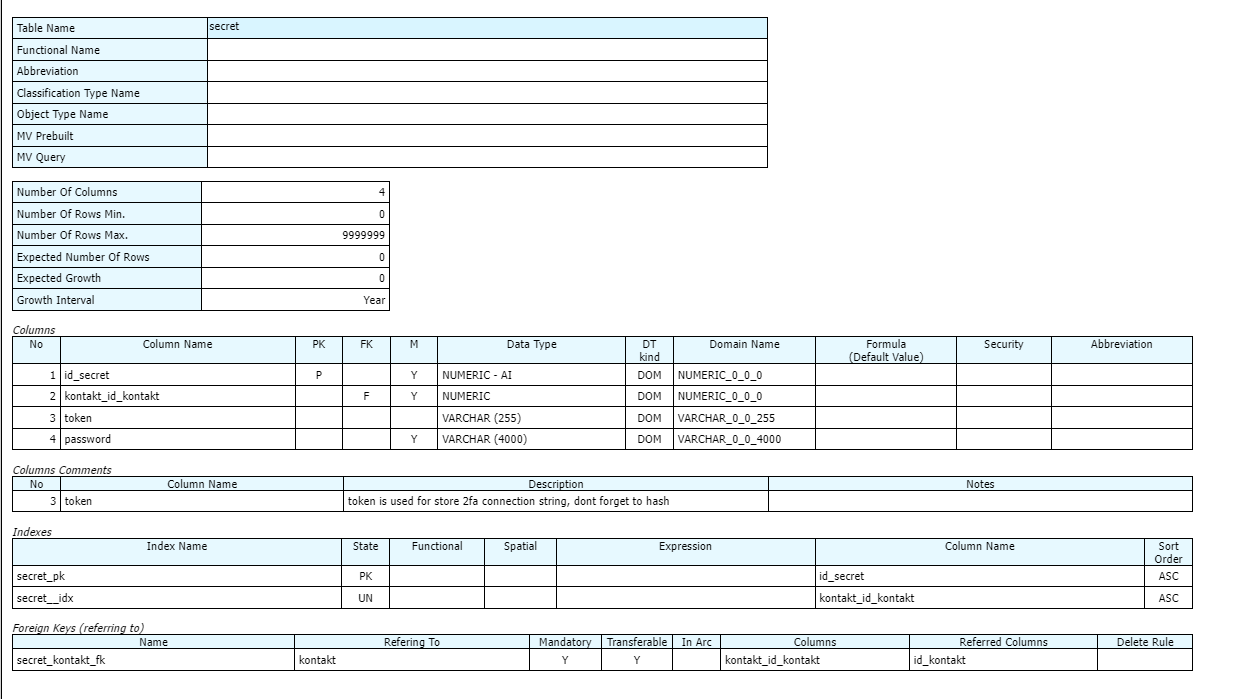
### Pozice



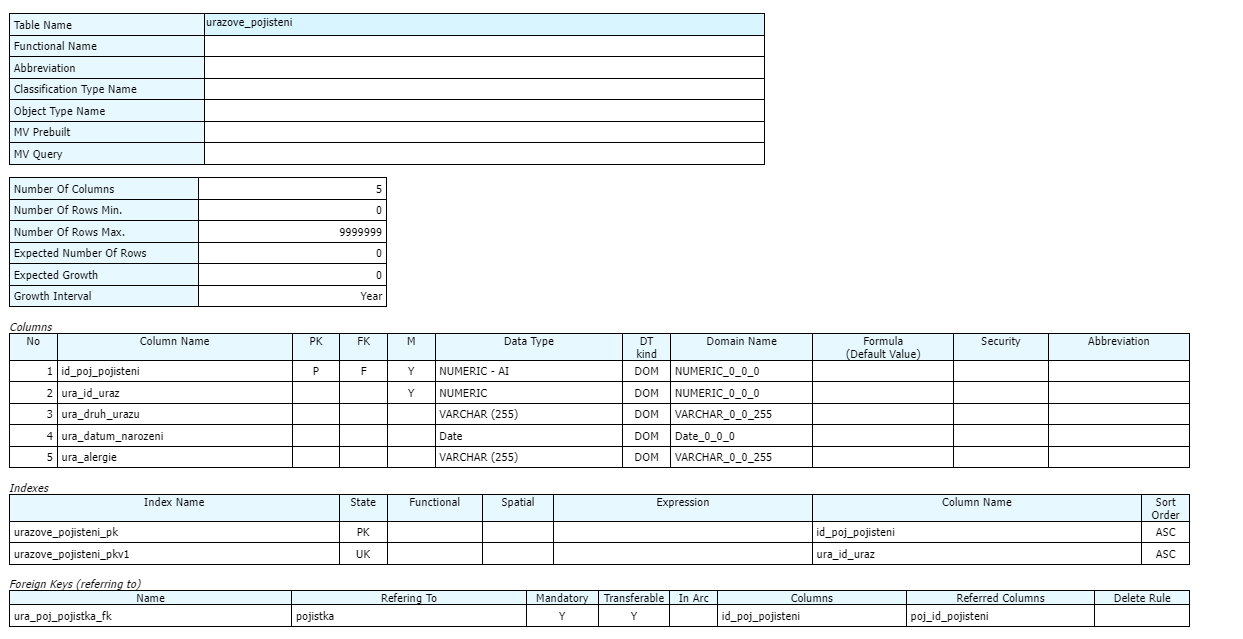
### Pripojisteni



### Secret



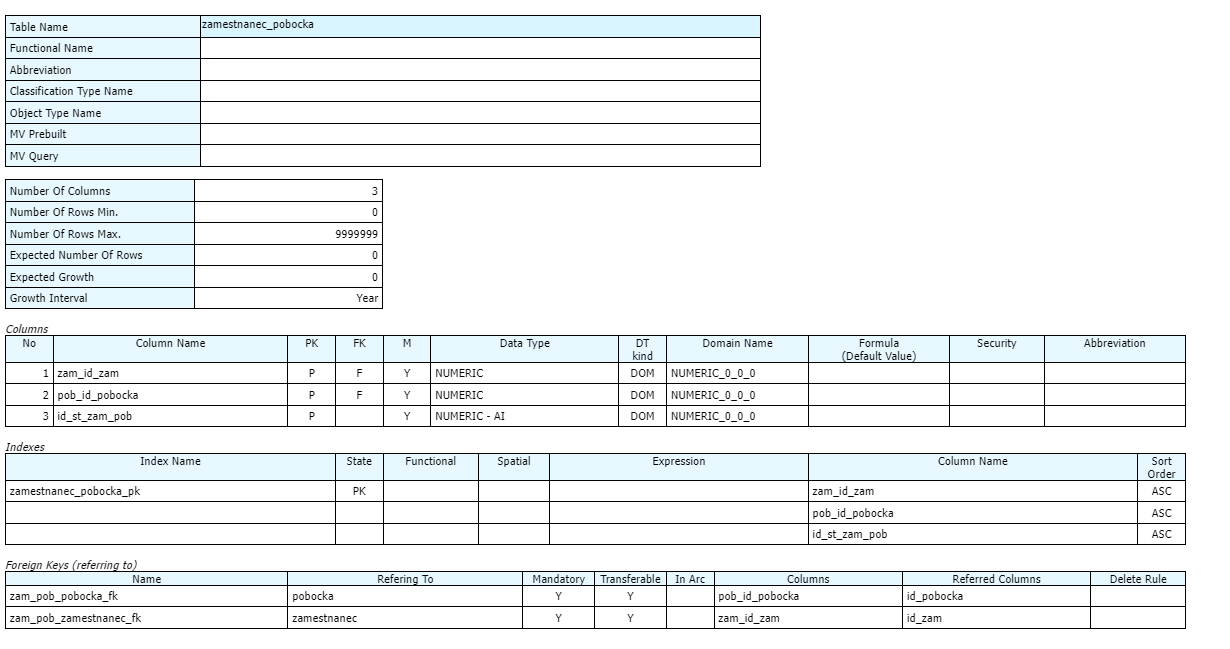
### Urazove\_pojisteni



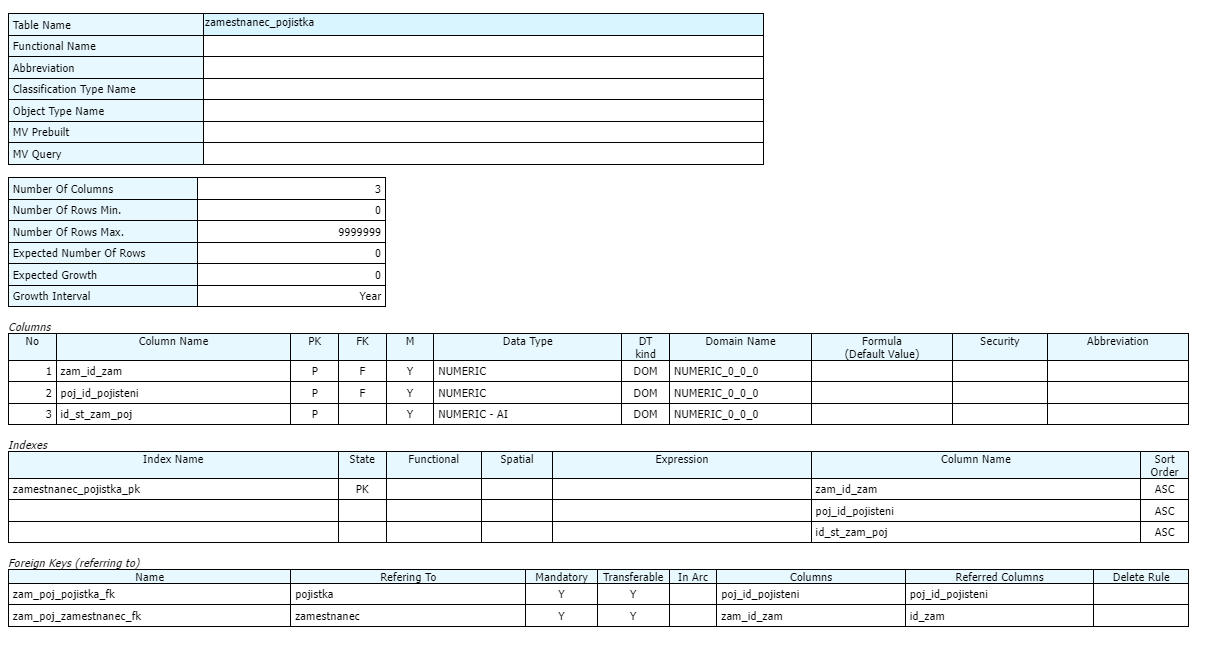
### Zaměstnanec



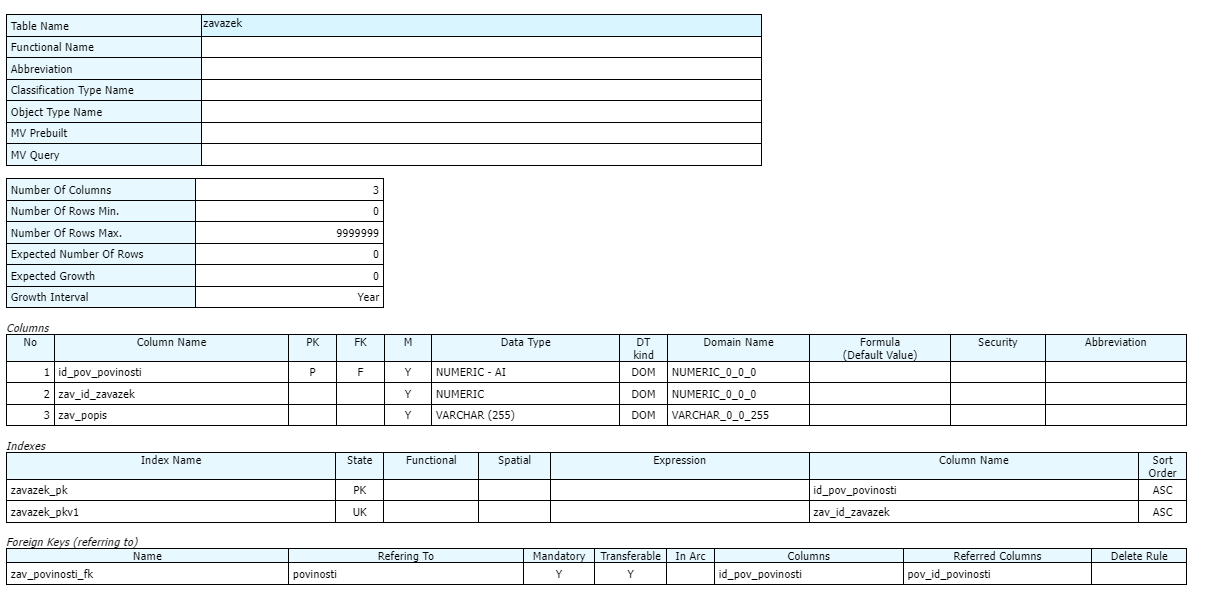
### Zamestnanec\_pobocka



### Zamestnanec\_pojistka



### Zavazek



## Číselníky

V rámci naší databázové aplikace pro systém pojišťovny využíváme číselníkové tabulky k efektivnímu řízení a správě specifických hodnot. zastávají klíčovou roli v několika důležitých aspektech našeho systému a zajišťují konzistentnost a srozumitelnost dat Při výběru dat z výčtového typu, který odpovídá číselníkové tabulce (např. Kraj nebo Permision), jsou využívána výběrová pole (ComboBox).

### Tabulka – kraj

* + Tato číselníková tabulka obsahuje všechny kraje České republiky, které určují bydliště našich klientů, celé je vedené jako číselník, protože z logiky věci se tyto údaje měnit spíše nebudou a je jen 14 možností. Pokud by se přeci jen kraje např. slučovali – Admin může hodnoty pro tento číselník změnit tak, aby na toto uživatelé nemuseli myslet

### Tabulka permision

* + Tato číselníková tabulka určuje, jakou roli obstarává přihlášený uživatel v databázové aplikaci. (např. Admin, Zaměstnanec – podrobně popsáno v kapitole 2.4)

## Skripty v Databázi

Následující kapitola bude postupně popisovat, jaké skripty a další součásti jsou použity v této práci.

### Kurzory

Kurzory jsou mechanismus v SQL a PL/SQL, který umožňuje pracovat s výsledky dotazů po řádcích. Poskytují prostředky pro iteraci přes víceřádkové výsledky, přičemž můžete jednotlivé řádky postupně číst, zpracovávat nebo měnit.

Rozdělují se na implicitní kurzory (automaticky spravované pro jednotlivé SQL operace, jako je SELECT INTO) a explicitní kurzory (uživatelem definované, poskytující větší kontrolu nad zpracováním dat).

#### c\_klienti

Tento PL/SQL blok deklaruje a používá explicitní kurzor pro iteraci nad výsledky dotazu, který kombinuje informace o klientech a jejich adresách. Každý řádek je načten do proměnné a formátovaný jako textový řetězec obsahující jméno, adresu a PSČ, který je zobrazen pomocí DBMS\_OUTPUT.PUT\_LINE. Po zpracování všech řádků je kurzor zavřen.

DECLARE

CURSOR c\_klienti IS

SELECT k.jmeno || ' ' || k.prijmeni || ' - ' || a.ulice || ' ' || a.cislo\_popisne || ', ' || a.mesto || ' ' || a.psc AS adresa

FROM klient k

JOIN adresa a ON k.adresa\_id\_adresa = a.id\_adresa;

v\_adresa VARCHAR2(4000);

BEGIN

-- Otevření explicitního kurzoru

OPEN c\_klienti;

-- Získání a zobrazení všech řádků

LOOP

FETCH c\_klienti INTO v\_adresa;

EXIT WHEN c\_klienti%NOTFOUND; -- Ukončení cyklu, pokud nejsou žádné řádky

DBMS\_OUTPUT.PUT\_LINE('Adresa klienta: ' || v\_adresa);

END LOOP;

-- Zavření kurzoru

CLOSE c\_klienti;

END;

#### ziskej\_adresu\_klienta

Tato funkce vrací kompletní adresu klienta, včetně jména, příjmení, ulice, čísla popisného a města, na základě jeho ID. Pomocí dotazu spojí informace o klientovi a jeho aktuální adrese z tabulek klient a adresa. Pokud klient neexistuje, vrátí zprávu "Klient neexistuje" nebo pokud není dostupná adresa, vrací informaci "Adresa není k dispozici".

CREATE OR REPLACE FUNCTION ziskej\_adresu\_klienta(p\_klient\_id NUMBER)

RETURN VARCHAR2

IS

v\_adresa VARCHAR2(255);

BEGIN

SELECT k.jmeno || ' ' || k.prijmeni || ', ' || a.ulice || ' ' || a.cislo\_popisne || ', ' || a.mesto

INTO v\_adresa

FROM klient k

JOIN adresa a ON k.adresa\_id\_adresa = a.id\_adresa

WHERE k.id\_klient = p\_klient\_id;

RETURN v\_adresa;

END;

/

### Procedury

Jsou uložené programy, které provádějí sadu instrukcí bez povinnosti vracet hodnotu, vhodné pro složitější logiku nebo manipulaci s daty.

#### zvys\_prijem\_davkove

Tato procedura zvýší příjmy všech klientů v tabulce klient o zadané procento. Prochází každého klienta, vypočítá nový příjem a aktualizuje odpovídající záznam. Nakonec změny uloží a vypíše potvrzující zprávu o provedené operaci.

CREATE OR REPLACE PROCEDURE zvys\_prijem\_davkove(p\_procenta IN NUMBER) IS

BEGIN

FOR klient IN (SELECT id\_klient, prijem FROM klient) LOOP

UPDATE klient

SET prijem = prijem + prijem \* (p\_procenta / 100)

WHERE id\_klient = klient.id\_klient;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Příjmy byly zvýšeny o ' || p\_procenta || '% pro všechny klienty.');

END;

/

#### ZMENIT\_KONTAKT\_KLIENTA

Tato procedura změní kontakt (email) klienta na nový, pokud tento email ještě neexistuje v databázi. Nejprve ověří, zda je email jedinečný, a pokud ne, vyvolá chybu. Pokud je email platný, provede aktualizaci záznamu v tabulce klient.

CREATE OR REPLACE PROCEDURE ZMENIT\_KONTAKT\_KLIENTA(

p\_id\_klient IN klient.id\_klient%TYPE,

p\_novy\_kontakt IN kontakt.email%TYPE

)

AS

v\_pocet NUMBER;

BEGIN

-- Zkontroluj, zda email již není použit

SELECT COUNT(\*) INTO v\_pocet

FROM kontakt

WHERE email = p\_novy\_kontakt;

IF v\_pocet > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Email již existuje v databázi.');

ELSE

-- Aktualizuj kontakt klienta

UPDATE klient

SET kontakt\_id\_kontakt = (SELECT id\_kontakt FROM kontakt WHERE email = p\_novy\_kontakt)

WHERE id\_klient = p\_id\_klient;

END IF;

END;

/

#### GENEROVAT\_STATISTIKY\_PRIJMU

Procedura vypočítává a zobrazuje průměrný příjem klientů rozdělených do tří věkových skupin: 18–30, 31–50 a 51+. Skupiny jsou určeny na základě věku klientů vypočítaného z jejich data narození. Výsledky jsou vypsány na výstup pomocí DBMS\_OUTPUT.

CREATE OR REPLACE PROCEDURE GENEROVAT\_STATISTIKY\_PRIJMU AS

BEGIN

FOR vek\_skupina IN (

SELECT CASE

WHEN FLOOR(MONTHS\_BETWEEN(SYSDATE, datum\_narozeni) / 12) BETWEEN 18 AND 30 THEN '18-30'

WHEN FLOOR(MONTHS\_BETWEEN(SYSDATE, datum\_narozeni) / 12) BETWEEN 31 AND 50 THEN '31-50'

ELSE '51+'

END AS vek\_skupina,

AVG(prijem) AS prumerny\_prijem

FROM klient

GROUP BY CASE

WHEN FLOOR(MONTHS\_BETWEEN(SYSDATE, datum\_narozeni) / 12) BETWEEN 18 AND 30 THEN '18-30'

WHEN FLOOR(MONTHS\_BETWEEN(SYSDATE, datum\_narozeni) / 12) BETWEEN 31 AND 50 THEN '31-50'

ELSE '51+'

END

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Věková skupina: ' || vek\_skupina.vek\_skupina || ' Průměrný příjem: ' || vek\_skupina.prumerny\_prijem);

END LOOP;

END;

/

#### VYHLEDAT\_KLIENTY\_V\_REGIONU

Tato procedura vyhledá všechny klienty, kteří mají adresu v zadaném kraji (regionu). Používá vstupní parametr pro ID kraje a prochází odpovídající klienty v tabulce klient. Pro každého nalezeného klienta vypíše jeho ID, jméno a příjmení.

CREATE OR REPLACE PROCEDURE VYHLEDAT\_KLIENTY\_V\_REGIONU(

p\_kraj\_id IN NUMBER

) AS

BEGIN

FOR klient IN (

SELECT k.id\_klient, k.jmeno, k.prijmeni

FROM klient k

JOIN adresa a ON k.adresa\_id\_adresa = a.id\_adresa

JOIN kraj kr ON a.kraj\_id\_kraj = kr.id\_kraj -- Připojení tabulky kraj

WHERE a.kraj\_id\_kraj = p\_kraj\_id -- Hledání podle ID kraje

) LOOP

-- Výstup klientů s odpovídajícím regionem

DBMS\_OUTPUT.PUT\_LINE('Klient ID: ' || klient.id\_klient || ' Jméno: ' || klient.jmeno || ' ' || klient.prijmeni);

END LOOP;

END;

/

### Funkce

Jsou uložené programy, které přijímají vstupy, provádějí operace a vrací výstup, často používané pro výpočty nebo zpracování dat.

#### historie\_bydliste

Tato funkce přijímá ID klienta a vrací seznam měst, kde klient v minulosti bydlel, jako jeden řetězec, oddělený čárkami. Používá funkci LISTAGG k agregaci měst záznamů z tabulky adres a spojuje je s klienty pomocí jejich ID. Pokud klient nemá žádnou historii adres, funkce vrací text "Žádná historie".

CREATE OR REPLACE FUNCTION historie\_bydliste(p\_id\_klient NUMBER)

RETURN VARCHAR2

IS

v\_historie VARCHAR2(4000);

BEGIN

SELECT LISTAGG(a.mesto, ', ') WITHIN GROUP (ORDER BY a.id\_adresa)

INTO v\_historie

FROM adresa a

JOIN klient k ON k.adresa\_id\_adresa = a.id\_adresa

WHERE k.id\_klient = p\_id\_klient;

RETURN NVL(v\_historie, 'Žádná historie');

END;

/

#### ziskej\_adresu\_klienta

Tato funkce vrací kompletní adresu klienta, včetně jména, příjmení, ulice, čísla popisného a města, na základě jeho ID. Pomocí dotazu spojí informace o klientovi a jeho aktuální adrese z tabulek klient a adresa. Pokud klient neexistuje, vrátí zprávu "Klient neexistuje" nebo pokud není dostupná adresa, vrací informaci "Adresa není k dispozici".

CREATE OR REPLACE FUNCTION ziskej\_adresu\_klienta(p\_klient\_id NUMBER)

RETURN VARCHAR2

IS

v\_adresa VARCHAR2(255);

BEGIN

SELECT k.jmeno || ' ' || k.prijmeni || ', ' || a.ulice || ' ' || a.cislo\_popisne || ', ' || a.mesto

INTO v\_adresa

FROM klient k

JOIN adresa a ON k.adresa\_id\_adresa = a.id\_adresa

WHERE k.id\_klient = p\_klient\_id;

RETURN v\_adresa;

END;

/

#### zvys\_prijem

Funkce zvyšuje příjem klienta o dané procento a vrací novou hodnotu jeho příjmu. Aktualizuje záznam příjmu v tabulce klient a výsledek ukládá do proměnné pomocí RETURNING. Pokud klient neexistuje, nebo příjem nelze zvýšit, funkce nevrací žádné další zprávy.

CREATE OR REPLACE FUNCTION zvys\_prijem(p\_klient\_id NUMBER, p\_procenta NUMBER)

RETURN NUMBER

IS

v\_novy\_prijem NUMBER;

BEGIN

UPDATE klient

SET prijem = prijem + prijem \* (p\_procenta / 100)

WHERE id\_klient = p\_klient\_id

RETURNING prijem INTO v\_novy\_prijem;

RETURN v\_novy\_prijem;

END;

/

#### prumer\_prijmu\_v\_meste

Tato funkce vypočítá a vrací průměrný příjem všech klientů v zadaném městě. Používá spojení tabulek klient a adresa k filtrování záznamů klientů podle města. Pokud v zadaném městě nejsou žádné záznamy, může vracet hodnotu NULL.

CREATE OR REPLACE FUNCTION prumer\_prijmu\_v\_meste(p\_mesto VARCHAR2)

RETURN NUMBER

IS

v\_prumer\_prijmu NUMBER;

BEGIN

SELECT AVG(prijem)

INTO v\_prumer\_prijmu

FROM klient k

JOIN adresa a ON k.adresa\_id\_adresa = a.id\_adresa

WHERE a.mesto = p\_mesto;

RETURN v\_prumer\_prijmu;

END;

#### ziskej\_adresu\_klienta

Funkce zkontroluje, zda klient existuje, a pokud ano, vrátí jeho adresu včetně ulice, čísla popisného, města a PSČ. Používá funkci NVL, aby zajistila, že pokud některá část adresy není dostupná, zobrazí se text "Neznámá". Pokud klient neexistuje, vrací zprávu "Klient neexistuje".

CREATE OR REPLACE FUNCTION ziskej\_adresu\_klienta(p\_klient\_id NUMBER)

RETURN VARCHAR2

IS

v\_adresa VARCHAR2(4000);

v\_pocet NUMBER;

BEGIN

-- Kontrola, zda klient existuje

SELECT COUNT(\*)

INTO v\_pocet

FROM klient

WHERE id\_klient = p\_klient\_id;

IF v\_pocet = 0 THEN

RETURN 'Klient neexistuje';

END IF;

-- Získání adresy klienta bez historie adres

SELECT NVL(a.ulice, 'Neznámá ulice') ' ' NVL(a.cislo\_popisne, 'Neznámé číslo') ', '

NVL(a.mesto, 'Neznámé město') ' ' NVL(a.psc, 'Neznámý PSČ')

INTO v\_adresa

FROM klient k

JOIN adresa a ON k.adresa\_id\_adresa = a.id\_adresa

WHERE k.id\_klient = p\_klient\_id;

RETURN NVL(v\_adresa, 'Adresa není k dispozici');

END;

/

### Trigery

Jsou automatické akce, které se spustí při určitých událostech v databázi (např. INSERT, UPDATE, DELETE), aby zajistily integritu dat nebo spustily další procesy.

#### zamestnanec\_pobocka\_check

Triger pro kontrolu zda není zaměstnanec přiřazen více než k jedné pobočce

CREATE OR REPLACE TRIGGER zamestnanec\_pobocka\_check BEFORE INSERT OR UPDATE ON zamestnanec\_pobocka FOR EACH ROW DECLARE l\_count NUMBER; BEGIN

SELECT COUNT(\*) INTO l\_count FROM zamestnanec\_pobocka WHERE zam\_id\_zam = :new.zam\_id\_zam;

IF l\_count > 1 THEN  
 RAISE\_APPLICATION\_ERROR(-20004, 'Zaměstnanec může být přiřazen pouze k jedné pobočce');  
END IF;

END;

#### klient\_datum\_narozeni

Triger na kontrolu klienta a jeho datumu narození, pokud datum není zadán nastaví ho na 1.1.2000

CREATE OR REPLACE TRIGGER klient\_datum\_narozeni

BEFORE INSERT ON klient FOR EACH ROW BEGIN

IF :new.datum\_narozeni IS NULL THEN :new.datum\_narozeni := TO\_DATE('2000-01-01', 'YYYY-MM-DD'); END IF; END;

#### KONTROLA\_EMAIL

Triger pro kontrolu validace a jedinečnosti emailové adresy

CREATE OR REPLACE TRIGGER KONTROLA\_EMAIL

BEFORE INSERT OR UPDATE ON kontakt

FOR EACH ROW

DECLARE v\_pocet INTEGER;

BEGIN

IF :NEW.EMAIL IS NOT NULL THEN

IF NOT REGEXP\_LIKE(:NEW.EMAIL, '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+.[A-Za-z]{2,}$') THEN RAISE\_APPLICATION\_ERROR(-20001, 'Neplatná emailová adresa.'); END IF;

SELECT COUNT(\*) INTO v\_pocet  
 FROM kontakt  
 WHERE email = :NEW.EMAIL  
 AND id\_kontakt != NVL(:NEW.ID\_KONTAKT, -1); -- Ignorovat aktuální záznam při UPDATE  
  
 IF v\_pocet > 0 THEN  
 RAISE\_APPLICATION\_ERROR(-20002, 'Emailová adresa již existuje.');  
 END IF;  
END IF;

END;

#### Stavové trigery

Aplikace rovněž obsahuje trigery pro každou tabulku když se použije INSERT, DELETE a UPDATE a rovněž když se někdo přihlásí do aplikace. Toto vše se zadává do separátní logovací tabulky.

### Pohledy

Neboli views, jsou virtuální tabulky vytvořené na základě SQL dotazů, které usnadňují práci s daty tím, že poskytují předdefinované způsoby jejich zobrazení nebo filtrování.

#### banky\_adresy

Tato procedura vytváří pohled, který kombinuje informace o bankách a jejich adresách pomocí spojení tabulek banka a adresa. Výstupem je seznam obsahující ID banky, název banky, ulici, čísla popisná a orientační, město a PSČ. Tento pohled umožňuje snadnější přístup k těmto informacím z jednoho místa.

CREATE OR REPLACE VIEW banky\_adresy AS

SELECT

b.id\_banka,

b.nazev AS banka,

a.ulice,

a.cislo\_orientacni,

a.cislo\_popisne,

a.mesto,

a.psc

FROM

banka b

JOIN adresa a ON b.adresa\_id\_adresa = a.id\_adresa;

#### pohledavky\_zavazky

Tento pohled kombinuje informace o povinnostech, pohledávkách a závazcích pomocí levého spojení mezi tabulkami povinosti, pohledavka a zavazek. Obsahuje ID povinnosti, data vytvoření a splatnosti, popisy pohledávek a závazků. Umožňuje zobrazit, které povinnosti mají odpovídající pohledávky nebo závazky, a jejich podrobnosti.

CREATE OR REPLACE VIEW pohledavky\_zavazky AS

SELECT

p.id\_pov\_povinosti AS povinnost\_id,

p.pov\_datum\_vytvoreni AS datum\_vytvoreni,

p.pov\_datum\_zplatnosti AS datum\_splatnosti,

poh.poh\_popis AS popis\_pohledavka,

zav.zav\_popis AS popis\_zavazek

FROM

povinosti p

LEFT JOIN pohledavka poh ON p.id\_pov\_povinosti = poh.id\_pov\_povinosti

LEFT JOIN zavazek zav ON p.id\_pov\_povinosti = zav.id\_pov\_povinosti;

/

#### splatne\_povinnosti

Tento pohled zobrazuje seznam povinností spolu s daty vytvoření a splatnosti, přičemž určuje, zda je povinnost již splatná. K určení splatnosti se využívá kontrola data splatnosti oproti aktuálnímu datu (SYSDATE), a výsledek je označen jako Ano nebo Ne. Výstupem jsou pouze povinnosti, které mají datum splatnosti dřívější než aktuální datum.

CREATE OR REPLACE VIEW splatne\_povinnosti AS

SELECT

id\_pov\_povinosti AS povinnost\_id,

pov\_datum\_vytvoreni AS datum\_vytvoreni,

pov\_datum\_zplatnosti AS datum\_splatnosti,

CASE

WHEN pov\_datum\_zplatnosti < SYSDATE THEN 'Ano'

ELSE 'Ne'

END AS je\_splatne

FROM

povinosti

WHERE

pov\_datum\_zplatnosti < SYSDATE;

/

#### zamestnanci\_pobocky

Tento pohled zobrazuje informace o zaměstnancích, jejich pozicích a pobočkách, kde pracují, spojením tabulek zamestnanec, pozice, zamestnanec\_pobocka a pobocka. Obsahuje ID zaměstnance, jeho jméno, příjmení, název pozice, název pobočky a počet zaměstnanců v pobočce. Pohled poskytuje přehled o organizační struktuře a počtu zaměstnanců v jednotlivých pobočkách.

CREATE OR REPLACE VIEW zamestnanci\_pobocky AS

SELECT

z.id\_zam,

z.jmeno,

z.prijmeni,

po.nazev AS pozice,

pb.nazev AS pobocka,

pb.pocet\_zamestnancu

FROM

zamestnanec z

JOIN pozice po ON z.pozice\_id\_pozice = po.id\_pozice

JOIN zamestnanec\_pobocka zp ON z.id\_zam = zp.zam\_id\_zam

JOIN pobocka pb ON zp.pob\_id\_pobocka = pb.id\_pobocka;

### Sekvence

Každá tabulká má svoji sekvenci pro id v taulce, kde pak fungují autoincrementy tak, aby nemuselo být ručně vkládáno id.

CREATE SEQUENCE "ST67053"."ZAMESTNANEC\_POBOCKA\_ID\_ST\_ZAM\_" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE NOKEEP NOSCALE GLOBAL ;

Pokud by již v databázi id byli, tak takto se dá manuálně zvýšit poslední id

ALTER SEQUENCE pojistka\_seq INCREMENT BY 1; -- Nastavíme dočasně přírůstek

ALTER SEQUENCE pojistka\_seq INCREMENT BY 1 START WITH 1001; -- Příští hodnota bude správná

SELECT pojistka\_seq.NEXTVAL FROM dual; -- Posuneme sekvenci ALTER

SEQUENCE pojistka\_seq INCREMENT BY 1; -- Vrátíme původní přírůstek

# Aplikace

Aplikace vychází ze zadání z celého projektu. Aplikace stojí na programovací jazyku C# a platformě .net 8, formuláře jsou postupně popsány v uživatelské dokumentaci. Popis kódu není součástí tohoto dokumentu, kód je separátně zdokumentován komentáři přímo v kódu.

Aplikace byla připravena na škálování a postupné přidávání rolí a jiných funkcí.

# Uživatelská dokumentace

## Základní popis používané aplikace

Aplikace OASIS je desktopová aplikace pro Windows (verze 8, 8.1, 10, 11), která vám umožní připojit se do systému pojišťovny.

## Instalace

Zkontrolujte, že jste na poslední verzi vašeho operačního systému

Nainstalujte si VPN klienta Cisco AnyConnect (podrobný návod zde <https://www.upce.cz/vpn-na-upa>) a připojte se na ni.

Stáhněte si klienta OASIS, buďto na stránkách pojišťovny, nebo na repositáři GitHub, kde je poslední veřejná verze: <https://github.com/Jitralar/databaze2-pojistovna2024>

## Přístupová oprávnění

Každý uživatel může mít Permisi s hodnotou 1 až 6 (podrobně popsáno v kapitole 1.4) při vytvoření účtu vám bude přidělena permise (nebo také role) 2. Poté už váš supervisor kontaktuje admina, a ten vám změní přístup, dle vaší pravomoci.

### Přihlašovací testovací údaje

|  |
| --- |
| Nezaregistrovaný uživatel: |
| Prohlížení menu |
| Login: lze se přihlásit bez přihlašovacích údajů |

|  |
| --- |
| Klient |
| Prohlížení menu, pojistky |
| Login: uzivatel@gmail.com, Heslo: abcde |

|  |
| --- |
| Zaměstnanec |
| Prohlížení menu, tabulka klientů, tabulka pojistek |
| Login: zamestnanec@gmail.com, Heslo: abcde |

|  |
| --- |
| Analytik |
| Prohlížení menu, statistiky - klientů, zaměstnanců, pojistek, poboček |
| Login: analytik@gmail.com, Heslo: abcde |

|  |
| --- |
| Supervizor |
| Prohlížení menu, statisky - klientů, zaměstnanců, pojistek, poboček, tabulka zaměstnanců |
| Login: supervizor@gmail.com Heslo: abcde |
| Admin |
| Prohlížení menu, emulace, upravit obsah, tabulek, zobrazit upravy a díky emulaci má všechny možnosti, co role předtím |
| Login: admin@gmail.com, Heslo: abcde |

Každý uživatel se přihlašuje pomocí svého uživatelského jména do systému (podle rozhodnutí pojišťovny to je aktuálně Email) a hesla, které se ukládá v systému (podrobně popsáno v 3.4.1 – Přihlášení do OASIS). Po přihlášení je defaultně přidělena role Klient (role 2) a přenastavit ji následně může Admin v kolonce „Upravit obsah tabulky“.

## Použití aplikace

Obsah obrázku text, snímek obrazovky, software, Webová stránka

Popis byl vytvořen automaticky

Aplikace je tvořena do dvou logických ovládání. Takzvané boční menu a hlavní obrazovka

Zde bude vaše ovládání s ohledem na vaši roli. Klient má pár tlačítek a možností. Oproti tomu Admin najde plejádu administrativních, analytických a jiných nástrojů, které mu umožní využívat svoji roli na maximum. Nabídka funkcí a možností je vázané na vaši roli.

### Vstup do OASIS

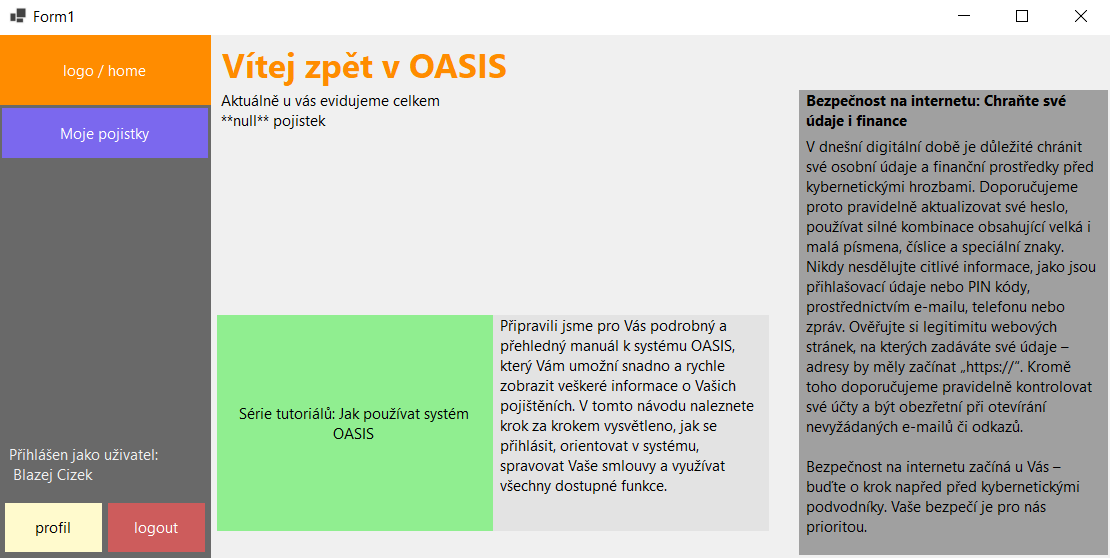
Po spuštění aplikace máte možnost se zaregistrovat, když nemáte učet, když už svůj učet máte můžete se přihlásit. Přihlašování do aplikace probíhá pomocí emailové adresy uživatele a hesla. V budoucnu se plánuje i integrace dvouúrovňového ověření, na které je již systém připraven.

Obsah obrázku text, snímek obrazovky, software, Písmo

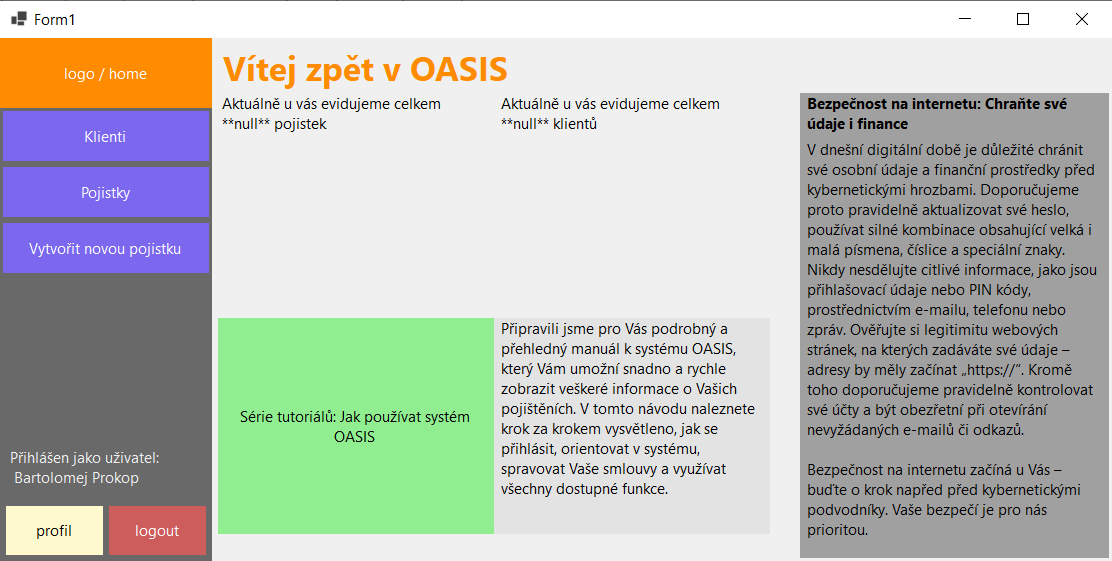
Popis byl vytvořen automaticky

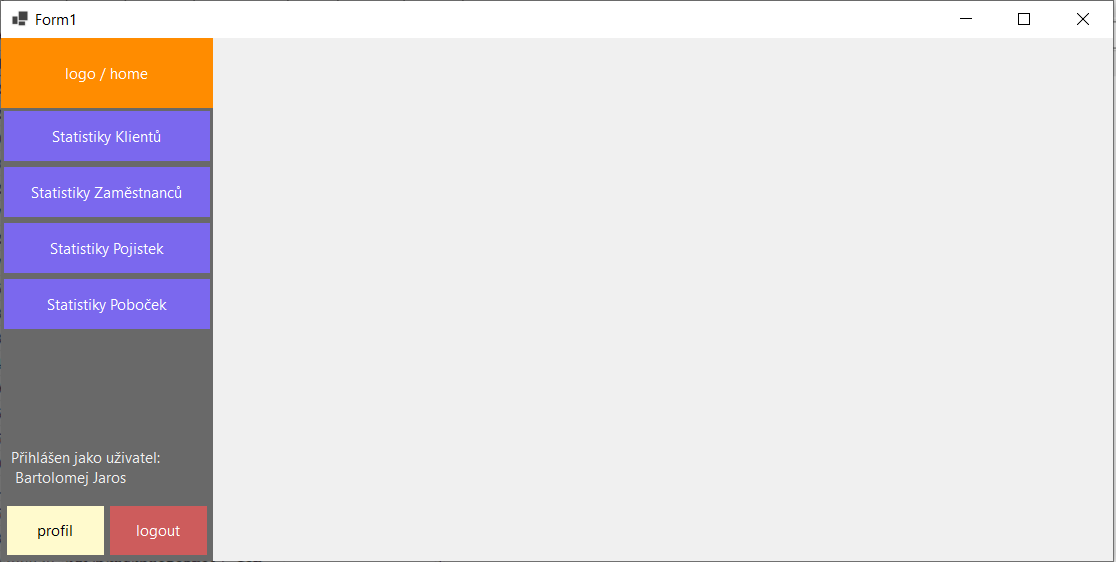
Okno, pokud uživatel bude pokračovat bez přihlášení (oprávnění „nezaregistrovaný uživatel“), kde má možnost se registrovat, registrovaný uživatel má možnost se přihlásit. Veškerá data jsou po přihlášení ihned uložena do databáze, tak aby jste mohli používat OASIS na plno.

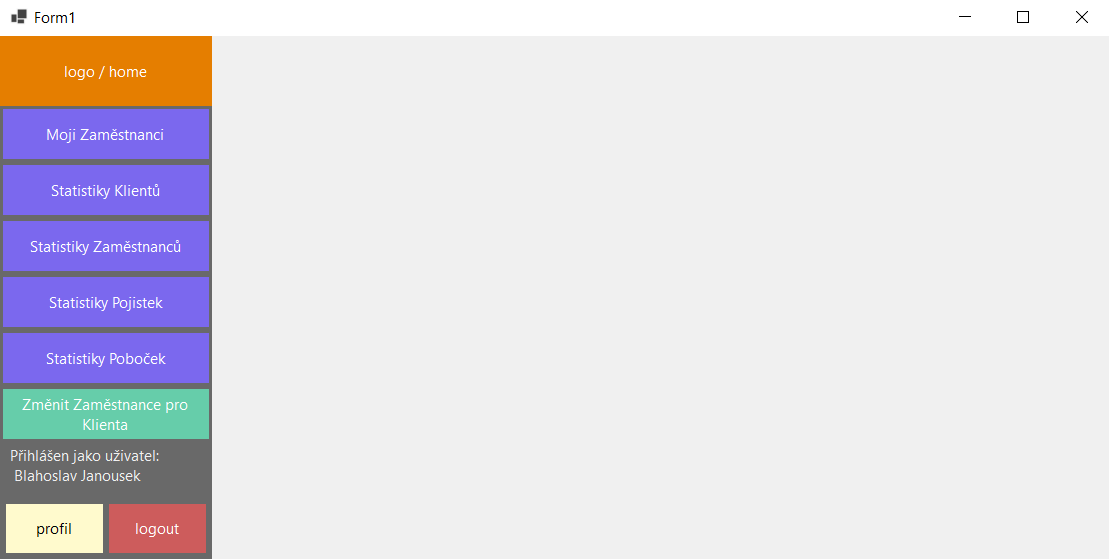
Pokud se vám nejde připojit, či máte problém s propojením k databázi. Doporučujeme vypnout a zapnout wi-fi na vašem zařízení, či ho restartovat. Z testovací zkušenosti víme že to vyřeší 99% problémů s připojením. Další častý problém je nepřipojení k VPN.

Okno po úspěšném přihlášení uživatele (oprávnění „Klient“). V levé části aplikace se nachází navigační menu, kde uživatel může přepínat okna. Klient si může zobrazit Domovskou stránku, své pojistky, profil.

Zároveň připravujeme i „infobox“ kde postupně se mohou zobrazovat tipy pro používání, tipy k bezpečnosti na internetu či informace o výpadkách či plánovaných odstávkách systému a tlačítko, které lze napojit na tutoriály, zaměstnanecký intranet či k manuálu pro sytém.

Okno po úspěšném přihlášení uživatele (oprávnění „Zaměstnanec“). V levé části aplikace se nachází navigační menu, kde uživatel může přepínat okna. Zaměstnanec si může zobrazit Domovskou stránku, Profil, Klienty, Pojistky, nebo může vytvořit novou pojistku

Okno po úspěšném přihlášení uživatele (oprávnění „Analytik“). V levé části aplikace se nachází navigační menu, kde uživatel může přepínat okna. Analytik si může zobrazit Domovskou stránku, Profil, Statistiky - Klientů, Zaměstnanců, Pojistek a Poboček

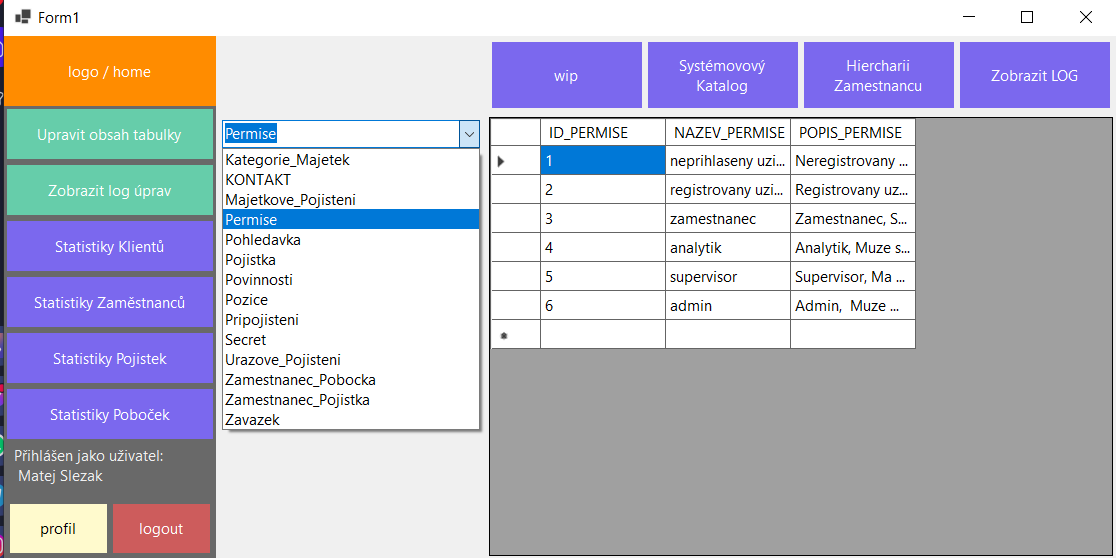


Okno po úspěšném přihlášení uživatele (oprávnění „Supervizor“). V levé části aplikace se nachází navigační menu, kde uživatel může přepínat okna. Supervizor si může zobrazit Domovskou stránku, Profil, Statistiky - Klientů, Zaměstnanců, Pojistek a Poboček, Změnu zaměstnance u klienta

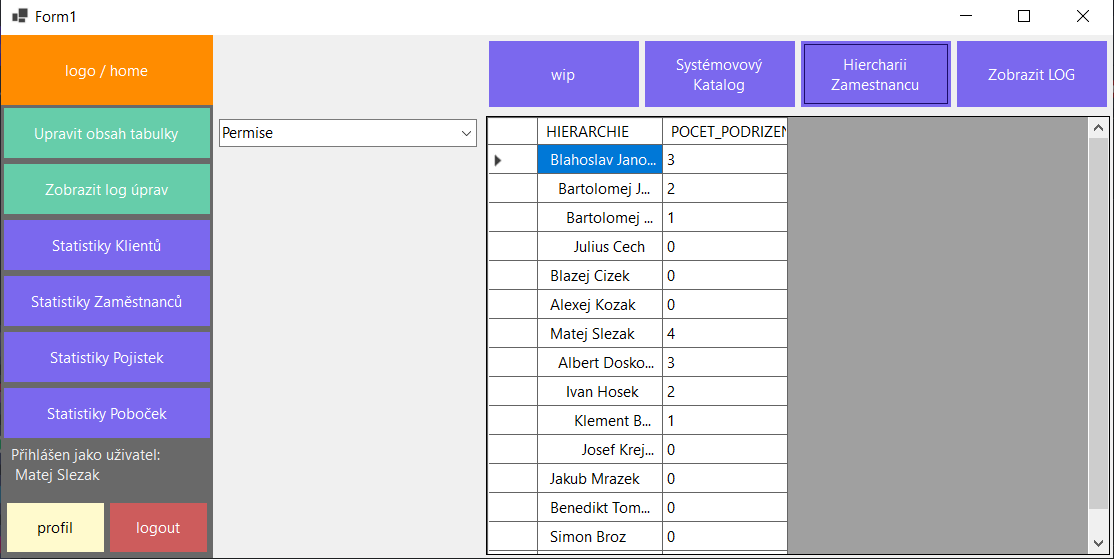
Obsah obrázku text, snímek obrazovky, software, Webová stránka

Popis byl vytvořen automaticky

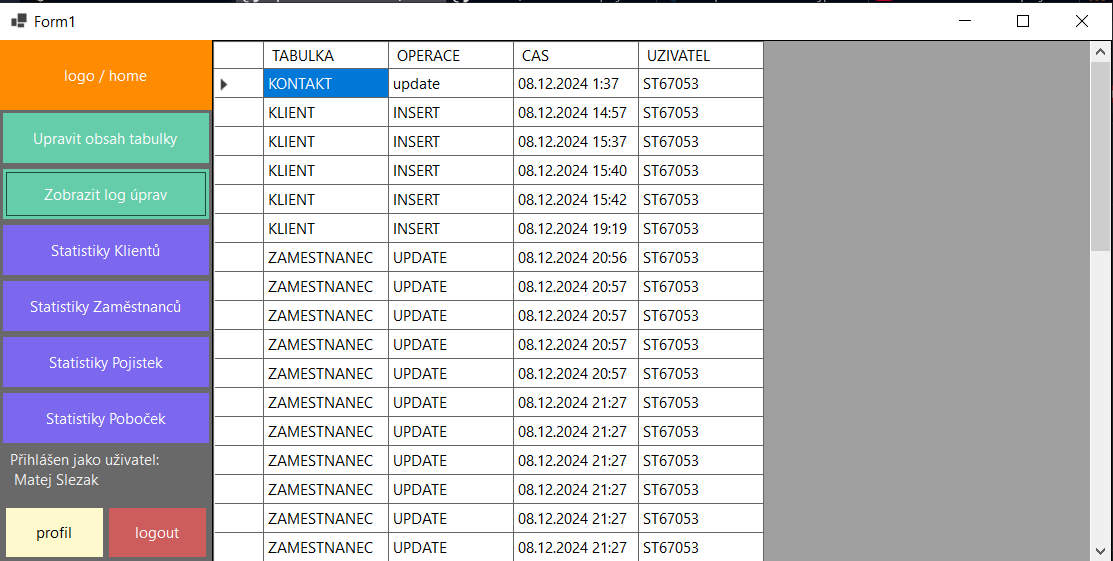
Okno po úspěšném přihlášení uživatele (oprávnění „Admin“). V levé části aplikace se nachází navigační menu, kde uživatel může přepínat okna. Admin si může zobrazit Domovskou stránku, Profil, Statistiky - Klientů, Zaměstnanců, Pojistek a Poboček, Upravovat tabulky, Zobrazit změny, dále se může emulovat mezi ostatní role.



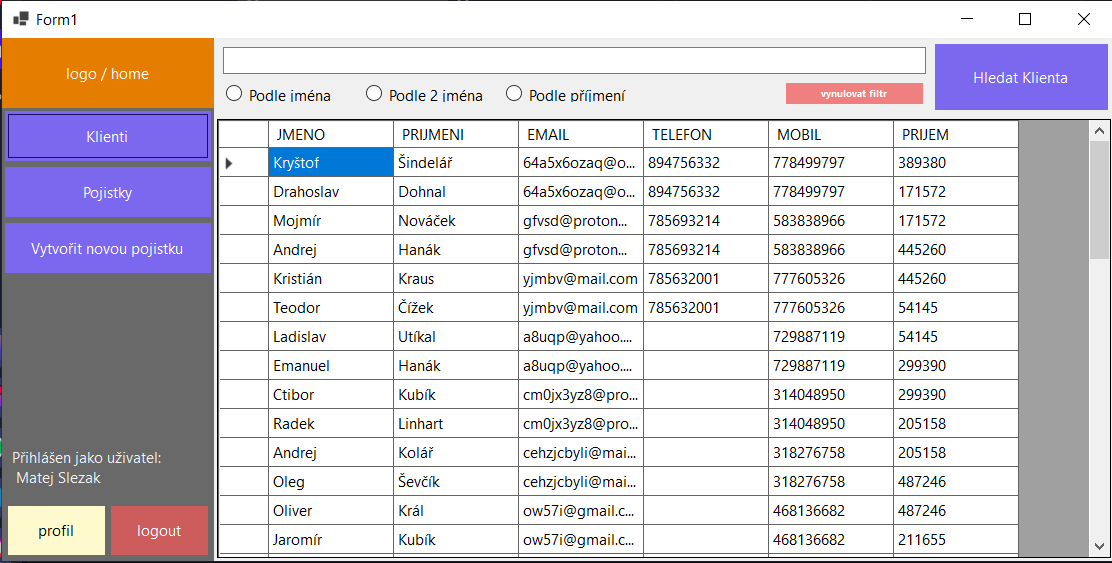
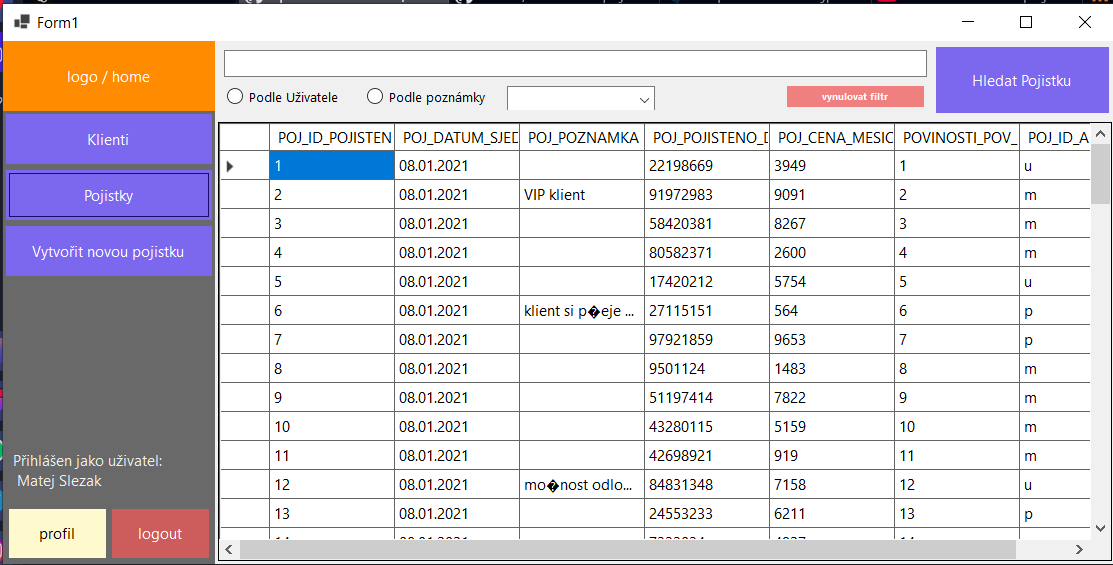
Admin si může zobrazit všechny tabulky, které jsou v databázi.



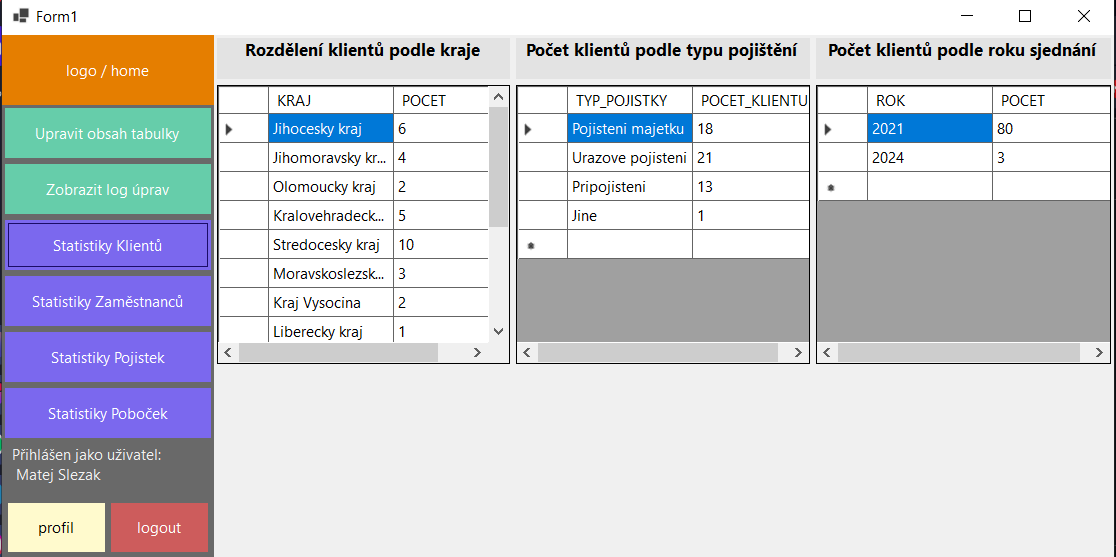
Administrátor si také může prohlédnout hierarchický strom zaměstnanců a podívat se, kolik má daný zaměstnanec podřízených.



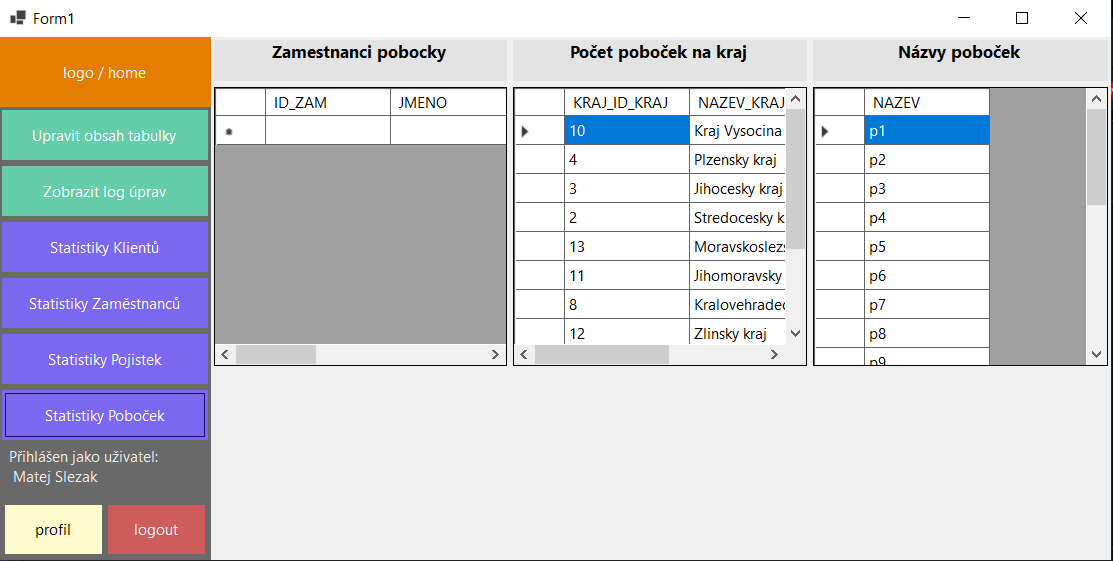
Administrátor si může si po kliknutí na „Zobrazit log úpravy“ zobrazit všechny uprávy které byly provedeny ať už přidaní, úprava nebo odstranění.

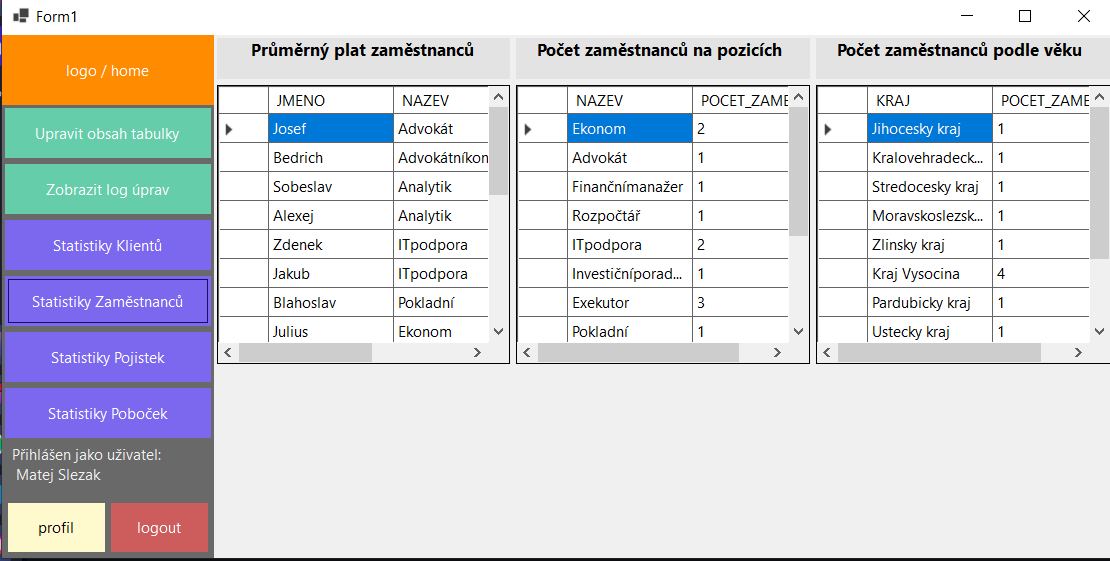
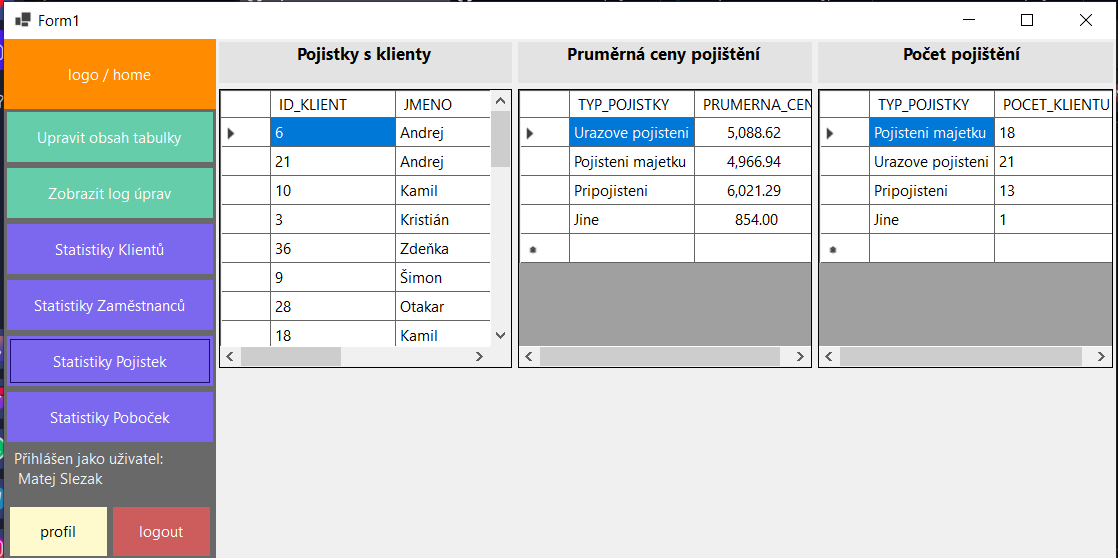


Zaměstnanec si po kliknutí na tlačítko “Pojistky” může zobrazit všechny pojistky, kteroé jsou aktivní a filtrovat si je podle poznámky, uživatele, nebo typu pojištění (např. Majetkové) poté je tam úplně stejná tabulka pro klienty kde si klienty může filtrovat podle jmen, prostředních jmen či příjmení – tak aby jednoduše mohl najít klienta, kterého potřebuje.



Analytik a admin mají možnost se přes tlačítko “Statistiky Klientů” dostat ke třem různým statistikám klientů, jak je vidět na obrázku, stejné statistiky jsou i pro zaměstnance, pobočky, pojistky viz. Obrázky níže





# Závěr

Aplikace slouží jako základ, na kterém by se dalo stavět. Bohužel transformace z BDAS1 byla náročnější, a její pravidelné týdenní úpravy nás dostali do bodu, že všech 30 bodů je vytvořeno v databázi, ale ve velké části se nevyužívají v aplikace. Trochu nás to mrzí, že se nám nepovedlo aplikaci plně dokončit.

Jinak uživatelské rozhraní je intuitivní a snadno použitelné, což přispívá k pozitivnímu uživatelskému zážitku. Funkčnost modulů, zejména komponent, poskytuje konkrétní hodnotu pro manipulaci s tabulkami dat a zlepšuje efektivitu práce.

V budoucnu má aplikace potenciál pro rozšíření funkcí. Je připravena na další změny a jednoduše rozšiřitelná, což poskytuje solidní základ pro další vývoj.

**Přílohy**

1. Zip File s kompletním repozitářem GitHub pro tento projekt
2. Dokumenty a vše z BDAS1
3. Program OASIS
4. Některé další dokumentační komentáře a informace, vycházející z repozitáře