UNIVERZITA PARDUBICE  
Fakulta elektrotechniky a informatiky

*Školní jídelna*

Seminární práce z předmětu Databázové systémy II

Pracovní tým:

Odintsova Polina,

Korotaeva Angelina

V Pardubicích dne 15.12.2025

Obsah

[Úvod 10](#_Toc177569060)

[Zadání 11](#_Toc177569061)

[Uživatelská dokumentace 12](#_Toc177569062)

[Základní popis používané aplikace 13](#_Toc177569063)

[Instalace 13](#_Toc177569064)

[Přístupová oprávnění 13](#_Toc177569065)

[Použití aplikace 13](#_Toc177569066)

[Programová dokumentace 20](#_Toc177569067)

[Datová část 21](#_Toc177569068)

[Analýza 21](#_Toc177569069)

[Fyzický model dat 31](#_Toc177569070)

[Číselníky 40](#_Toc177569071)

[Pohledy 41](#_Toc177569072)

[Funkce 43](#_Toc177569073)

[Uložené procedury 44](#_Toc177569074)

[Spouště 46](#_Toc177569075)

[Indexy 49](#_Toc177569076)

[Sekvence 53](#_Toc177569077)

[Systémový katalog 54](#_Toc177569078)

[Aplikace 57](#_Toc177569079)

[Použité prostředí 67](#_Toc177569080)

[Řízení uživatelských účtů 67](#_Toc177569081)

[Moduly 68](#_Toc177569082)

[Formuláře 69](#_Toc177569083)

[Orientace ve zdrojovém kódu 70](#_Toc177569084)

[Změny z původního návrhu 79](#_Toc177569085)

[Závěr 81](#_Toc177569086)

Úvod

Jsme manažeři školní jídelny na státní střední škole číslo 5, která byla založena v 19. století. Dnes naše škola poskytuje vzdělání studentům z různých společenských vrstev a naše jídelna se stará o stravování jak studentů, tak i zaměstnanců školy.

Minulý rok jsme se rozhodli vytvořit databázový systém pro efektivní správu stravování, který nám umožní lepší organizaci každodenních operací, tykajících se zprávy jídelny. Letos se tým rozhodl na základě vylepšeného databázového systému, jehož základem je loňský projekt, vytvořit aplikaci, která by dokázala vyhovět našim studentům i zaměstnancům a zpříjemnit jejích každodenní život po stránce stravovaní v naší jídelně.

Každý **strávník (student nebo pracovník)** má jedinečné *ID* *strávníka*, své *jméno*, *příjmení* a také sledovaný *zůstatek* na účtu, aby mohl objednávat jídlo. Strávníci mají *typ strávníka* a mohou být buď studenti, u kterých evidujeme *rok narození* a **třídu** (*ID* a *číslo třídy*), nebo pracovníci školy, u kterých sledujeme *telefonní číslo* a pracovní **pozici** (*ID* a *název*).

Navíc máme zaznamenány **adresy** všech strávníků, včetně *města*, *ulice* a *PSČ*. Někteří strávníci mohou mít zdravotní omezení, jako jsou **alergie** (jejich *ID*, *název* a dotčený *produkt*) nebo **dietní omezení** (*ID*, *název* a *popis*), které je třeba respektovat. Tyto údaje o alergiích a omezeních máme přehledně evidované.

Naše jídelna vytváří týdenní **menu**, která obsahují různé druhy **jídel**. Každé jídlo má své *ID*, *název*, *popis*, *kategorii* (např. polévka, hlavní jídlo, salát) a *cenu*. Menu je aktualizováno každý den dvakrát podle *typu menu* (snídaně nebo oběd) od pondělí do pátku.

Jednotlivá jídla jsou sestavena ze **složek** (surovin), které máme na skladě. **Složky** mají své *ID*, *název*, *měrnou jednotku* (např. kilogramy, litry atd.) a také *datum platnosti*, aby byla zajištěna čerstvost.

Strávníci si mohou objednávat jídlo pomocí naší aplikace. Každá **objednávka** má své *ID*, *datum vytvoření*, *celkovou cenu* a případné *poznámky*. Objednávka obsahuje jednotlivé **položky**, u kterých sledujeme konkrétní jídla, jejich *množství* a *cenu*. Pokud dojde k jakékoliv změně v přípravě objednávky, systém umožňuje zaznamenat její **stav** (např. čekající, připravená), kde evidujeme *ID* a *název* stavu.

Strávníci mohou své objednávky hradit různými způsoby – *hotově* při vyzvednutí ve škole, *kartou* nebo přes *školní účet*. U každé **platby** evidujeme *ID* platby, *datum*, *částku* a *metodu* platby.

Naše aplikace nám pomáhá udržovat přehled o všech aspektech provozu jídelny – od evidence strávníků a jejich požadavků až po správu dodávek a přípravu jídel. Věříme, že tento systém přispěje ke zlepšení kvality služeb jídelny a usnadní její každodenní provoz.

# Zadání

Naše organizace: Státní střední škola číslo 5, maximálně 20 pracovníků, 100 studentů.

IT prostředí: aplikace byla vytvořená v programovacím jazyce C# na platformě .NET prostřednictvím Visual Studio s využitím MVVM architektury. Databázová část systému je vytvořená pomocí Oracle Datamodeler a realizovaná pomocí Oracle sqldeveloper. Systém je určen pro školní prostředí, kde je síť lokální (LAN) a pracovní stanice jsou vybaveny operačním systémem Windows. Dosavadní využívání IT zahrnuje administrativní vedení dat o jídelně a evidenci objednávek v databázích.

Požadavky:

* Sledované údaje: informace o strávnicích (studentech a pracovnících), objednávky jídel, menu, alergie a dietní omezení, platby a historie objednávek.
* Vstupy:
  + - Přihlášení uživatele a zadávání objednávek.
    - Výběr jídel a zadávání plateb.
    - Zadávání, výběr nebo úpravy alergií a dietních omezení strávníků (studenti/pracovnici).
    - Administrátorské změny v tabulkách systému (menu, uživatelé, objednávky atd.).
* Výstupy:
  + - Potvrzení o úspěšné objednávce.
    - Seznam všech objednávek strávníka.
    - Výpis dat pro administrátora.
    - Zobrazení detailů strávníka.
* Přístupová práva:
  + - Administrátor – správa všech dat.
    - Uživatelské účty (pracovníci a studenti) – přehled menu, zadávání a kontrola vlastních objednávek a svých dat.

Uživatelská dokumentace

## Základní popis používané aplikace

Aplikace je určena pro školní jídelnu a bude používána studenty a zaměstnanci naší školy. Registrace je možná pouze pro uživatele, kteří jsou jíž registrovaní v databázi. Pomocí aplikace uživatelé si mohou prohlédnout aktuální menu, snadno objednávat jídla podle svých chutí, alergií nebo dietní omezení, sledovat své objednávky a jejich historii. Aplikace neřeší objednávky mimo školu ani platby mimo školní systém.

## Instalace

Aplikace je dostupná jako .exe soubor a nevyžaduje instalaci.

## Přístupová oprávnění

První uživatel registrovaný v aplikaci se automaticky stává administrátorem (admin). Při registraci admin zadává svou e-mailovou adresu a heslo a také své osobní údaje. Po registraci může admin přidávat základní data potřebná pro používání aplikace ostatními uživateli, například:

* seznam pozicí zaměstnanců,
* seznam tříd studentů,
* seznam jídel a menu,
* nejběžnější alergie a dietní omezení.

Ostatní uživatelé, studenti nebo pracovníci školy, mají roli **uživatel (user)**. Nejprve se musejí registrovat, poté se přihlásit a teprve potom mohou zadávat objednávky.

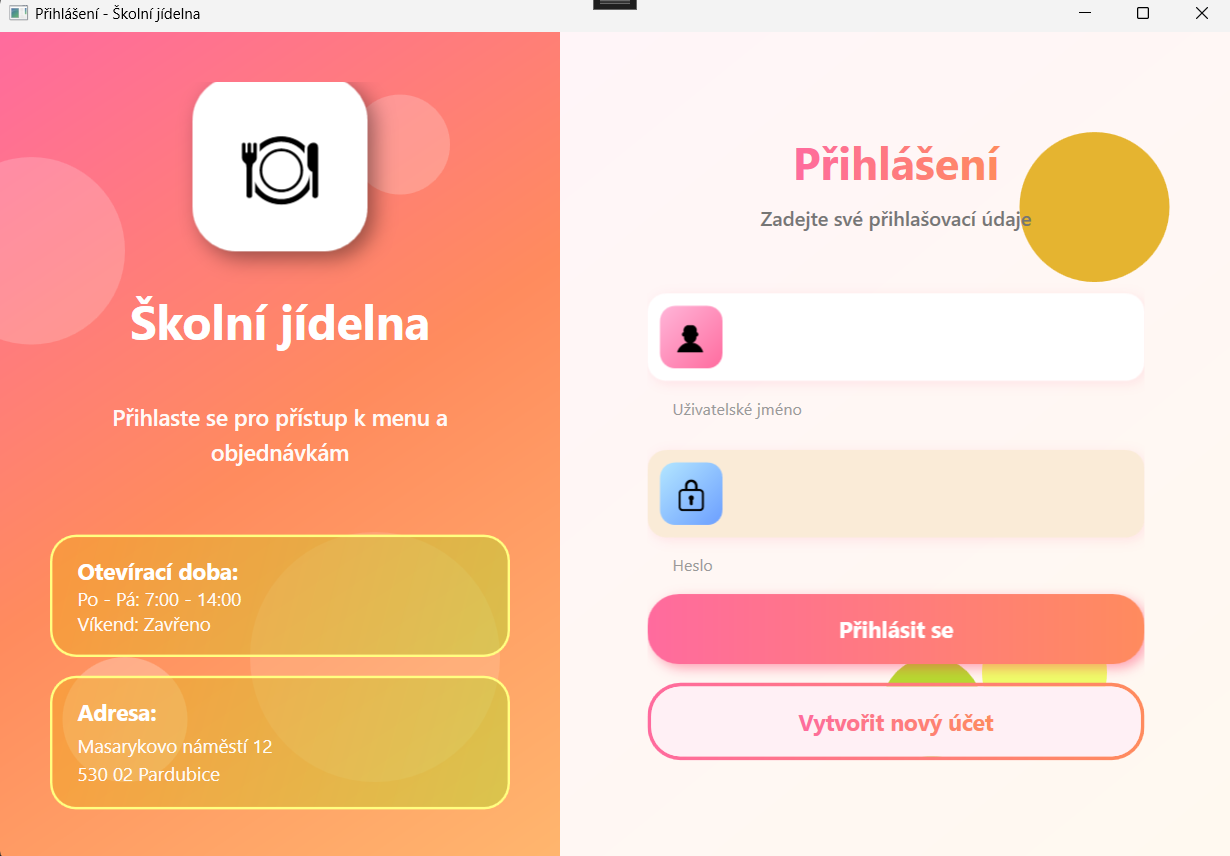
**Ukázkové přihlašovací údaje:**

* **Admin:** janN@email.cz / heslo123
* **Uživatel (student):** marie@seznam.cz / heslo123
* **Uživatel (pracovník):** karlNovotny@gmail.com / heslo123

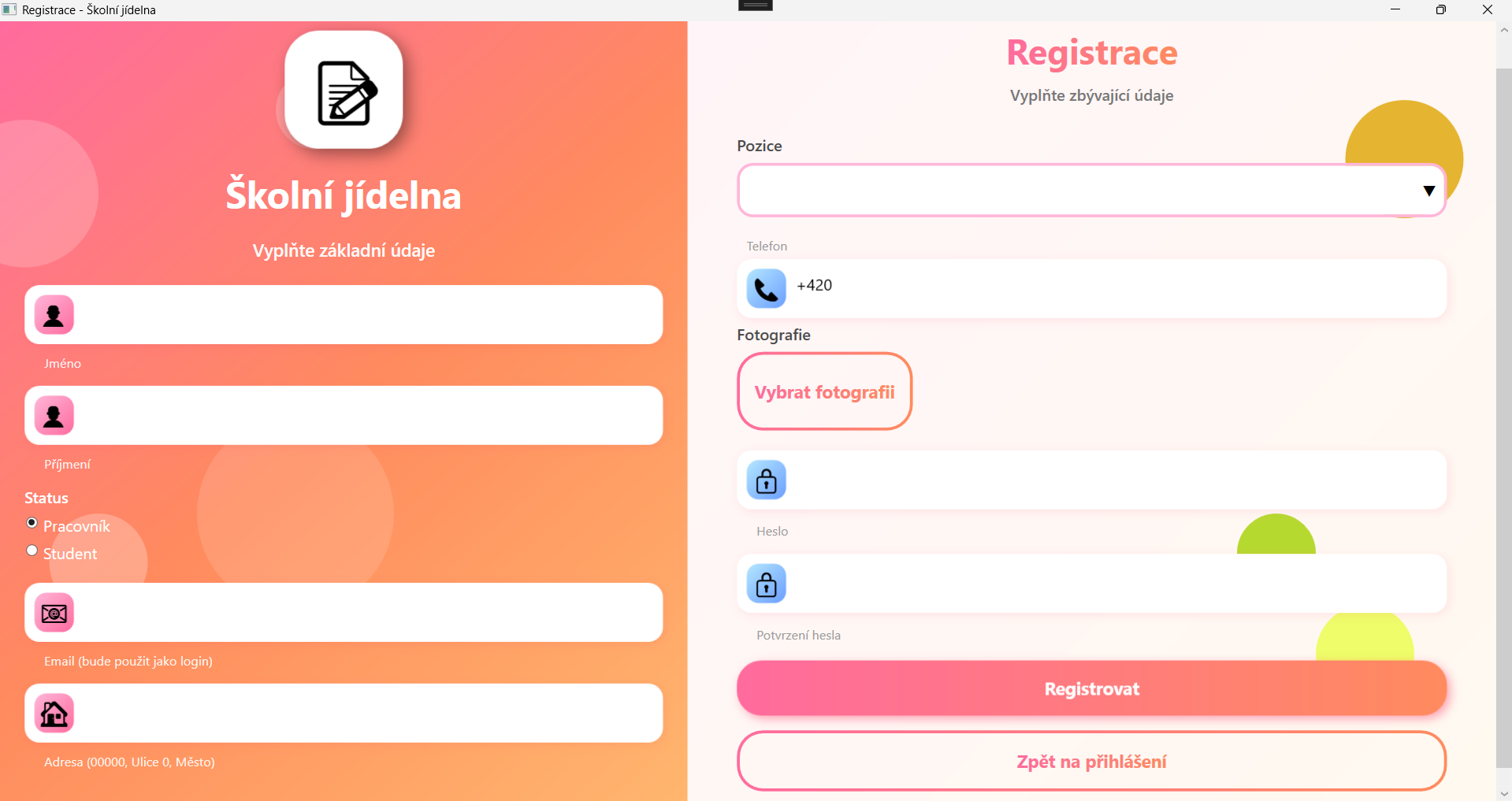
## Použití aplikace

Tato aplikace slouží k objednávání jídel a správě uživatelského profilu. Níže je uveden přehled jednotlivých modulů aplikace a jejich využití.

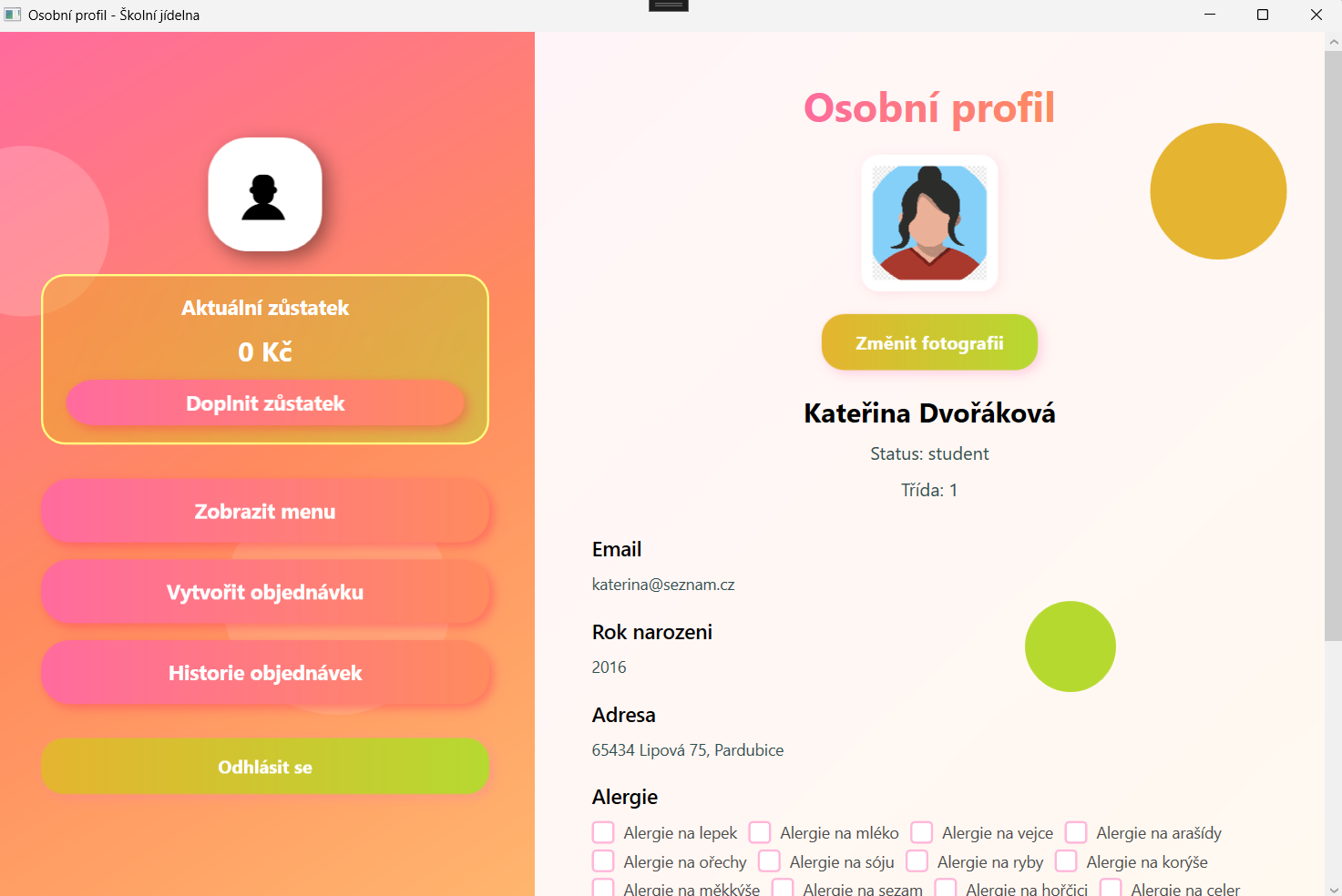
**1. Registrace a přihlášení**



Pokud nejste registrovaný student nebo pracovník naší školy, klikněte na tlačítko **„Registrace“**. Vyplňte své osobní údaje, e-mailovou adresu a vytvořte si heslo.

  
Po úspěšné registraci se přihlaste pomocí e-mailu a hesla, které jste zadali při registraci.

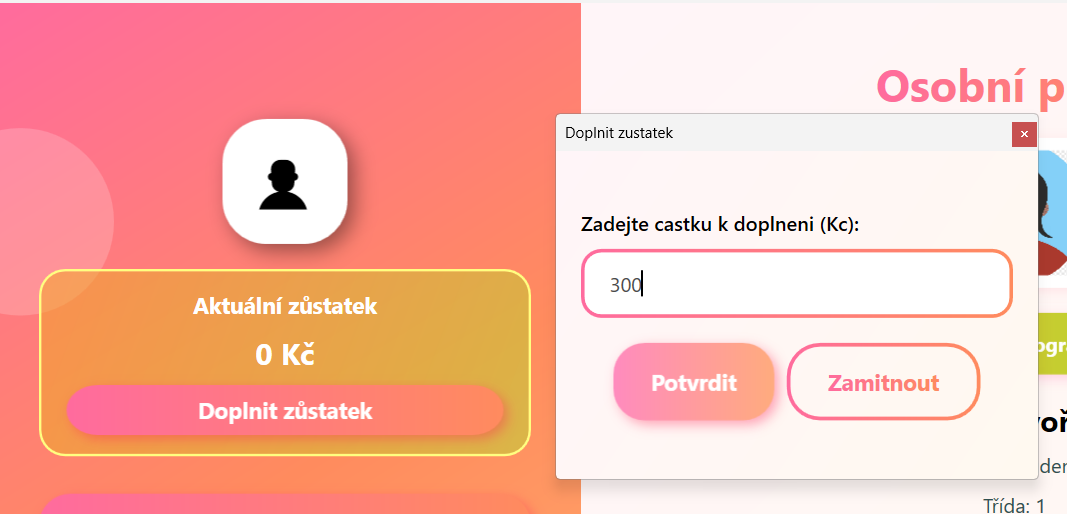
**2. Uživatelský profil**

****

Po přihlášení se zobrazí váš **uživatelský profil**. V této části můžete:

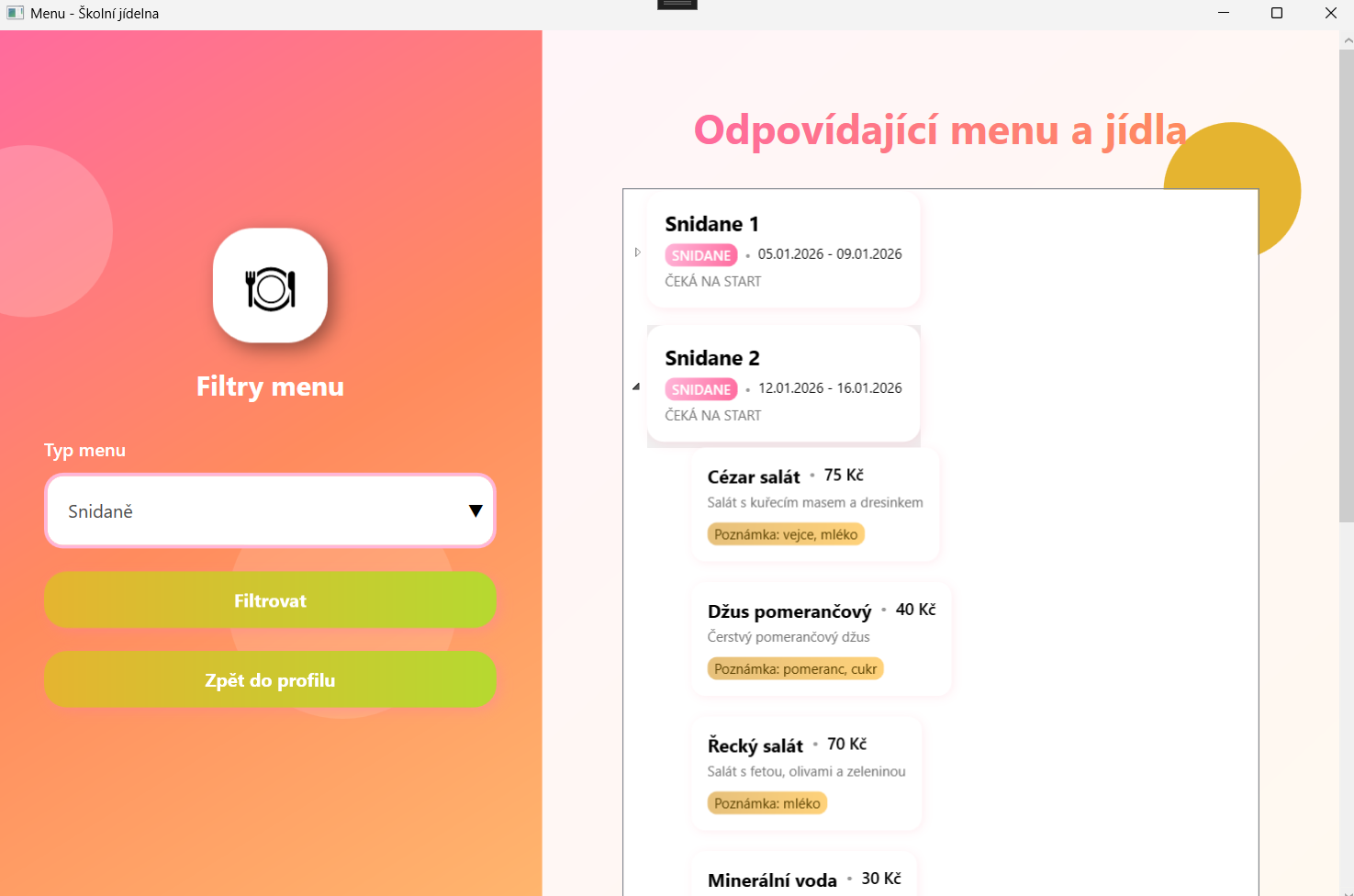
* zobrazit a upravit své osobní údaje,
* zadat alergie nebo dietní omezení (pokud jste studentem musíte však tyto změny provádět s pomocí administrátora),
* změnit profilovou fotografii,
* zobrazit aktuální **zůstatek na účtu (částku na účtu)**.

Pokud chcete, můžete si dobít zůstatek na účtu kliknutím na tlačítko **„Doplnit zůstatek“**. Tyto peníze lze následně použít k zaplacení objednávek v aplikaci.



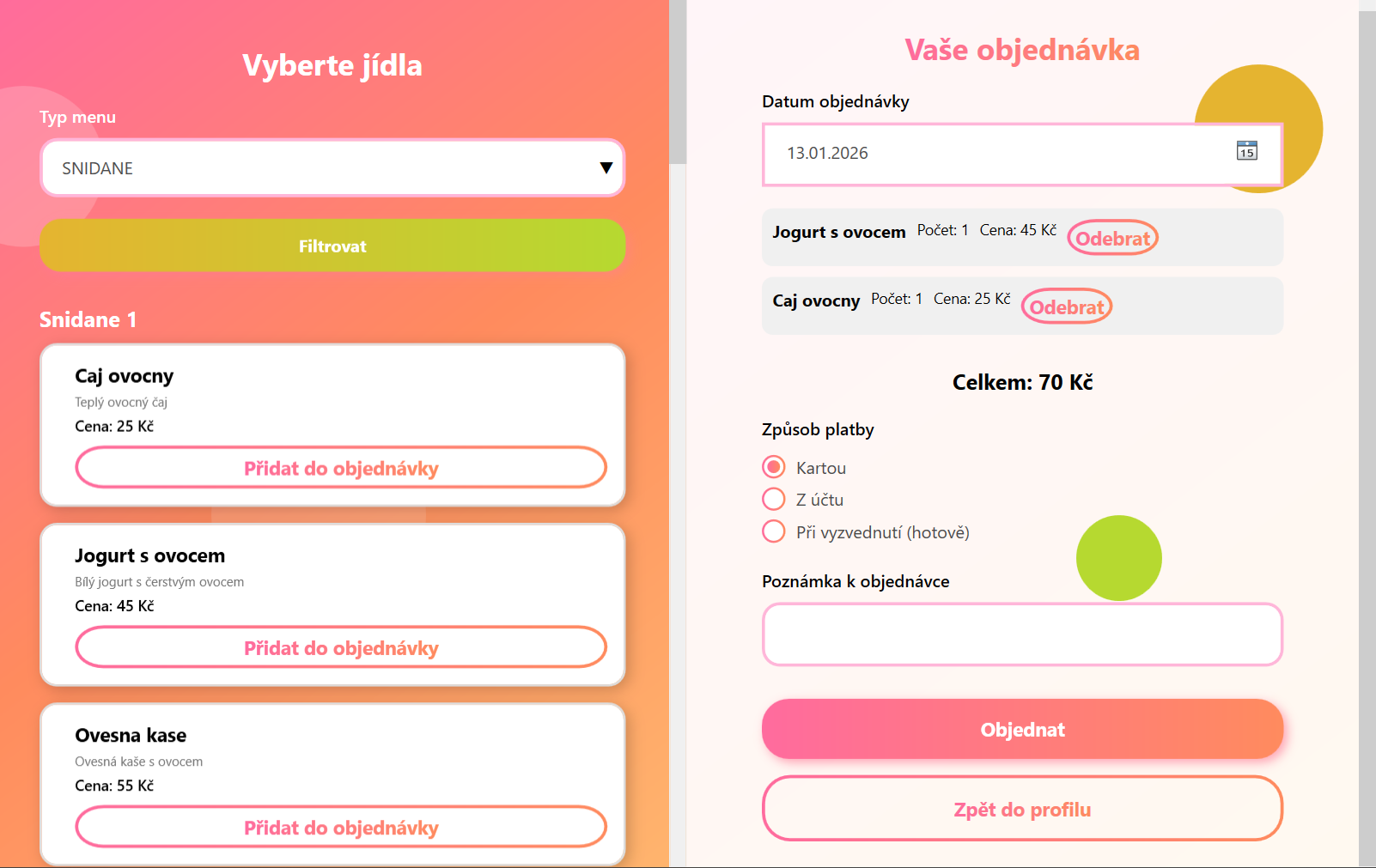
**3. Zobrazení menu**

Pro zobrazení aktuálního menu klikněte na tlačítko **„Zobrazit menu“**.  
Menu lze filtrovat podle typu jídla (snídaně nebo oběd) pomocí tlačítka **„Filtrovat“**.

  
Pro návrat na hlavní stránku klikněte na **„Zpět do profilu“**.

**4. Vytvoření objednávky**

Pro vytvoření nové objednávky klikněte na **„Vytvořit objednávku“**.

  
V této části můžete:

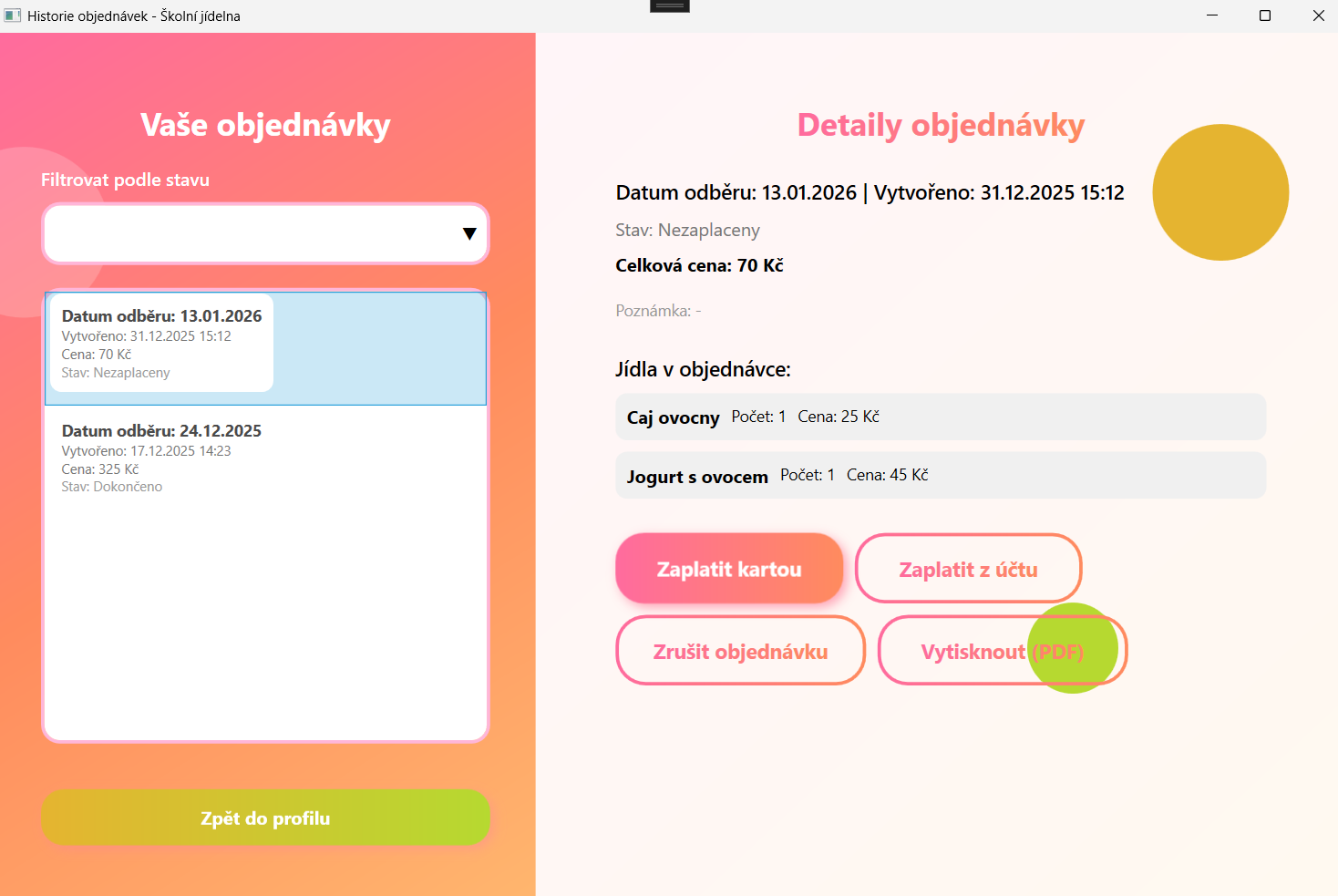
* filtrovat jídla podle typu menu pomocí tlačítka **„Filtrovat“**,
* vidět zvýraznění jídel nevhodných pro vaše alergie nebo dietní omezení (červeně jsou označená jídla, na jejíchž složky máte alergii, a žlutě označená jídla neodpovídají Vaším dietním omezením),
* zvolit datum vyzvednutí objednávky (mimo víkendy, aktuální nebo předešlé data),
* vybrat způsob platby,
* zadat případné požadavky do pole **Poznámky**.

Objednávku odešlete kliknutím na tlačítko **„Objednat“**.

Pro návrat na hlavní stránku použijte tlačítko **„Zpět do profilu“.**

**5. Historie objednávek**

V části **„Historie objednávek“** můžete zobrazit všechny své předchozí objednávky.

  
Pomocí filtru lze objednávky snadno vyhledávat.  
U nezaplacených objednávek je možné provést platbu online až do doby vyzvednutí.  
Dále zde můžete:

* stáhnout potvrzení objednávky ve formátu PDF,
* zrušit objednávku před jejím vyzvednutím.

Pro návrat na hlavní stránku klikněte na **„Zpět do profilu“**.

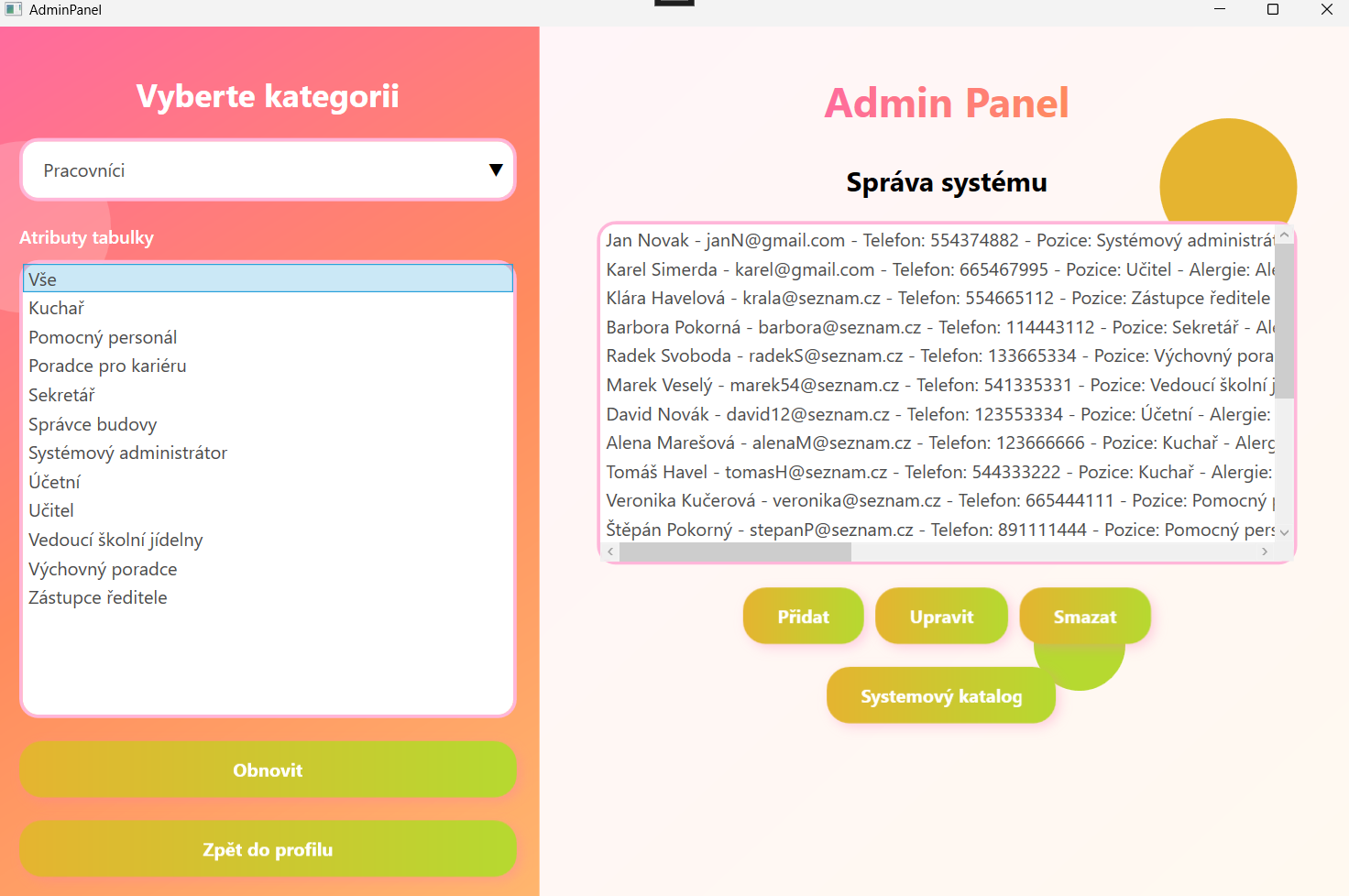
**6. Odhlášení**

Z aplikace se můžete kdykoliv odhlásit kliknutím na tlačítko **„Odhlásit se“**.

**7. Administrátorský panel**

****

Administrátor má v profilu k dispozici kromě jiného tlačítko **„Admin panel“**, kde může spravovat a upravovat údaje v aplikaci.



Programová dokumentace

# Datová část

## Analýza

## Strukturální a procedurální pravidla

**Strukturální pravidla:**

* **SP1:** Každá objednávka musí mít datum, celkovou cenu a musí být přiřazená stavu.
* **SP2:** Jedna objednávka musí mít alespoň jednu položku.
* **SP3:** Každý strávník musí mít jméno, příjmení, email a typ strávníka.
* **SP4:** Každý strávník může mít přiřazené alergie a dietní omezení.
* **SP5:** Jedna alergie nebo dietní omezení nemůže být přiřazeno stejnému strávníkovi vícekrát.
* **SP6:** Každá platba musí mít přiřazenou částku, datum a způsob platby.
* **SP7:** Platba může být propojena pouze s jedním strávníkem.
* **SP8:** Každá adresa musí obsahovat město, ulici a PSČ.
* **SP9:** Více strávníků může být propojeno s jednou adresou.
* **SP10:** Každá složka musí mít název a jednotku měření.
* **SP11:** Každé jídlo musí mít název, popis, cenu a kategorii.
* **SP12:** Každé menu musí mít název, typ a datum platnosti.
* **SP13:** Každá položka musí obsahovat množství a cenu.
* **SP14:** Studenti musejí mít uvedený datum narození, pracovníci musejí mít uvedené telefonní číslo.
* **SP15:** Každá objednávka musí mít právě jeden stav.
* **SP16:** Každý strávník může mít jeden nebo více souborů (osobní foto)

**Procedurální pravidla:**

* **PP1.** Zaregistrovat se může pouze uživatel, jehož základní data jsou v systému jíž evidovaná.
* **PP2.** Částka platby nemůže být záporná, způsob platby může nabývat hodnot: „hotově při vyzvednutí“, „karta“, „z učtu”.
* **PP3.** Strávník nemůže mít záporný zůstatek. Pokud nemá na účtu dostatek peněz, nemůže zaplatit objednávku z učtu.
* **PP4.** Zaměstnanec může pracovat pouze na 1 pozici, pokud je platnost pozice „do“ již v minulosti, nemůže zaměstnanec na této pozici pracovat.
* **PP5.** Student může patřit pouze do jedné třídy.
* **PP6.** Data o menu a platbách by měla být pravidelně archivována a zálohována.
* **PP7.** Objednávky lze zadávat pouze na datum v budoucnu. Objednávky pro aktuální nebo minulé datum nejsou povolené.
* **PP8.** Pokud se strávník nepřihlásí ke svému účtu po dobu delší než 3 měsíce, nebude mu umožněno se do účtu přihlásit a účet bude označen jako neaktivní.

## Jednoduchá analýza

**Jednoduchá analýza**

1. **Záznam entit**
2. **STRÁVNÍK**

# *id\_stravnik*

*\* jmeno*

*\* primeni*

*\* email*

*\* heslo*

*\* zůstatek*

*\* role*

*\* aktivita*

*\* typ strávníka*

*\* ID\_adresa*

1. **STUDENT**

*# id\_stravnik*

*\* vek*

*\* ID\_trida*

1. **TRIDA**

*# id\_trida*

*\* číslo\_tridy*

1. **PRACOVNIK**

*# id\_stravnik*

*\* telefon*

*\* ID\_pozice*

1. **POZICE**

*# id\_pozice*

*\* nazev*

1. **PLATBA**

**#** *id\_platba*

*\* datum*

*\* castka*

*\* metoda*

1. **ALERGIE**

*# id\_alergie*

*\* nazev*

*\* produkt*

1. **STRAVNIK\_ALERGIE**

*\* ID\_alergie*

*\* ID\_stravnik*

1. **DIETNI\_OMEZENI**

*# id\_omezeni*

*\* nazev*

*\* popis*

1. **STRAVNIK\_OMEZENI**

*\* ID\_omezeni*

*\* ID\_stravnik*

1. **ADRESA**

*# id\_adresa*

*\* psc*

*\* ulice*

*\* mesto*

1. **OBJEDNAVKA**

*# id\_objednavka*

*\* datum*

*\* celkova\_cena*

*o poznamka*

*\* ID\_stravnik*

*\* ID\_stav*

1. **STAV**

*# id\_stav*

*\* nazev*

1. **POLOZKA**

*\* množství*

*\* cena\_polozky*

*\* ID\_objednavka*

*\* ID\_jidlo*

1. **JIDLO**

*# id\_jidlo*

*\* nazev*

*\* popis*

*\* kategorie*

*\* cena*

*o poznamka*

*\* ID\_menu*

**16. MENU**

*# id\_menu*

*\* nazev*

*\* typ\_menu*

*\* time\_od*

*\* time\_do*

**17. SLOZKA\_JIDLO**

*\* množství*

*o poznámka*

*\* ID\_jidlo*

*\* ID\_slozka*

1. **SLOZKA**

*# id\_slozka*

*\* nazev*

*\* merna\_jednotka*

*\* datum\_platnosti*

1. **SOUBOR**

*# id\_soubor*

*\* nazev*

*\* typ*

*\* pripona*

*\* obsah*

*\* datum\_nahrani*

*o datum\_modifikace*

*o operace*

*\* tabulka*

*\* id\_zaznam*

*o ID\_stravnik*

1. **LOG**

*# id\_log*

*\* tabulka*

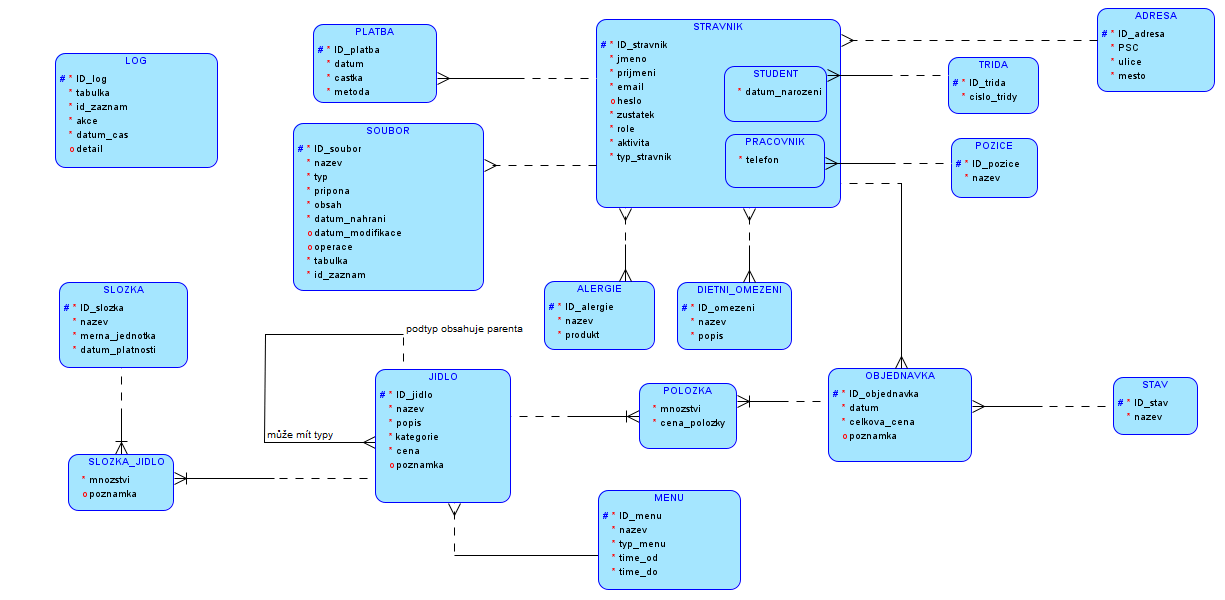
*\* id\_zaznam*

*\* akce*

*\* datum\_cas*

*o detail*

## Konceptuální schéma – ERD



### Diskuze smyček

V modelu nejsou žádné smyčky.

### ERDish věty

Strávník – Adresa:

* Každý aktivní strávník někde bydli.
* Na určité adrese může bydlet strávník.

Jeden uživatel může mít pouze 1 adresu, ale na dané adrese může žít více osob.

Strávník – Alergie (Typ relace: Many to many, přes Alergie\_Strávníka):

* Strávník může mít alergii.
* Alergie se může týkat strávníka.

Jeden strávník může mít více alergií a jednu alergii může sdílet více strávníků.

Strávník – Dietní omezeni (Typ relace: Many to many, přes Omezeni\_Strávníka):

* Strávník může mít dietní omezeni.
* Dietní omezeni se může objevit u strávníka.

Jeden strávník může mít více dietních omezení a jedno dietní omezeni se může vyskytovat u více strávníků.

Strávník – Platba:

* Strávník může mít platbu.
* Platby musí být přiřazená strávníku.

Jeden strávník může mít mnoho plateb, ale každá platba musí být přiřazená pouze jednomu strávníku.

Student – Třída:

* Student musí být v určité třídě.
* Třída může mít studenta.

Jeden student musí být pouze v jedné třídě, ale ve stejné třídě může být více studentů.

Pracovník – Pozice:

* Pracovník musí být na určité pozici.
* Pozice se může vztahovat k pracovníku.

Jeden pracovník musí zastávat pouze jednu pozici, ale na stejné pozici může být více pracovníků.

Strávník – Objednávka:

* Strávník může mít objednávku.
* Objednávka musí patřit určitému strávníku.

Jeden strávník může mít více objednávek, ale každá objednávka musí patřit pouze jedenomu strávníku.

Objednávka – Stav:

* Objednávka musí mít stav.
* Stav může být přirazen k objednávce.

Jedna objednávka musí mít současně pouze jeden stav, ale jeden stav může být přiřazen více objednávkám současně.

Objednávka – Jídlo (Typ relace: Many to many, přes Položka):

* Objednávka musí mít jídla.
* Jídla lze přiřadit k objednávce.

Jedna objednávka se může skládat z několika jídel, jedno jídlo může být obsaženo v několika objednávkách.

Jídlo – Složka (Typ relace: Many to many, přes Složka\_Jídla):

* Jídlo může obsahovat určitou složku.
* Složka může být obsažena v jídle.

Jedno jídlo se může skládat z několika složek, jedna složka může být použita ve více jídlech.

Jídlo – Jídlo (Typ relace: Rekurzivní relace, Jeden to Many);

* Jedno parent jídlo může být nadřazeno více child jídlům.
* Každé child jídlo musí mít právě jedno parent jídlo.

Tato relace je realizována pomocí cizího klíče parent\_jidlo, který odkazuje na primární klíč id\_jidlo ve stejné tabulce **JIDLA**.

Jídlo – Menu:

* Jídlo může být v menu.
* Menu musí obsahovat jídla.

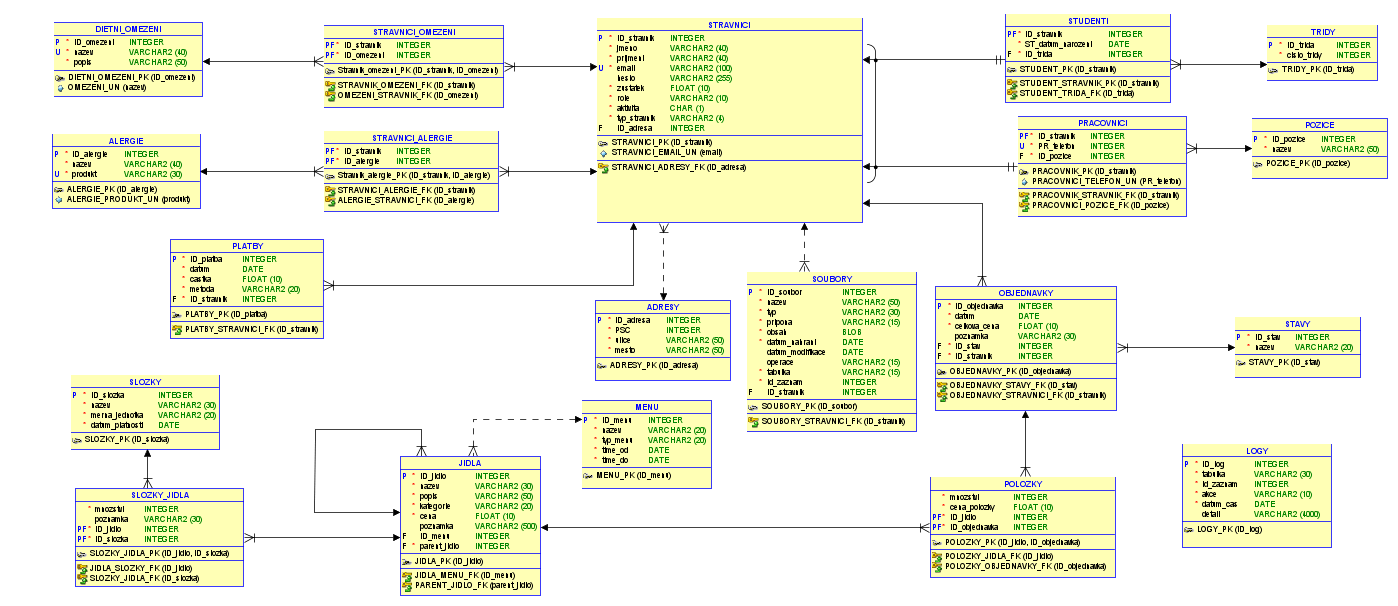
Jedno jídlo se může objevit ve více menu, ale každé menu musí mít uvedené aspoň jedno jídlo.

Soubor - Strávnik:

* Soubor musí patřit strávníku.
* Strávník může mít soubory.

Strávník může mít soubory, ale každý soubor musí patřit jednomu strávníku.

## Relační model dat



Relační model dat pro databázi školní jídelny byl vytvořen na základě konceptuálního modelu (ERD) a transformován do podoby relační databáze. Tento model obsahuje všechny entity, jejich atributy, primární a cizí klíče a vztahy mezi tabulkami.

Tabulky jsou pojmenovány v množném čísle, například “ADRESY”, “STRAVNICI”, “PRACOVNICI”.

**Transformace a mapovaní**

1. **Primární klíče (PK):**

Každá tabulka má svůj primární klíč, který zajišťuje jedinečnost každého záznamu.

* Tabulka **STRAVNICI** má primární klíč id\_stravnik.
* Tabulka **OBJEDNAVKY** má primární klíč id\_objednavka
* Tabulka **JIDLA** má primární klíč id\_jidlo atd.

1. **Cizí klíče (FK):**

Cizí klíče propojují entity mezi sebou, aby byly definovány vztahy.

* Tabulka **PRACOVNICI** obsahuje ID\_pozice jako cizí klíč, který odkazuje na tabulku **POZICE**.
* Tabulka **STUDENTI** obsahuje ID\_trida jako cizí klíč odkazující na tabulku **TRIDY**.
* Tabulka **JIDLA** obsahuje ID\_menu jako cizí klíč odkazující na tabulku **MENU** atd.

1. **Vazební tabulky:**

Vazby M:N byly rozloženy na samostatné tabulky s primárními a cizími klíči.

* Tabulka **STRAVNICI\_ALERGIE** propojuje tabulky **ALERGIE** a **STRAVNICI**. Obsahuje primární klíč složený z atributů id\_alergie a id\_stravnik.
* Tabulka **STRAVNICI\_OMEZENI** propojuje tabulky **DIETNI\_OMEZENI** a **STRAVNICI** atd.

**Supertyp a subtyp**

* **STRAVNICI** funguje jako supertyp pro dvě podtřídy:
  + **STUDENTI:** Mají atribut věk, který sleduje věk studenta.
  + **PRACOVNICI:** Mají atributy telefon, který sleduje telefon pracovníka.

Do tabulky **STRAVNICI** byl přidán atribut *typ\_stravnik (ST, PR)*. Subentity sdílejí primární klíč id\_stravnik ze supertypu **STRAVNICI** a obsahují specifické atributy pro daný typ osoby.

# Implementace databáze

Sekvence a trigery jsme přidaly do některých tabulek z několika důvodů:

1. **Automatická generace primárních klíčů**  
   Sekvence a trigery zajišťují, že každý nový řádek v tabulce bude mít unikátní hodnotu v primárním klíči (např. id\_objednavka nebo id\_menu). To eliminuje riziko duplicitních hodnot a usnadňuje správu databáze.
2. **Jednodušší vkládání dat**  
   Díky triggerům není při vkládání dat nutné manuálně specifikovat hodnotu primárního klíče. Hodnota se automaticky generuje pomocí sekvence, což zjednodušuje zápisy do tabulek a snižuje riziko chyb.

Máme sekvence pro tabulky STRAVNICI, PLATBY, OBJEDNAVKY, JIDLA, MENU, ADRESY, ALERGIE, DIETNI\_OMEZENI, SLOZKY, SOUBORY, LOGY, TRIDY, POZICE.

Například:

CREATE SEQUENCE s\_str START WITH 1 INCREMENT BY 1 NOCACHE;

## Fyzický model dat

A close-up of a document

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

A close-up of a form

AI-generated content may be incorrect.

A screenshot of a document

AI-generated content may be incorrect.

A close-up of a form

AI-generated content may be incorrect.

A screenshot of a document

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a document

AI-generated content may be incorrect.

A table of information with text

AI-generated content may be incorrect.

A close-up of a form

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a document

AI-generated content may be incorrect.

## Číselníky

**1) Číselník stavů (STAVY)**

**Popis: Používá se k určení stavu objednávky.**

**Zdroj: Každá objednávka má určitý stav, který se mění v průběhu jejího zpracování.**

**Příklad hodnot:**

* **V procesu**
* **Hotovo**

1. **Číselník tříd (TRIDY)**

**Popis: Určuje seznam všech tříd, do kterých chodějí studenti.**

**Zdroj: Ve škole každý student chodí do určité třídy.**

**Příklad hodnot:**

* Třída 2
* Třída 4

1. **Číselník pozicí pracovníků (POZICE)**

**Popis: určuje seznam všech pozicí pracovníků.**

**Zdroj: V práci má každý pracovník určitou pozici.**

**Příklad hodnot:**

* Systémový administrátor
* Učitel

## Pohledy

V této kapitole jsou popsány databázové pohledy použité v projektu. Pohledy slouží k přehlednému zpřístupnění dat aplikační vrstvě, zjednodušují SQL dotazy v aplikaci a zajišťují jednotnou strukturu vracených dat.

1. V\_JIDLA\_MENU

Pohled zobrazuje seznam jídel včetně jejich základních vlastností a informace o menu, do kterého jídlo patří. Slouží pro zobrazení nabídky jídel v aplikaci.

**Využití v projektu:**  
Používán v aplikační vrstvě pro zobrazení menu (snídaně, obědy apod.) koncovému uživateli.

1. V\_JIDLA\_SLOZENI

Pohled poskytuje detailní informace o složení jednotlivých jídel. Obsahuje seznam ingrediencí, jejich množství a měrné jednotky, agregované pomocí funkce LISTAGG.

**Využití v projektu:**  
Používán při zobrazení detailu jídla, zejména pro informování uživatelů o složení pokrmů (alergie, dietní omezení).

1. V\_OBJ\_PREHLED\_ADMIN

Pohled zobrazuje základní přehled objednávek – identifikaci objednávky, jméno zákazníka, stav objednávky, datum a celkovou cenu v administrační části aplikace.

**Využití v projektu:**  
Používán pro zobrazení seznamu objednávek v administrační části aplikace.

1. V\_OBJEDNAVKY\_HISTORIE\_DETAIL

Pohled poskytuje kompletní přehled objednávek včetně položek, stavů, dat vytvoření a vyzvědnutí. Obsahuje také informace z logovací tabulky.

**Využití v projektu:**  
Používán pro historii objednávek strávníka a pro administrativní přehledy.

1. V\_STR\_OMEZENI\_ALERGIE

Pohled zobrazuje seznam strávníků a jejich alergie a dietní omezení. Hodnoty jsou agregovány pomocí LISTAGG.

**Využití v projektu:**  
Používán při filtrování jídel a kontrole vhodnosti pokrmů pro konkrétního uživatele.

1. V\_STRAVNICI\_FULL

Pohled sdružuje kompletní informace o strávníkovi – osobní údaje, adresu, kontaktní informace, roli a případně pozici nebo třídu.

**Využití v projektu:**  
Používán v uživatelském profilu a v administrační části aplikace.

1. V\_STRAVNICI\_LOGIN

Pohled vrací pouze aktivní uživatele a jejich přihlašovací údaje.

**Využití v projektu:**  
Používán při autentizaci uživatele (login).

1. V\_STUD\_TRIDA

Pohled zobrazuje studenty, jejich osobní údaje a přiřazenou třídu.

**Využití v projektu:**  
Používán v administrační části aplikace pro přehled studentů podle tříd.

1. V\_PR\_POZICE

Pohled zobrazuje pracovníky školy, jejich pozici, kontaktní údaje a případné alergie nebo dietní omezení.

**Využití v projektu:**  
Používán v administrační části aplikace pro správu zaměstnanců.

## Funkce

V této kapitole jsou popsány databázové funkce použité v projektu. Funkce slouží k výpočtům a logickému zpracování dat na databázové vrstvě a jsou využívány v aplikační vrstvě při zobrazování informací uživateli a při kontrole aplikační logiky.  
SQL skripty funkcí jsou uvedeny v **Příloze A – SQL skripty**.

1. F\_CELKOVA\_CENA

Funkce vrací celkovou cenu objednávky na základě její položek a jejich množství. Výsledná cena je vypočtená jako součet součinů ceny položky a množství jednotlivých položek v objednávce. Pokud objednávka neobsahuje žádné položky, funkce vrací hodnotu 0.

**Využití v projektu:**  
Používá se v aplikační vrstvě při zobrazení detailu objednávky a při výpočtu celkové ceny objednávky.

1. F\_OBJEDNAVKA\_STAV

Funkce vrací textový přehled počtu objednávek strávníka podle jejich stavů (v procesu, dokončeno, zrušeno, nezaplaceno). Pro každý stav je vypočítán počet objednávek pomocí agregačních funkcí.

**Využití v projektu:**  
Používá se v uživatelském profilu pro zobrazení přehledu objednávek konkrétního strávníka.

1. F\_STAV\_MENU

Funkce určuje aktuální stav menu na základě časového období platnosti menu. Podle porovnání aktuálního data se začátkem a koncem platnosti menu vrací jeden z následujících stavů:

* “ČEKÁ NA START“
* “AKTIVNÍ“
* “UKONČENO“

Pokud zadané menu neexistuje, funkce vrací informaci o neexistenci menu.

**Využití v projektu:**  
Používá se při zobrazování nabídky jídel pro určení, zda je menu aktuálně dostupné.

1. F\_ZUSTATEK

Funkce kontroluje, zda má strávník na svém účtu dostatečný zůstatek pro zaplacení zadané částky. Na základě porovnání vrací jeden z následujících stavů:

* „OK“ – dostatečný zůstatek
* „NEDOSTATECNY“ – nedostatečný zůstatek
* „NEEXISTUJE“ – strávník neexistuje

**Využití v projektu:**  
Používá se při vytváření objednávky a při kontrole možnosti její úhrady.

## Uložené procedury

**Povinné procedury (nezahrnující vkládání a modifikaci dat)**

1. P\_ODSTR\_STAR\_OBJ

**Účel:**  
Dávková procedura pro odstranění objednávek starších než 12 měsíců.

**Využití v projektu:**  
Procedura je určena pro administrativní údržbu databáze a je spouštěna z administrační části aplikace nebo manuálně správcem systému.

1. P\_SOUBORY\_STRAVNIKA

**Účel:**  
Procedura vypisuje seznam souborů přiřazených ke konkrétnímu strávníkovi.

**Využití v projektu:**  
Používá se v profilu uživatele nebo v administrační části aplikace pro přehled nahraných souborů (např. profilová fotografie).

1. P\_STATISTIKA\_STAVU

**Účel:**  
Statistická procedura zobrazující počet objednávek pro jednotlivé stavy.

**Využití v projektu:**  
Využívána v administrativním přehledu objednávek a pro kontrolu aktuálního stavu systému.

1. P\_VYPIS\_JIDEL

**Účel:**  
Procedura zajišťuje výpis všech jídel v databázi včetně jejich názvů a cen.

**Využití v projektu:**  
Používá se pro rychlou orientaci v nabídce jídel, zejména při správě menu.

Ostatní procedury

1. P\_CREATE\_EMPTY\_STRAVNIK

**Účel:**  
Vytváří nový neaktivní účet strávníka se základními údaji (jméno, příjmení, email, typ). Heslo a adresa nejsou nastavena, zůstatek je 0, aktivita = NEAKTIVNÍ.

**Využití v projektu:**  
Je volána při registraci nového strávníka administrátorem, pro základní vytvoření účtu před jeho aktivací.

1. P\_INSERT\_OBJ

**Účel:**  
Vytváří novou objednávku a ukládá její základní údaje.

**Využití v projektu:**  
Je volána při vytváření nové objednávky, zejména jako součást transakční procedury.

1. P\_INSERT\_POLOZKA

**Účel:**  
Přidává jednotlivé položky do objednávky a ukládá jejich cenu.

**Využití v projektu:**  
Používá se při sestavování objednávky uživatelem.

1. P\_PRIDAT\_FOTO

**Účel:**  
Ukládá profilovou fotografii strávníka do databáze.

**Využití v projektu:**  
Používá se při registraci nebo úpravě profilu uživatele.

1. P\_REGISTER\_PRACOVNIK

**Účel:**  
Aktivuje neaktivního strávníka jako pracovníka a přiřazuje mu adresu, heslo, zůstatek a pozici.

**Využití v projektu:**  
Používá se při registraci pracovníka z aplikační vrstvy.

1. P\_REGISTER\_STUDENT

**Účel:**  
Aktivuje neaktivního strávníka jako studenta a přiřazuje mu adresu, heslo, zůstatek a třídu.

**Využití v projektu:**  
Používá se při registraci studenta z aplikační vrstvy.

1. TRANS\_CREATE\_ORDER

**Účel:**  
Transakční procedura pro vytvoření kompletní objednávky včetně položek.

**Využití v projektu:**  
Je volána při potvrzení objednávky uživatelem.

1. TRANS\_REGISTER\_PRACOVNIK

**Účel:**  
Bezpečnostní transakční procedura pro registraci pracovníka. Zajišťuje commit/rollback.

**Využití v projektu:**  
Používá se z administrační části aplikace pro zajištění bezpečné registrace pracovníka.

1. TRANS\_REGISTER\_STUDENT

**Účel:**  
Bezpečnostní transakční procedura pro registraci studenta. Zajišťuje commit/rollback.

**Využití v projektu:**  
Je volána při registraci studenta z aplikační vrstvy.

## Spouště

Projekt využívá rozsáhlý systém databázových spouští, které zajišťují integritu dat, automatizaci operací, logování změn a implementaci obchodních pravidel na databázové vrstvě.

1. **Logovací spouště (Audit dat)**

**Popis:**

Logovací spouště zaznamenávají všechny změny dat (INSERT, UPDATE, DELETE) do tabulky LOGY.  
Každá operace obsahuje informaci o tabulce, typu akce, identifikátoru záznamu a času provedení.

**Příklady spouští:**

* T\_STRAVNICI\_LOG
* T\_OBJEDNAVKY\_LOG
* T\_PLATBY\_LOG

**Použití v projektu:**

* audit změn,
* historie záznamů,
* přístupné pouze administrátorovi,
* splnění bodů **14 a 21 zadání**.

**Tabulky:**

* ADRESY
* ALERGIE
* DIETNI\_OMEZENI
* STRAVNICI
* JIDLA
* OBJEDNAVKY
* POLOZKY
* MENU
* PLATBY
* SLOZKY
* STUDENTI
* PRACOVNICI
* SLOZKY\_JIDLA
* STAVY
* STRAVNICI\_ALERGIE
* STRAVNICI\_OMEZENI

1. **Spouště pro automatické generování ID**

**Popis:**

Tyto spouště automaticky doplňují hodnoty umělých primárních klíčů pomocí databázových sekvencí.  
Aplikace tak nikdy nemanipuluje s ID hodnotami přímo.

**Příklady spouští:**

* T\_STR\_ID
* T\_OBJ\_ID
* T\_JID\_ID

**Použití v projektu:**

* vkládání záznamů z formulářů,
* splnění bodů 2 a 13 zadání.

**Tabulky:**

* STRAVNICI
* OBJEDNAVKY
* JIDLA
* MENU
* PLATBY
* ADRESY
* LOGY
* SOUBORY
* ALERGIE
* DIETNI\_OMEZENI
* SLOZKY
* POZICE
* STAVY
* TRIDY

1. **Spouště pro kontrolu obchodních pravidel a automatické zpracování dat**

Projekt využívá několik databázových spouští, které implementují obchodní logiku aplikace přímo na databázové vrstvě. Tyto spouště zajišťují:

* správnou práci s finančními prostředky,
* automatické doplnění výchozích hodnot,
* kontrolu stavů objednávek,
* ochranu proti nekonzistenci dat.

Spouště fungují **pro ochranu obchodních pravidel a zajištění integrity dat**.

1. T\_OBJ\_PLATBA (kontrola zůstatku)

**Popis:**  
Před vložením nové objednávky spoušť kontroluje, zda má strávník dostatečný zůstatek na účtu. Pokud částka objednávky přesahuje zůstatek, je vyvolána databázová výjimka (RAISE\_APPLICATION\_ERROR) a objednávka není vytvořena.

**Použití v aplikaci:**

* ochrana proti zápornému zůstatku,
* validace objednávky na databázové vrstvě,
* splnění bodu 14 a 16 zadání (kontrola integrity a transakční bezpečnost).

1. T\_OBJ\_PLATBA (automatická úprava zůstatku)

**Popis:**  
Po vložení objednávky je částka automaticky odečtena ze zůstatku strávníka. Po vložení platby je zůstatek strávníka automaticky navýšen o zaplacenou částku.

**Použití v aplikaci:**

* zajištění konzistence finančních dat,
* transakční bezpečnost při změnách zůstatku,
* automatická aktualizace stavu účtu.

1. T\_OBJ\_DATE a T\_OBJ\_STATUS (automatické doplnění výchozích hodnot)

**Popis:**  
Před vložením nové objednávky spouště doplňují výchozí hodnoty:

* T\_OBJ\_DATE – pokud není zadáno datum, automaticky se nastaví na datum nejbližšího z následujících pracovních dní.
* T\_OBJ\_STATUS – pokud není zadán stav objednávky, nastaví se výchozí stav (id\_stav = 3).

**Použití v aplikaci:**

* zjednodušení práce uživatele,
* zajištění konzistentních dat v tabulce objednávek,
* eliminace chyb při ručním zadávání hodnot.

1. T\_PLATBA\_ZUSTATEK (automatická aktualizace po platbě)

**Popis:**  
Po vložení záznamu do tabulky platby se zůstatek strávníka automaticky navýší o vloženou částku.

**Použití v aplikaci:**

* automatická synchronizace zůstatku s platbami,
* zajištění aktuálního stavu účtu,
* ochrana integrity finančních dat.

1. T\_STAV\_OBJEDNAVKY (automatická změna stavu objednávky)

**Popis:**  
Před vložením nebo aktualizací dat objednávky spoušť kontroluje, zda objednávka splňuje podmínky pro změnu stavu:

* Pokud je stav objednávky 1 a datum objednávky je dnešní, stav se automaticky změní na 2.

**Použití v aplikaci:**

* automatická aktualizace stavu objednávek podle pravidel,
* zajištění správné logiky workflow objednávek,
* minimalizace ruční úpravy stavů.

1. **ARC spouště (Specializace STRAVNICI)**

**Popis:**

ARC spouště zajišťují správnou specializaci entity STRAVNICI na podtypy STUDENTI a PRACOVNICI.  
Databáze kontroluje, že typ strávníka odpovídá hodnotě diskriminátoru typ\_stravnik.

**Příklady spouští:**

* ARC\_FKARC\_1\_STUDENTI
* ARC\_FKARC\_1\_PRACOVNICI

**Použití v projektu:**

* ochrana datového modelu,
* implementace specializace (ARC constraint),
* splnění bodu 20 zadání.

## Indexy

V databázové aplikaci jsou využity různé typy indexů, které zajišťují rychlejší vyhledávání dat, podporují integritu databáze a optimalizují spojení mezi tabulkami. Indexy lze rozdělit do několika kategorií:

Automaticky vytvořené indexy pomocí sql datamodeler:

1. **Primární a unikátní indexy (PK / UK)**

Primární a unikátní indexy jsou automaticky vytvářeny při definici **primárních a unikátních klíčů**. Tyto indexy:

* zajišťují **integritu dat** (není možné vložit duplicitní hodnoty),
* umožňují rychlé vyhledávání podle klíčových sloupců.

Příklady použití:

* Primární klíče: adresy.id\_adresa, jidla.id\_jidlo, objednavky.id\_objednavka, stravnici.id\_stravnik
* Unikátní klíče: alergie.produkt, dietni\_omezeni.nazev, stravnici.email

1. **Indexy pro spojení tabulek (cizí klíče / FK)**

Pro sloupce, které se často používají při JOIN operacích, jsou vytvořeny indexy. Tyto indexy nejsou klíče samy o sobě, ale výrazně zrychlují dotazy, které filtrují nebo spojují tabulky.

**Příklady a důvody použití:**

|  |  |  |
| --- | --- | --- |
| Index | Tabulka . Sloupec | Proč se používá |
| obj\_str\_fk\_idx | objednavky.id\_stravnik | Rychlé spojení objednávek s uživateli |
| obj\_stav\_fk\_idx | objednavky.id\_stav | Rychlé filtrování objednávek podle stavu |
| plat\_str\_fk\_idx | platby.id\_stravnik | Zrychlení zpracování plateb podle strávníka |
| jdl\_menu\_fk\_idx | jidla.id\_menu | Spojení jídel s menu |
| slozky\_jdl\_fk\_idx | slozky\_jidla.id\_jidlo | Spojení složek jídel s jídly |
| jdl\_slozk\_fk\_idx | slozky\_jidla.id\_slozka | Spojení složek jídel se složkami |
| omez\_str\_fk\_idx | stravnici\_omezeni.id\_omezeni | Spojení omezení strávníků s typy diet |
| str\_omez\_fk\_idx | stravnici\_omezeni.id\_stravnik | Spojení omezení s konkrétním strávníkem |
| alerg\_str\_fk\_idx | stravnici\_alergie.id\_alergie | Spojení alergií se strávníky |
| str\_alerg\_fk\_idx | stravnici\_alergie.id\_stravnik | Spojení alergií s konkrétním strávníkem |
| st\_trid\_fk\_idx | studenti.id\_trida | Spojení studentů s třídou |
| st\_str\_fk\_idx | studenti.id\_stravnik | Spojení studentů s daty strávníka |
| pr\_pozice\_fk\_idx | pracovnici.id\_pozice | Spojení pracovníků s pozicemi |
| pr\_str\_fk\_idx | pracovnici.id\_stravnik | Spojení pracovníků s daty strávníka |

**Důvody použití:**

* Zrychlení dotazů, které filtrují podle cizích klíčů,
* Optimalizace JOIN operací,
* Zlepšení výkonu při generování reportů, výpisů a statistik.

1. **Indexy na neklíčových sloupcích**

V naší databázi jsou neklíčové indexy primárně využity pro cizí klíče (viz tabulku výše). V případě potřeby by bylo možné vytvořit indexy i na dalších sloupcích, které se často používají ve WHERE nebo ORDER BY dotazech pro další optimalizaci výkonu.

## Sekvence

V naší databázové aplikaci jsou sekvence použity pro **automatické generování unikátních identifikátorů** (ID) u hlavních tabulek. To umožňuje, aby každý nový záznam měl jedinečný primární klíč, aniž by uživatel nebo aplikace museli zadávat ID ručně.

Sekvence jsou implementovány následovně:

* **Vytvoření sekvence** – každá sekvence je vytvořena příkazem CREATE SEQUENCE s nastavením START WITH 1 a INCREMENT BY 1.
* **Volání sekvence** – při vložení nového záznamu je hodnota ID automaticky doplněna pomocí NEXTVAL sekvence ve **triggeru**.

**Používané sekvence a jejich implementace**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabulka** | **Sloupec** | **Název sekvence** | **Popis** | **Implementace** |
| adresy | id\_adresa | s\_adr | Unikátní ID adresy | Trigger t\_adr\_id doplňuje ID při vložení |
| alergie | id\_alergie | s\_aler | Unikátní ID alergie | Trigger t\_aler\_id |
| jidla | id\_jidlo | s\_jid | Unikátní ID jídla | Trigger t\_jid\_id |
| logy | id\_log | s\_log | Unikátní ID logu | Trigger t\_log\_id |
| menu | id\_menu | s\_menu | Unikátní ID menu | Trigger t\_menu\_id |
| objednavky | id\_objednavka | s\_obj | Unikátní ID objednávky | Trigger t\_obj\_id |
| dietni\_omezeni | id\_omezeni | s\_omez | Unikátní ID typu dietního omezení | Trigger t\_omez\_id |
| platby | id\_platba | s\_plat | Unikátní ID platby | Trigger t\_plat\_id |
| pozice | id\_pozice | s\_poz | Unikátní ID pracovní pozice | Trigger t\_poz\_id |
| slozky | id\_slozka | s\_sloz | Unikátní ID složky jídel | Trigger t\_sloz\_id |
| soubory | id\_soubor | s\_soub | Unikátní ID souboru | Trigger t\_soub\_id |
| stavy | id\_stav | s\_stav | Unikátní ID stavu objednávky | Trigger t\_stav\_id |
| stravnici | id\_stravnik | s\_str | Unikátní ID strávníka | Trigger t\_str\_id |
| tridy | id\_trida | s\_trida | Unikátní ID třídy | Trigger t\_trida\_id |

**Implementace sekvencí**

1. **Vytvoření sekvence:**

CREATE SEQUENCE s\_obj START WITH 1 INCREMENT BY 1 NOCACHE;

1. **Volání sekvence ve triggeru:**

:NEW.id\_objednavka := s\_obj.NEXTVAL;

Tímto způsobem je zajištěno, že každá vložená objednávka automaticky získá jedinečný identifikátor.

## Systémový katalog

Pro zobrazení vlastností použitých databázových objektů aplikace využívá systémový katalog Oracle. Výpis obsahuje názvy tabulek, sloupce, primární a cizí klíče, indexy, pohledy, triggery, procedury, funkce a sekvence. Pomocí těchto dotazů lze snadno získat přehled o všech objektech v databázi.

Přehled vybraných vlastností databázových objektů

1. Názvy tabulek

SELECT table\_name

FROM user\_tables;

1. Povolení NULL hodnot

SELECT column\_name, nullable

FROM user\_tab\_columns

WHERE table\_name = 'STRAVNICI';

1. Primární klíče

SELECT constraint\_name, table\_name

FROM user\_constraints

WHERE constraint\_type = 'P’ AND constraint\_name NOT LIKE 'BIN$%';

1. Cizí klíče

SELECT constraint\_name, table\_name

FROM user\_constraints

WHERE constraint\_type = 'R' AND constraint\_name NOT LIKE 'BIN$%';

1. Indexy

SELECT index\_name, table\_name, uniqueness

FROM user\_indexes;

1. Pohledy

SELECT view\_name

FROM user\_views;

1. Triggery

SELECT trigger\_name, table\_name, triggering\_event

FROM user\_triggers;

1. Procedury

SELECT object\_name

FROM user\_objects

WHERE object\_type = 'PROCEDURE';

1. Funkce

SELECT object\_name

FROM user\_objects

WHERE object\_type = 'FUNCTION';

1. Sekvence

SELECT sequence\_name

FROM user\_sequences;

## Hierarchický dotaz

Datová vrstva aplikace využívá hierarchický dotaz pro zobrazení struktury nabídky jídel. Hierarchie je tvořena úrovněmi **menu → jídla → složky**, což odpovídá reálnému fungování školní jídelny. Hierarchický dotaz je realizován pomocí klauzule CONNECT BY, přičemž výsledky jsou využity v aplikaci pro přehledné zobrazení složení jídel a jejich návazností.

SELECT

LPAD(' ', LEVEL \* 2) || nazev AS hierarchie,

typ,

LEVEL AS uroven

FROM (

SELECT

m.id\_menu AS id,

NULL AS parent\_id,

m.nazev AS nazev,

'MENU' AS typ

FROM menu m

UNION ALL

SELECT

j.id\_jidlo AS id,

j.id\_menu AS parent\_id,

j.nazev AS nazev,

'JÍDLO' AS typ

FROM jidla j

UNION ALL

SELECT

s.id\_slozka AS id,

sj.id\_jidlo AS parent\_id,

s.nazev AS nazev,

'SLOŽKA' AS typ

FROM slozky s

JOIN slozky\_jidla sj ON sj.id\_slozka = s.id\_slozka

)

START WITH parent\_id IS NULL

CONNECT BY PRIOR id = parent\_id

ORDER SIBLINGS BY nazev;

**Ukázka výsledku (výpis)**

Menu Oběd

Řízek

Kuřecí maso

Strouhanka

Olej

Bramborová kaše

Brambory

Mléko

Máslo

# Aplikace

Dále je uveden podrobnější popis samotné aplikace.

* **Úvod aplikace:**

Úvodním bodem aplikace pro všechny uživatele je vítací stránka, která poskytuje základní informace o provozní době a adrese naší školní jídelny. Pro přístup k dalším údajům a funkcionalitě aplikace je nutné se přihlásit zadáním e‑mailové adresy do pole **„Uživatelské jméno“** a hesla do pole **„Heslo“**. Noví uživatelé si mohou vytvořit účet pomocí tlačítka **„Vytvořit nový účet“**.

Obsah obrázku text, snímek obrazovky, Operační systém, software

Obsah generovaný pomocí AI může být nesprávný.

* **Registrační stránka:**

Po kliknutí na tlačítko **„Vytvořit nový účet“** se otevře stránka určená pro registraci nového uživatele.

Obsah obrázku text, snímek obrazovky, software, Operační systém

Obsah generovaný pomocí AI může být nesprávný.

Obsah obrázku text, snímek obrazovky, software, Operační systém

Obsah generovaný pomocí AI může být nesprávný.

V **levé části obrazovky** se nachází formulář pro vyplnění základních osobních údajů, jako je jméno, příjmení, e‑mailová adresa a adresa bydliště. Zde se také nastavuje status uživatele – **Pracovník** nebo **Student** (výchozí nastavení je Pracovník).

V **pravé části obrazovky** lze pomocí tlačítka **„Vybrat fotografii“** nahrát profilovou fotografii. Pod ní se nachází pole pro zadání a potvrzení hesla.

Pokud je zvolen status **Pracovník**, zobrazí se v pravé části navíc combobox pro výběr pracovní pozice a pole pro zadání telefonního čísla. V případě, že je vybrán status **Student**, combobox umožňuje výběr třídy a pomocí kalendáře lze zadat datum narození.

Po vyplnění údajů se pomocí tlačítka **„Registrovat“** provede registrace uživatele. Zrušit registraci a vrátit se na úvodní stránku lze prostřednictvím tlačítka **„Zpět na přihlášení“**.

* **Hlavní obrazovka:**

Hlavní obrazovkou aplikace je uživatelský profil. V systému existují dva typy profilů – **profil běžného uživatele** a **profil administrátora**.

Obsah obrázku text, snímek obrazovky, software, Operační systém

Obsah generovaný pomocí AI může být nesprávný.

Obsah obrázku text, snímek obrazovky, Webové stránky, software

Obsah generovaný pomocí AI může být nesprávný.

Obsah obrázku text, snímek obrazovky, Operační systém, software

Obsah generovaný pomocí AI může být nesprávný.

Profil běžného uživatele obsahuje jeho profilovou fotografii, kterou lze kdykoliv změnit pomocí tlačítka **„Změnit fotografii“**, dále status uživatele, třídu nebo pracovní pozici, osobní údaje a přehled alergií a dietních omezení.

Pokud je uživatel **pracovníkem**, může si alergie a dietní omezení nastavovat nebo upravovat sám – stačí zaškrtnout odpovídající políčka a kliknout na tlačítko **„Uložit změny“**. V případě, že je uživatel **studentem**, tyto změny provádět nemůže a veškeré úpravy týkající se dietních omezení musí řešit prostřednictvím administrátora.

Součástí každého profilu je také informace o aktuálním zůstatku na účtu a možnost tento zůstatek navýšit. K tomu slouží tlačítko **„Doplnit zůstatek“**. Po jeho stisknutí se otevře modální okno pro zadání částky. Kliknutím na **„Potvrdit“** se částka připíše, tlačítko **„Zamítnout“** okno zavře bez provedení změny.

**Obsah obrázku text, snímek obrazovky, Písmo

Obsah generovaný pomocí AI může být nesprávný.**

Každý profil obsahuje také tlačítko **„Odhlásit se“**, které umožňuje odhlášení z aplikace a návrat na úvodní přihlašovací stránku.

Funkcionalita dalších tlačítek v profilu bude popsána v následujících bodech.

Obsah obrázku text, snímek obrazovky, Operační systém, software

Obsah generovaný pomocí AI může být nesprávný.

**Profil administrátora**

Profil administrátora se od běžného profilu liší přítomností tlačítka **„Admin panel“**. Po jeho kliknutí se otevře administrátorský panel, který umožňuje kompletní správu dat a celého systému.

* **Práce s databází (zobrazení dat):**

**Obsah obrázku text, snímek obrazovky, Webová stránka, software

Obsah generovaný pomocí AI může být nesprávný.**

Práce s daty v aplikaci se provádí prostřednictvím **admin panelu**. Pomocí comboboxu v horní části lze vybrat kategorii dat, která chce administrátor zobrazit (např. Strávníci, Pracovníci, Menu, Jídla, Složky apod.). Kritéria výběru je možné dále upřesnit pomocí seznamu umístěného pod comboboxem.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Obsah generovaný pomocí AI může být nesprávný.

V pravé části obrazovky se nachází seznam položek odpovídajících zvolené kategorii. Kliknutím na konkrétní položku ji administrátor označí a může s ní provádět další operace, jako je **úprava** nebo **odstranění**. Tyto akce se provádějí pomocí tlačítek **„Upravit“** a **„Smazat“** umístěných pod seznamem.

Obsah obrázku text, snímek obrazovky, Operační systém

Obsah generovaný pomocí AI může být nesprávný.

Pokud je v comboboxu vybrána kategorie **Studenti** nebo **Pracovníci**, může administrátor pomocí tlačítka **„Přidat“** vytvořit nový uživatelský záznam a zadat jeho základní údaje.

Obsah obrázku text, snímek obrazovky, Webové stránky, Webová stránka

Obsah generovaný pomocí AI může být nesprávný.

Tlačítko **„Přepnout“** umožňuje administrátorovi otevřít profil libovolného uživatele a zobrazit jeho detailní informace. Z profilu se lze vrátit zpět do admin panelu pomocí tlačítka **„Návrat do adminu“**.

**Obsah obrázku text, snímek obrazovky, software, Webová stránka

Obsah generovaný pomocí AI může být nesprávný.**

Tlačítko **„Systémový katalog“** umožňuje administrátorovi nahlédnout přímo do databázového systému. Tento náhled je pouze pro čtení a neumožňuje provádět žádné změny. Pomocí tlačítka **„Zavřít“** se administrátor vrátí zpět do admin panelu.

Pro návrat z admin panelu do vlastního profilu slouží tlačítko **„Zpět do profilu“**.

## Menu:

Aktuální nabídku jídel lze zobrazit pomocí tlačítka **„Zobrazit menu“** v levém panelu uživatelského profilu.

Obsah obrázku text, snímek obrazovky, Webová stránka, Webové stránky

Obsah generovaný pomocí AI může být nesprávný.

Uživatel může v comboboxu zvolit typ menu a stisknutím tlačítka **„Filtrovat“** zobrazit pouze odpovídající položky. V pravé části obrazovky se následně zobrazí seznam dostupných menu, která lze rozkliknout pro detailní přehled jednotlivých jídel.

Návrat zpět do profilu zajišťuje tlačítko **„Zpět do profilu“**.

* **Objednávaní jídel:**

Objednávku lze vytvořit pomocí tlačítka **„Vytvořit objednávku“** v levém panelu uživatelského profilu. Obsah obrázku text, snímek obrazovky, design

Obsah generovaný pomocí AI může být nesprávný.

Na stránce objednávky si uživatel v comboboxu zvolí typ menu (snídaňové nebo obědové) a tlačítkem **„Filtrovat“** zobrazí odpovídající nabídku jídel.

Jednotlivé položky menu mohou být barevně označeny:

* **žlutě**, pokud jídlo neodpovídá dietním omezením uživatele,
* **červeně**, pokud obsahuje složku, na kterou má uživatel alergii.

Jídla se do objednávky přidávají pomocí tlačítka **„Přidat do objednávky“**.

V pravé části obrazovky se zadává datum vyzvednutí objednávky – nelze zvolit víkendy, minulá ani aktuální data. Zde je také možné vidět položky v košíku, případně je odebrat tlačítkem **„Odebrat“**, zkontrolovat celkovou cenu, vybrat způsob platby a přidat poznámku.

Pokud uživatel zvolí platbu kartou, zobrazí se modální okno pro zadání údajů.

Obsah obrázku text, snímek obrazovky, Operační systém, číslo

Obsah generovaný pomocí AI může být nesprávný.

Při volbě platby z účtu se částka odečte z konta, pokud je na něm dostatek prostředků; v opačném případě se platba neprovede.

Objednávku je možné dokončit tlačítkem **„Objednat“**. Tlačítko **„Zpět do profilu“** vrací uživatele zpět do profilu bez provedení jakýchkoli změn.

* **Historie objednávek:**

Historii objednávek lze zobrazit pomocí tlačítka **„Historie objednávek“** v levém panelu uživatelského profilu.

**Obsah obrázku text, snímek obrazovky, software, Webová stránka

Obsah generovaný pomocí AI může být nesprávný.**

Na této stránce může uživatel filtrovat své objednávky podle stavu prostřednictvím comboboxu.

Kliknutím na konkrétní objednávku se v pravé části obrazovky zobrazí její detail. Uživatel zde vidí datum vyzvednutí, datum a čas vytvoření, aktuální stav, celkovou částku, případnou poznámku a seznam objednaných položek.

Pokud byla při vytvoření objednávky zvolena platba při vyzvednutí, může uživatel až do dne vyzvednutí změnit způsob platby pomocí tlačítek **„Zaplatit kartou“** nebo **„Zaplatit z účtu“**.

Obsah obrázku text, snímek obrazovky, software, Webová stránka

Obsah generovaný pomocí AI může být nesprávný.

Objednávku lze také zrušit, a to nejpozději den před plánovaným vyzvednutím, pomocí tlačítka **„Zrušit objednávku“**.

Tlačítko **„Vytisknout (PDF)“** umožňuje vygenerovat potvrzení objednávky ve formátu vhodném pro tisk.

Návrat zpět do profilu zajišťuje tlačítko **„Zpět do profilu“**.

## Použité prostředí

Projekt byl vytvořen ve vývojovém prostředí **Microsoft Visual Studio**, které slouží k vývoji, testování a ladění aplikace. Visual Studio poskytuje podporu pro programovací jazyk **C#** a platformu **.NET**, které byly použity pro implementaci aplikační logiky.

Aplikace je navržena jako **databázově orientovaný projekt** s využitím architektury **MVVM (Model–View–ViewModel)**. Tato architektura zajišťuje oddělení uživatelského rozhraní od aplikační logiky a práce s daty, což zvyšuje přehlednost, udržovatelnost a možnost dalšího rozšíření aplikace.

* **Model** obsahuje datovou strukturu aplikace a zajišťuje komunikaci s databází.
* **View** představuje uživatelské rozhraní aplikace.
* **ViewModel** zprostředkovává komunikaci mezi modelem a pohledem a zpracovává uživatelské akce.

Databázová vrstva je realizována pomocí **Oracle Database**, přičemž správa databáze a tvorba SQL dotazů probíhala v nástroji **Oracle SQL Developer**. Aplikace komunikuje s databází pomocí SQL dotazů a databázového připojení.

Použití databáze Oracle zajišťuje spolehlivé ukládání dat, podporu transakcí a vysokou úroveň zabezpečení. Kombinace technologií Visual Studio, architektury MVVM a databáze Oracle umožňuje vytvořit stabilní a přehlednou databázovou aplikaci.

## Řízení uživatelských účtů

Řízení uživatelských účtů v aplikaci je realizováno pomocí modelu **RBAC (Role-Based Access Control)**, tedy řízení přístupových práv na základě rolí uživatelů.

Uživatelé aplikace jsou evidováni v databázové tabulce **STRAVNICI**, která obsahuje základní identifikační údaje, přihlašovací informace a atributy určující oprávnění uživatele. Přístup do aplikace je umožněn pouze aktivním uživatelům.

Autentizace uživatele probíhá na základě kombinace **e-mailové adresy a hesla**. Po úspěšném přihlášení je uživateli přiřazena role, která určuje rozsah dostupných funkcí v aplikaci.

**Úrovně přístupů (Role)**

Aplikace rozlišuje několik uživatelských rolí:

* **Administrátor**
  + správa uživatelských účtů,
  + správa databázových záznamů (jídla, složky, menu),
  + přístup ke všem funkcím aplikace.
* **Pracovník a Student**
  + správa objednávek,
  + práce s nabídkou jídel,
  + omezený přístup k administrátorským funkcím,
  + zobrazení nabídky jídel,
  + vytváření objednávek,
  + náhled na vlastní objednávky a platby,
  + Pracovník má možnost editace vlastních alergii a dietní omezení, ale student nemá možnost editovat alergie a dietní omezení – tyto údaje nastavuje administrátor.

Role uživatele je uložena v atributu role tabulky **STRAVNICI**, přičemž další rozlišení uživatelů (např. student nebo pracovník) je realizováno pomocí atributu typ\_stravnik.

## Moduly

**Modul registrace a přihlášení**

Tento modul zajišťuje registraci nových uživatelů a jejich přihlašování do aplikace. Obsahuje ověření přihlašovacích údajů a správu uživatelských účtů. Bez přihlášení není možné aplikaci používat.

**Modul uživatelského profilu**

Modul uživatelského profilu slouží ke správě osobních údajů uživatele. Uživatel zde může upravovat své údaje, zadávat alergie nebo dietní omezení, měnit profilovou fotografii a sledovat zůstatek na účtu.

**Modul menu**

Tento modul zobrazuje aktuální nabídku jídel. Uživatel si může menu filtrovat podle typu (např. snídaně, oběd) a zobrazit podrobnosti jednotlivých položek.

**Modul objednávek**

Modul objednávek umožňuje vytváření, správu a rušení objednávek. Obsahuje výběr jídel, volbu data vyzvednutí, způsobu platby a zobrazení historie objednávek.

**Modul plateb**

Tento modul zajišťuje práci s finančními prostředky v aplikaci. Umožňuje dobití zůstatku na účtu a platbu objednávek pomocí dostupných platebních metod.

**Modul administrace**

Administrátorský modul je určen pro správce systému. Umožňuje správu uživatelů, menu a objednávek a úpravu údajů v aplikaci.

## Formuláře

**Formulář pro registraci strávníků.**

Formulář registrace slouží k vytvoření nového uživatelského účtu. Uživatel zde zadává své osobní údaje, e-mailovou adresu a heslo. Po odeslání formuláře je účet uložen do databáze a uživatel se může přihlásit do aplikace.

**Formulář přihlášení.**

Formulář přihlášení umožňuje uživateli přístup do aplikace. Obsahuje pole pro zadání e-mailu a hesla. Po úspěšném ověření údajů je uživatel přesměrován do svého profilu.

**Formulář dobití zůstatku**

Formulář dobití zůstatku umožňuje uživateli vložit peníze na svůj účet v aplikaci. Uživatel zadá částku, kterou chce dobít, a potvrdí platbu. Po úspěšném dokončení se zůstatek na účtu aktualizuje.

**Formulář vytvoření objednávky**

Formulář pro vytvoření objednávky slouží k výběru jídel z menu. Uživatel zvolí položky, datum vyzvednutí, způsob platby a může zadat poznámku k objednávce. Po odeslání formuláře je objednávka uložena do systému.

## Orientace ve zdrojovém kódu

Aplikace je navržena s využitím architektury **MVVM (Model–View–ViewModel)**, která zajišťuje přehledné oddělení uživatelského rozhraní, aplikační logiky a práce s daty. Díky této architektuře je zdrojový kód lépe strukturovaný, přehledný a snadno udržovatelný.

**App (SkolnaJidelna\App.xaml, SkolnaJidelna\App.xaml.cs)**

**Kde:** soubory App.xaml a App.xaml.cs v kořenovém adresáři projektu SkolnaJidelna.

**Proč:** vstupní bod WPF aplikace. Definuje globální zdroje (styly, konvertory), životní cyklus aplikace a obsluhu startu/ukončení. Inicializuje konfiguraci a služby před zobrazením hlavního okna.

**MainWindow (SkolnaJidelna\ MainWindow.xaml, SkolnaJidelna\ MainWindow.xaml.cs)**

**Kde:** soubory MainWindow.xaml a MainWindow.xaml.cs v kořenovém adresáři projektu SkolnaJidelna.

**Proč:** hlavní okno aplikace. Obsahuje základní navigaci a rozvržení UI, nastavuje DataContext (obvykle MainWindowViewModel), zpracovává události hlavní stránky a spouští hlavní scénáře (přihlášení, otevření profilu, přechody mezi sekcemi).

**Model**

Vrstva **Model** obsahuje datové třídy a logiku práce s databází. Tyto třídy reprezentují základní entity aplikace, jako jsou uživatelé, menu, objednávky a položky objednávek.  
Model zajišťuje načítání, ukládání a aktualizaci dat v databázi a obsahuje také databázové funkce pro výpočty a kontroly dat (např. výpočet celkové ceny objednávky nebo kontrolu stavu menu).

**Seznam tříd a view v Models:**

* **Datové třídy (entity):** Adresa, Alergie, DietniOmezeni, Jidlo, Log, Menu, Objednavka, Platba, Polozka, Pozice, Pracovnik, Slozka, SlozkaJidlo, Soubor, Stav, Stravnik, StravnikAlergie, StravnikOmezeni, Student, Trida – reprezentují jednotlivé tabulky v databázi a jejich vztahy.
* **Pohledy (views):** VJidlaMenu, VJidlaSlozeni, VObjHistorieDetail, VObjPrehledAdmin, VPracovnikPozice, VStravnikLogin, VStudTrida, VStravnikOmezeniAlergie, VStravniciFull – poskytují předpřipravené dotazy pro zobrazení složených nebo kombinovaných dat v aplikaci.

Celkově tato složka poskytuje **přímé mapování databázových struktur do objektového modelu aplikace**, což zajišťuje konzistenci dat, jednoduchou manipulaci s nimi a snadnou integraci s vrstvou ViewModel.

**View**

Vrstva **View** představuje uživatelské rozhraní aplikace. Obsahuje jednotlivé obrazovky, formuláře a vizuální prvky, se kterými uživatel pracuje.  
View nezpracovává aplikační logiku, ale pouze zobrazuje data získaná z ViewModelu a předává uživatelské akce (např. kliknutí na tlačítka nebo odeslání formuláře) dále do aplikace.

**AdminPanel (AdminPanel.xaml.cs)**

**Konstruktor():** inicializuje komponent, nastavuje DataContext a bindingy.

**Události / button handlers:** přepínání entit, filtry, volání VM příkazů.

**AdminProfileWindow**

**Konstruktor(email):** inicializuje okno profilu admina/uživatele podle e-mailu.

**OnLoaded():** načte data, nastaví bindingy.

**CloseButton\_Click():** zavře okno.

**CreateStravnikDialog**

**Konstruktor():** inicializuje dialog pro vytvoření strávníka.

**OkButton\_Click():** validuje pole, zavře dialog s DialogResult=true.

**CancelButton\_Click():** zavře dialog s DialogResult=false.

**EditDialogWindow**

**Konstruktor():** inicializace, nastavení DataContext pro editor vlastností.

**SaveButton\_Click():** uloží změny přes VM a zavře dialog.

**CancelButton\_Click():** zavře dialog bez uložení.

**LoginWindow**

**Konstruktor():** inicializace, nastavení DataContext, příprava polí pro login.

**LoginButton\_Click():** provede přihlášení (validace, dotaz do DB), otevře profil.

**CancelButton\_Click():** zavře okno.

**MenuWindow**

**Konstruktor():** inicializuje okno menu, nastaví MenuViewModel.

**RefreshButton\_Click():** načte menu s filtrem.

**CloseButton\_Click():** zavře okno.

**OrderWindow**

**Konstruktor():** inicializuje okno objednávky, nastaví OrderViewModel.

**ConfirmButton\_Click():** potvrzení objednávky, spuštění platby / uložení.

**CancelButton\_Click():** zavře okno.

**AddItem\_Click() / RemoveItem\_Click():** přidá/odebere položku z objednávky.

**OrderHistoryWindow**

**Konstruktor():** inicializace okna historie, nastaví OrderHistoryViewModel.

**FilterChanged():** aplikuje filtr podle data/stavu.

**CloseButton\_Click():** zavře okno.

**PaymentWindow**

**Konstruktor(PaymentMethod):** inicializuje okno platby, zobrazuje panel (Card / Account / Cash).

**PayButton\_Click():** potvrzení platby, zavře s DialogResult=true.

**CancelButton\_Click():** zrušení platby, zavře s DialogResult=false.

**RechargeBalanceWindow**

**Konstruktor(email):** inicializuje okno doplnění zůstatku, pamatuje e-mail uživatele.

**ConfirmButton\_Click():** validuje částku, aktualizuje zůstatek / zapíše platbu, otevře PaymentWindow, zavře s DialogResult=true.

**CancelButton\_Click():** zavře dialog s DialogResult=false.

**StravnikListView**

**Konstruktor():** inicializace seznamu strávníků, nastavení DataContext.

**OnSelectionChanged():** zpracuje výběr položky, otevře detail/editaci.

**SystemovyKatalogWindow**

**Konstruktor():** inicializace okna systémového katalogu.

**RefreshButton\_Click():** načte data katalogu.

**CloseButton\_Click():** zavře okno.

**UserProfileWindow**

**Konstruktor(email):** inicializace okna profilu uživatele, propojení s UserProfileViewModel.

**ChangePhoto\_Click():** výběr souboru a změna fotografie přes VM.

**SaveAlergies\_Click():** uloží změny alergií/diet přes VM.

**Recharge\_Click():** otevře RechargeBalanceWindow pro doplnění zůstatku.

**ViewModel**

Vrstva ViewModel slouží jako prostředník mezi View a Modelem. Obsahuje aplikační logiku, reaguje na akce uživatele, připravuje data pro zobrazení a zajišťuje volání databázových operací (např. vytvoření objednávky, historie objednávek, kontrola zůstatku).

**Třída BaseViewModel**

**Popis:** základní ViewModel s implementací INotifyPropertyChanged.

**OnPropertyChanged(name):** upozorní UI na změnu vlastnosti.

**Třída MainWindowViewModel**

**Popis:** logika hlavního okna: navigace a stav uživatele.

**Metody/Commandy:** správa aktuálních stránek a přepínání view.

**Třída AdminViewModel**

**Popis:** panel administrátora: načítání entit, filtry, editace a ukládání.

**Metody:**

* LoadEntityTypesAsync() – inicializace typů entit
* LoadItemsForSelectedEntityAsync() – načtení položek podle filtru
* OnSelectedItemChanged(item) – příprava vlastností pro editaci
* SaveAsync() – uloží změny vybrané položky
* DeleteAsync() – odstraní položku a související entity
* LoadClassesAsync(), LoadPositionsAsync(), LoadFoodCategoriesAsync(), ... – načtení hodnot filtrů
* CreateEmptyStravnikAsync(...), AddNewStravnikAsync(owner) – vytváření nového strávníka
* GetIdValue(entity) – získání ID jako řetězce

**Třída AdminProfileViewModel**

**Popis:** profil administrátora, načítání dat podle e-mailu, agregovaná informace.

**Metody/Commandy:** načítání/aktualizace profilu, zpracování UI příkazů.

**Třída UserProfileViewModel**

**Popis:** profil uživatele (student/pracovník), načítání dat, foto, alergie/diety.

**Metody:**

* LoadAsync(email) – načtení profilu
* SaveChangesAsync() – uloží změny alergií a diet
* ChangePhotoAsync() – změna fotografie
* CreateInitialsImage(firstName, lastName) – obrázek s iniciály
* CreateInitialsImageCore(firstName, lastName) – jádro generování obrázku
* TableExists(db, tableName) – kontrola existence tabulky
* OnPropertyChanged(name) – upozornění UI přes Dispatcher

**Třída OrderViewModel**

**Popis:** tvorba objednávky, výběr jídel, výpočet ceny, volba platby.

**Metody:** přidání/odebrání položek, potvrzení/storno, načtení menu.

**Třída OrderHistoryViewModel**

**Popis:** historie objednávek uživatele, filtry podle dat/stavu.

**Metody:** načtení historie, detailní zobrazení, aktualizace.

**Třída MenuViewModel**

**Popis:** správa menu (seznam, filtr podle typu, detaily).

**Metody:** načtení menu, výběr menu, aktualizace view.

**Třída SystemovyKatalogViewModel**

**Popis:** systémový katalog, zobrazení slovníků a parametrů.

**Metody:** načtení sekcí, zpracování výběru, filtrování.

**Třída JidloViewModel**

**Popis:** správa jídel, složení a kategorie.

**Metody:** načtení jídel, výběr položky, editace složení.

**Třída StravnikListViewModel**

**Popis:** seznam strávníků, vyhledávání a filtrování.

**Metody:** načtení seznamu, filtr, výběr položky.

**Třída RegisterViewModel**

**Popis:** registrace nového uživatele.

**Metody:** validace vstupu, odeslání dat, zpracování výsledku.

**Třída PropertyViewModel**

**Popis:** wrapper pro editovatelnou vlastnost v Property editoru.

**Konstruktor:** (name, type, value, onChanged)

**Vlastnosti:** EditorType, ItemsSource – ovládají typ editoru (text, datum, seznam, checkboxy).

**Třída EditableProperty**

**Popis:** model editovatelné vlastnosti pro editory.

**Vlastnosti/Metody:** jméno, typ, hodnota, validátory.

**Třída** **PropertyEditorSelector**

**Popis:** výběr editoru pro vlastnost podle typu/metadat.

**Metody:** určení vhodného kontrolu pro editaci.

**Třída ItemViewModel**

**Popis:** položka seznamu s entitou a krátkou anotací.

**Konstruktor:** (entity, summary)

**Třídy SelectableAlergie / SelectableDiet**

**Popis:** položka seznamu alergií/diét s možností výběru.

**Vlastnosti:** Alergie/Diet, IsSelected.

**Třída EntityTypeDescriptor**

**Popis:** popis typu entity pro admin panel.

**Vlastnosti:** Name, EntityType, LoaderAsync

**LoaderAsync():** načtení seznamu objektů vybraného typu s filtry.

**Data**

Obsahuje DbContext (AppDbContext) a konfiguraci přístupu k databázi (řetězec připojení, logování EF Core).

**Třída AppDbContext (Data)**

**Dědičnost:** DbContext (EF Core)

**Účel:**

* Centrální připojení k Oracle databázi.
* Mapování modelů (Models) na tabulky a pohledy v databázi.
* Poskytuje DbSet<> pro CRUD operace a LINQ dotazy.
* Konfigurace logování SQL a nastavení připojení.

**Klíčové prvky a metody:**

* **DbSet<>**: Stravnik, Student, Pracovnik, Adresa, Objednavka, Platba, Menu, Jidlo, Pozice, Trida a pohledy (VStravnikLogin, VStudTrida, VPracovnikPozice, VJidlaSlozeni, VObjHistorieDetail) – pro dotazy a změny dat.
* **OnConfiguring(optionsBuilder)**: načítá connection string z appsettings.json, povoluje logování EF, nastavuje UseOracle.
* **OnModelCreating(modelBuilder)**: konfigurace mapování tabulek/pohledů, primárních a složených klíčů, sekvence S\_STR.NEXTVAL pro Stravnik.IdStravnik, konverze bool → int, ToView().HasNoKey() pro pohledy.
* **Database.GetDbConnection()**: přístup k nízkoúrovňovému DbConnection pro raw SQL a uložené procedury.
* Standardní EF metody (Add, Update, Remove, SaveChanges) pro sledování a ukládání změn.

**Helpers**

Obsahuje **pomocné třídy a utility** pro uživatelské rozhraní a ViewModel: například RelayCommand pro MVVM příkazy, konvertory, metody pro formátování a validaci dat.

**Neobsahuje obchodní logiku**, pouze znovupoužitelné nástroje a funkce.

Usnadňuje opakované operace a zvyšuje čitelnost kódu ve View a ViewModel.

**Třída InverseBoolConverter**

**Popis:** Konvertor pro XAML, který invertuje logickou hodnotu. Zjednodušuje binding viditelnosti a stavů bez další logiky ve ViewModel.

**Convert:** přijímá bool, vrací jeho negaci; pokud je null u cílového bool?, vrací true („zobrazit při absenci dat“).

**ConvertBack:** invertuje zpět pro dvoucestný binding, jinak vrací Binding.DoNothing.

**Třída PathToImageConverter**

**Popis:** Konvertor řetězce cesty k souboru na BitmapImage pro XAML Image.

**Convert:** vytváří BitmapImage z absolutní cesty, používá OnLoad pro uvolnění souboru; při chybě vrací null, nezpůsobuje výjimku v UI.

**ConvertBack:** není podporováno, vrací Binding.DoNothing.

**Třída PasswordBoxAssistant**

**Popis:** Pomocník pro dvoucestnou vazbu PasswordBox k vlastnosti ve ViewModel. Řeší omezení WPF (Password není dependency property).

**BindPasswordProperty:** zapíná/vypíná synchronizaci s vlastností.

**BoundPasswordProperty:** uchovává aktuální heslo (string), synchronizuje se s PasswordBox.Password.

**UpdatingPasswordProperty:** interní flag proti rekurzivní aktualizaci při programovém nastavení hesla.

**Set/GetBindPassword:** nastavení/čtení flagu vazby na elementu.

**Set/GetBoundPassword:** nastavení/čtení aktuálního hesla na elementu.

**OnBindPasswordChanged:** připojuje/odpojuje PasswordChanged handler při zapnutí/vypnutí vazby.

**OnBoundPasswordChanged:** reaguje na změny hesla z VM, aktualizuje PasswordBox.Password bez cyklu.

**PasswordChanged:** reaguje na vstup uživatele a zapisuje do BoundPassword pro VM.

**Třída RelayCommand**

**Popis:** Univerzální implementace ICommand pro MVVM. Umožňuje definovat příkazy přes delegáty bez samostatných tříd.

**Konstruktor:** přijímá delegát Execute a volitelný CanExecute pro kontrolu dostupnosti příkazu.

**CanExecute:** vrací, zda příkaz může být proveden; ovlivňuje stav tlačítek v UI.

**Execute:** spouští předanou logiku příkazu.

**CanExecuteChanged:** propojuje s CommandManager.RequerySuggested pro automatickou aktualizaci UI.

**RaiseCanExecuteChanged:** manuálně spustí přepočet dostupnosti příkazu, např. při změně SelectedItem.

**Services**

Poskytuje **servisy**, které abstraktně zpracovávají infrastrukturu a interakci s prostředím: například IFileDialogService a jeho implementace pro výběr souborů, notifikační služby apod.

Používány ve ViewModel přes **dependency injection (DI)** pro testovatelnost a izolaci, bez přímého volání UI/OS API.

Slouží k centralizovanému a čistému přístupu k externím funkcím a prostředkům systému.

**Rozhraní IfileDialogService**

**Popis:** Abstrakce nad systémovým dialogem výběru souboru.

**OpenFileDialog(filter):** otevře dialog, vrací absolutní cestu nebo null. VM získává cestu bez přímého volání WPF.

**Třída FileDialogService**

**Popis:** Implementace IFileDialogService přes OpenFileDialog.

**OpenFileDialog(filter):** zobrazuje dialog s filtrem, vrací vybranou cestu. Umožňuje nahradit mocky v testech.

**Rozhraní IWindowService**

**Popis:** Služba pro navigaci a otevírání oken. Odstraňuje logiku vytváření oken z ViewModel.

**ShowAdminProfile(adminEmail):** otevře okno profilu admina.

**ShowUserProfile(email, isPracovnik):** otevře okno profilu uživatele pro různé role.

**Třída WindowService**

**Popis:** Implementace IWindowService, otevírá okna přes Dispatcher na UI vlákně.

**ShowAdminProfile(adminEmail):** spouští AdminProfileWindow.Show() na UI vlákně.

**ShowUserProfile(email, isPracovnik):** obdobně otevírá UserProfileWindow.

**Rozhraní IStravnikRepository**

**Popis:** Repository pro práci se Stravnik (CRUD). Umožňuje DI, mocky a testovatelnost.

**GetAllAsync():** získá všechny záznamy.

**GetByIdAsync(id):** najde záznam podle klíče.

**AddAsync(entity):** přidá nový záznam.

**UpdateAsync(entity):** aktualizuje záznam.

**DeleteAsync(id):** odstraní záznam podle ID.

**Třída StravnikRepository**

**Popis:** EF Core implementace IStravnikRepository. Centralizuje EF, skryje detaily, snadná náhrada mockem.

**GetAllAsync():** AsNoTracking + ToListAsync — rychlé načtení bez trackování.

**GetByIdAsync(id):** FindAsync pro přímý PK hledání.

**AddAsync(entity):** AddAsync + SaveChangesAsync — bezpečná transakce přidání.

**UpdateAsync(entity):** Update + SaveChangesAsync — jednotný vzor aktualizace.

**DeleteAsync(id):** FindAsync + Remove + SaveChangesAsync — bezpečné odstranění, pokud existuje.

## Změny z původního návrhu

## Změny v logickém a konceptuálním modelech

Entita STRAVNIK má nový atributy

* Email a heslo – pro přihlášení strávníků do účtů.
* Role – slouží pro sledovaní přihlašovacích prav strávníků (user, admin).
* Aktivita – sledovaní aktivity strávníka, pokud se strávník nepřihlásí do účtu déle než 3 měsíce, nebude se moci do účtu přihlásit.

Entita STUDENT má změnu:

* Rok narození – atribut věk není vhodný pro dlouhodobé sledování věku studenta, protože by bylo nutné jej každý rok aktualizovat. Proto je vhodnější evidovat rok narození.

Entita ADRESA má změnu:

* Město – atribut obec nemá smysl, protože všichni strávníci bydlí v Pardubickém kraji, avšak konkrétní město se může lišit (např. Pardubice, Přelouč).

Entita TYDEN je odstraněna:

* Entita neměla smysl, protože se menu mění podle typu (snídaně nebo oběd) a času a zároveň se liší podle jednotlivých týdnů, což by bylo složité na evidenci a správu.

Entita SOUBOR byla doplněna:

* podle zadání bylo potřeba evidovat soubory (např. textové nebo pdf soubory a fotografie) přiřazené ke strávníkovi.

Entita LOG byla doplněna:

* Na základě zadání bylo pro zajištění evidence a kontroly dat v tabulkách nutné sledovat jejich změny.

Relace jidla sama na sebe

* Přidán cizí klíč parent\_jidlo v tabulce jidla, odkazující na id\_jidlo téže tabulky.
* Umožňuje definovat hierarchii nebo varianty jídel (např. hlavní jídlo a jeho podvarianty), což zjednodušuje práci s menu a zobrazení složených jídel v aplikaci.

Volitelný FK id\_adresa u stravnici

* Cizí klíč id\_adresa v tabulce stravnici není povinný (NULLABLE).
* Administrátor může předem vložit uživatele (strávníky) školy do databáze jako neaktivní. Strávník se může později registrovat a při registraci se doplní jeho adresa a další údaje, pokud už je v databázi. Tím se zajišťuje flexibilita při správě uživatelů a minimalizuje duplicitní zadávání dat.

## Změny ve fyzickém modelu dat

Odstranění zbytečných indexů:

Databáze obsahovala zbytečné a nefunkční indexy nad identifikátory tabulek, proto byly tyto indexy odstraněny.

Závěr

Naše aplikace slouží ke objednávání a sledování objednávek ve školní jídelně. Obsahuje informace o strávnících, jejich objednávkách, nabízených jídlech a také alergiích a dietních omezeních jednotlivých návštěvníků. Každé objednávce odpovídá právě jeden způsob platby, a to hotovost, platební karta nebo platba z účtu. Databáze je navržena v souladu se zásadami normalizace, které minimalizují nadbytečnost dat a umožňují efektivní správu pomocí SQL dotazů. Aplikace je realizována v souladu s architekturou MVVM.

Při práci na tomto projektu jsme si výrazně zlepšili schopnosti v oblasti návrhu databází a osvojili jsme si tvorbu pohledů, procedur, funkcí. Prohloubili jsme své znalosti nástrojů Oracle SQL Developer Data Modeler a Oracle SQL Developer. Kromě toho jsme se seznámili s architekturou MVVM a zlepšili jsme své dovednosti v programovacím jazyce C#. Tyto znalostí a dovednosti budou pro nás velmi přínosné při práci na dalších databázových a softwarových projektech.

Každý projekt lze dále rozvíjet a ani tento není výjimkou. Prostor pro zlepšení vidíme především v posílení zabezpečení aplikace a v jejím rozšíření na další platformy. Do budoucna by bylo vhodné vytvořit nejen desktopovou, ale také mobilní aplikaci, která by strávníkům umožnila pohodlnější a dostupnější používání celého systému.

**Přílohy**

1. **Skripty SQL** - všechny skripty pro vytvoření databáze, tabulek, relací, triggerů a počátečních dat.
2. **Zdrojové kódy aplikace** – kompletní zdrojový kód aplikace, včetně XAML, ViewModelů, grafiky a dalších podpůrných souborů. Samostatně spustitelná aplikace.
3. **Samostatně spustitelná aplikace** – pokud je k dispozici, nebo návod na její zprovoznění.