

Gymnázium, Praha 6, Arabská 14

Obor programování



Ročníková práce

Elektronická evidence potravin

Duben 2022

Vojtěch Franc 2. E

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 29. 4. 2022

vlastnoruční podpis autora

Anotace

V této práci se zabývám tvorbou programu pomocí JavaFx pro evidenci potravin s využitím jejich čárových kódů. Hlavní téma práce je fungování synchronizace databáze, která může fungovat i v offline režimu, a do které může přistupovat několik zařízení z různých platforem najednou. V práci se tak zabývám fungováním čárových kódů a jejich generováním pro nové potraviny s účelem omezení možných kolizí vygenerovaných kódů.

Klíčová slova

Evidence potravin, čárové kódy, EAN-13, generování čárových kódů, synchronizace databázového systému.

Annotation

In this thesis, I am developing a program using JavaFx to keep track of food items using their barcodes. The main topic of the thesis is the functioning of the database synchronization, which can also work in offline mode, and which can be accessed by several devices from different platforms at the same time. I am also dealing with the functioning of barcodes and their generating for new foods in order to reduce possible collisions of the generated codes.

Obsah

1. Úvod	6
2. Návod k instalaci	7
3. Grafické rozhraní	8
4. Potravina	11
5. Zjednodušený diagram programu	12
6. Synchronizace	13
7. EAN-13 standard	14
7.1. EAN kód potravin	14
7.2. Generování vlastních kódů	15
8. Ukázky kódu	16
9. Tisk etiket	17
10. Závěr	18
10.1. Možnosti rozšíření	18
12. Bibliografie	19
13. Seznam obrázků	20

1. Úvod

V domácnostech se vyplývá zhruba třetina nakoupených potravin. Výroba potravin se na globální produkci oxidu uhličitého podílí téměř z desetiny. Na vině je mnoho faktorů včetně nevhodně navržených obalů nebo nevhodného skladování. Mimo tyto faktory jde většinou o překročení minimální doby trvanlivosti způsobené nedostatkem přehledu o skladovaných potravinách, což v mnoha případech doprovází také nakupování redundantního množství těchto potravin. (1) (2)

V rámci snahy o dosažení klimatických závazků a omezení plýtvání, které přináší také etické problémy, by bylo vhodné věnovat pozornost hledání řešení i v tomto odvětví. Proto jsem se rozhodl vytvořit tento evidenční systém, který se zaměřuje primárně na použití v domácnostech. Myšlenkou vytvořit evidenční systém jsem se inspiroval z části existujícím systémem evidence mražených potravin v podobě papírového sešitu.

Tento systém není určen pro evidenci potravin např. v lednici, jelikož se do ní přistupuje často a není tak obtížné mít přehled o jejím obsahu. Uživatelé by tak mohli díky časové úspoře vynechávat evidenci potravin, což by snížilo relevanci celé databáze a celý systém by tak mohl získat nálepku, že není užitečný, což by mohlo mít za následek nechuť ho implementovat i v podmínkách, kde by to dávalo větší smysl. Primární využití tohoto projektu je zamýšleno pro použití v místnostech určených pro skladování trvanlivějších potravin, do kterých se tak často nechodí a ve kterých je obtížné si o potravinách udržet přehled. Pro snadné vyhledávání v databázi projekt počítá s využitím čárových kódů, kterými jsou potraviny opatřeny. Uživatel je bude načítat pomocí čtečky čárových kódů nebo pomocí kamery v online verzi této aplikace.

Přínos této aplikace bude spočívat v zprehlednění skladovaných potravin, možnosti řazení na základě data spotřeby a upozornění na procházející potraviny. Správa skladovaných potravin by také měla zredukovat nakupování přebytečných potravin, jelikož uživatel bude moci odkudkoliv do databáze přistupovat a zkontrolovat aktuální množství skladovaných potravin.

2. Návod k instalaci

I přes to, že aplikace dokáže správu potravin provádět offline, pro první přihlášení do systému vyžaduje připojení k MySQL databázi. Jelikož servery nabízející databáze s hostováním webových stránek neumožňují přístup k databázím odjinud a služby nabízející databáze pro tyto účely jsou placené, je nutné si pro správné fungování projektu vytvořit vlastní lokální databázi. Já jsem k tomu použil program Xampp. Ve správě PhpMyAdmin je nutné vytvořit novou databázi s tabulkou nazvanou uživatelé (podrobnosti nastavení tabulky viz obr.). Jelikož je databáze používána i pro webovou verzi tohoto projektu, tabulka obsahuje více polí, než počítačová verze využije. Tabulka tedy musí nutně obsahovat jen položky: id, jméno, heslo a email.

#	Název	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí	Komentáře	Další
<input type="checkbox"/> 1	id	int(11)			Ne	Žádná		AUTO_INCREMENT
<input type="checkbox"/> 2	jméno	varchar(1000)	utf8_unicode_ci		Ne	Žádná		
<input type="checkbox"/> 3	email	varchar(1000)	utf8_unicode_ci		Ne	Žádná		
<input type="checkbox"/> 4	heslo	varchar(1000)	utf8_unicode_ci		Ne	Žádná		
<input type="checkbox"/> 5	misto	varchar(1000)	utf8_unicode_ci		Ne	Žádná		
<input type="checkbox"/> 6	ikonkamista	varchar(1000)	utf8_unicode_ci		Ne	Žádná		
<input type="checkbox"/> 7	webkamera	int(11)			Ne	Žádná		
<input type="checkbox"/> 8	autozalohovani	int(11)			Ne	Žádná		
<input type="checkbox"/> 9	autoupozorninaexpiraci	varchar(30)	utf8_unicode_ci		Ne	Žádná		

Obrázek 1 Struktura databáze uživatelů

Jelikož samotná aplikace nemá funkci pro vytvoření nového uživatelského profilu (aplikace uživatele přesměruje na webovou verzi, kde to je možné), existují dvě různé cesty pro vytvoření účtu. První možností je spustit a nakonfigurovat PHP verzi projektu. Z repositáře tohoto projektu na GitHubu je nutné stáhnout EEP.zip a tento soubor v případě použití Xampp rozbalit do požadované složky (defaultně to je C:\xampp\htdocs). Upravit soubor db.php a doplnit do něj údaje o vytvořené databázi (název databáze, uživatelské jméno a heslo). Poté je možné pokračovat zadáním url adresy do internetového prohlížeče (localhost/EEP/public_html/uvod/index.html). Zde je možné kliknout na tlačítko vytvořit účet a pokračovat vyplňováním formulářů.

Druhá snazší možnost je uživatelský účet v databázi vlastnoručně vytvořit. Nejprve je nutné zadat do tabulky uživatelé nový záznam. K tomu poslouží záložka vložit. Položky jako je jméno a email lze bez problému vyplnit textem v běžném tvaru, ale heslo musí být zakódováno pomocí SHA3-512. To lze provést např. pomocí webové stránky: browserling.com/tools/sha3-hash. U položek webkamera a autozalohovani je nutné vyplnit 0 nebo 1. Poté je nutné vytvořit další tabulku, která se bude jmenovat stejně, jako je vyplněné jméno v seznamu uživatelů. Struktura tabulky je znázorněna na obrázku níže. Posledním krokem je zadání parametrů pro připojení k databázi do samotného projektu. Informace o parametrech připojení k databázi jsou v souboru DB.java, tam je nutné zadat název databáze, uživatelské jméno a heslo. Poté se bude možné do aplikace přihlásit pomocí zadaných údajů.

#	Název	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí	Komentáře	Další
<input type="checkbox"/> 1	typ	varchar(100)	utf8_general_ci		Ne	Žádná		
<input type="checkbox"/> 2	id	int(11)			Ne	Žádná		AUTO_INCREMENT
<input type="checkbox"/> 3	ean	varchar(500)	utf8_general_ci		Ano	NULL		
<input type="checkbox"/> 4	jméno	varchar(200)	utf8_general_ci		Ano	NULL		
<input type="checkbox"/> 5	kategorie	varchar(100)	utf8_general_ci		Ano	NULL		
<input type="checkbox"/> 6	spotreba	varchar(100)	utf8_general_ci		Ano	NULL		
<input type="checkbox"/> 7	mnozství	varchar(100)	utf8_general_ci		Ano	NULL		
<input type="checkbox"/> 8	jednotky	varchar(100)	utf8_general_ci		Ano	NULL		
<input type="checkbox"/> 9	poznámky	varchar(200)	utf8_general_ci		Ano	NULL		

Obrázek 2 Struktura databáze potravin

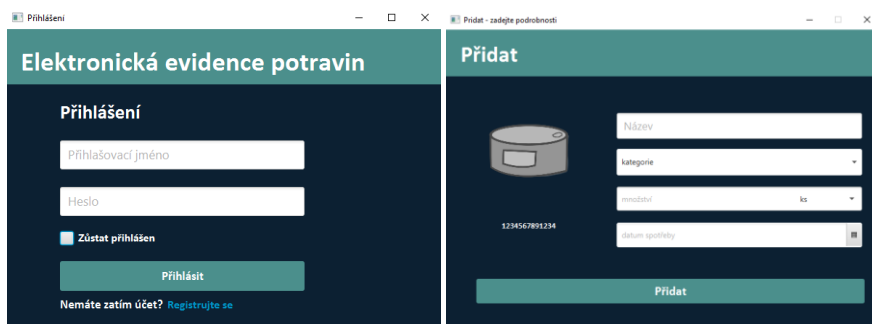
3. Grafické rozhraní

Pro tvorbu grafického prostředí jsem použil JavaFx a Scene Builder. Potraviny jsem se rozhodl reprezentovat pomocí karet s obrázkem, který se nastavuje na základě zvolené kategorie a pomáhá tak uživateli se v položkách lépe orientovat a působit srozumitelněji, než by působila tabulka se jmény a parametry. Ideální by bylo každé potravíně vytvořit samostatnou ikonku, což by bylo náročné, ale jednou z možností je po nasbírání dostatečného množství dat vytvořit seznam nejčastěji vyskytujících se potravin (podle jejich kódu), pro které by se vyplatilo vytvořit vlastní ikonky. Zbytek potravin by dále získával svou ikonku na základě kategorie. Oproti tabulce však po přidání velkého množství potravin je stávající způsob zobrazování potravin nepřehledný a počítá se s využitím čtečky čárového kódu k rychlému nalezení dané potraviny nebo jiné způsoby hledání v tabulce.



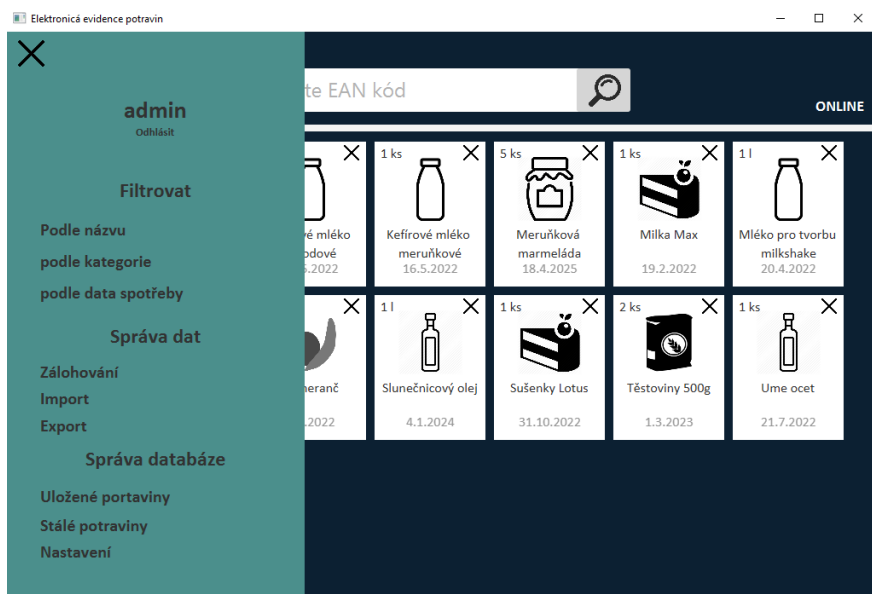
Obrázek 3 Vygenerovaná paleta barev pro tento projekt

Jedním z hlavních problémů, na které jsem v průběhu tvorby narazil, bylo omezení přístupu ke grafickým prvkům aplikace z jiných vláken. Řešením by mohlo být veškeré potřebné operace provádět jen na tom vlákně, které má ke grafickým prvkům přístup. Tímto způsobem fungovaly rané verze této aplikace. Problémem však bylo, že v průběhu komunikace s databází nebylo možné současně s UI interagovat. Podařilo se mi najít řešení použitím třídy `ScheduledService`, která umožňuje opakovaně provádět určitý požadavek v daném intervalu. Hlavní výhodou je, že komunikace s databází nepřerušuje chod hlavního grafického vlákna a prováděný kód má možnost přistupovat k prvkům uživatelského rozhraní. Hlavní okno vytváří dvě instance této třídy, první se opakuje každou vteřinu a kontroluje, zdali nedošlo ke změně obsahu. Druhá instance každých třicet vteřin kontroluje, zdali není nutné aktualizovat stav online databáze. Pokud nebyly provedeny změny, které by bylo potřeba provést v online databázi, tak si program z databáze stáhne data a nahradí jimi stávající uložená data v programu, aby se do stávajícího systému zanesly změny provedené jinými přístupy do databáze.

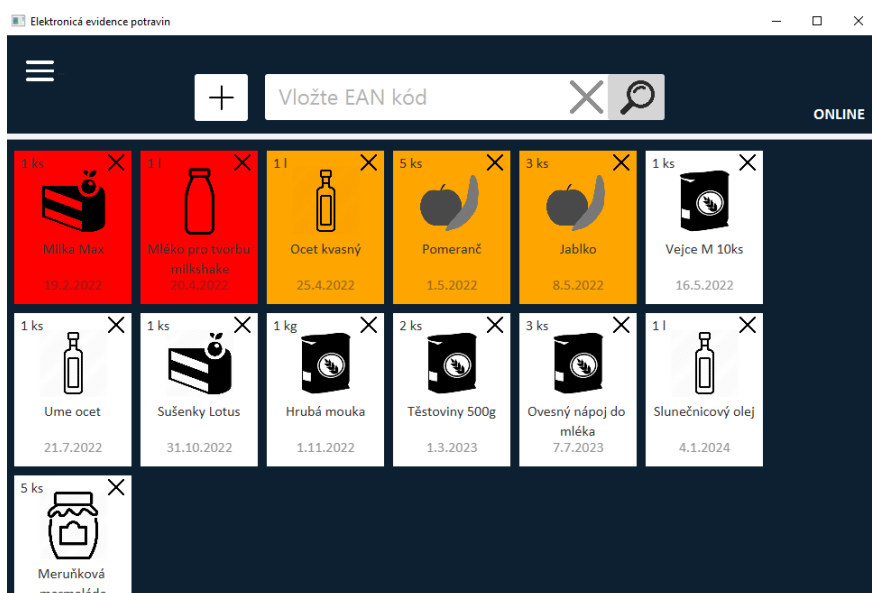


Obrázek 5 Okno přihlašování

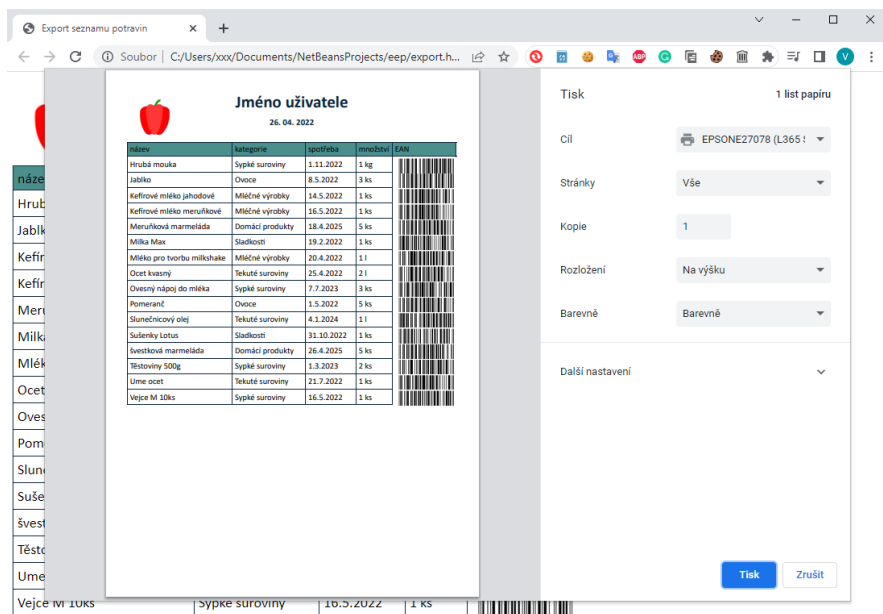
Obrázek 4 Okno přidání potraviny



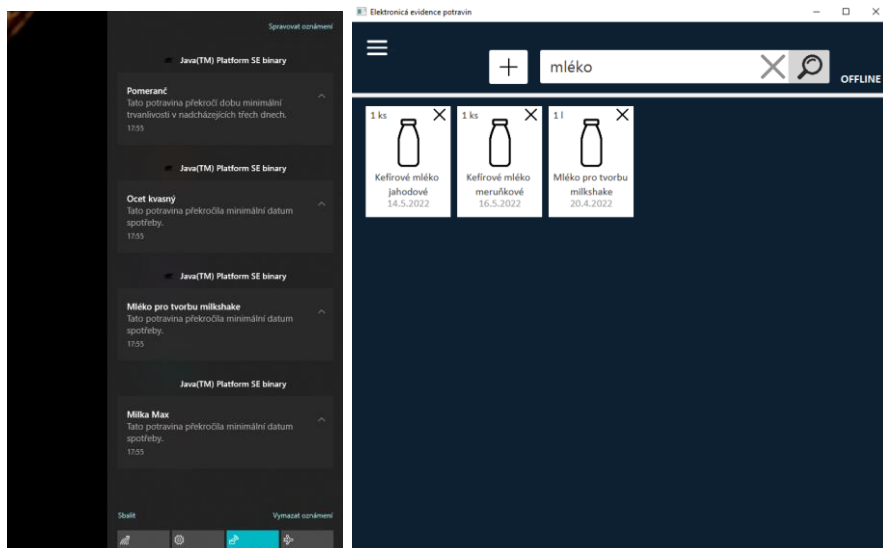
Obrázek 6 Ukázka hlavního okna a menu



Obrázek 7 Ukázka řazení podle spotřeby (červená – prošlé potraviny, oranžová – projde do 14 dní)



Obrázek 8 Ukázka exportu seznamu potravin



Obrázek 10 Notifikace

Obrázek 9 Ukázka hledání potravin podle názvu

4. Potravina

U potravin zaznamenávám následující atributy:

Id – je automaticky přidělován

Název – povinná položka zadávaná uživatelem

Kategorie – povinná položka, seznam kategorií prozatím není možné měnit

EAN kód – povinná položka naskenovaná čtečkou nebo vygenerovaná programem

Spotřeba – nepovinná položka zadaná uživatelem

Množství – povinná položka zadaná uživatelem

Jednotky – povinná položka, seznam jednotek nemůže uživatel měnit

Id potravinám přiděluje MySQL databáze. V případě, že program není k databázi připojen, vytváří dočasná identifikační čísla potravin, která se od přiřazených databází liší tím, že jsou záporná.

Název potraviny se používá ke snazší orientaci v databázi mezi potravinami. Jelikož potravina neobsahuje možnost přidat dodatečné informace jako je popis, slouží jako primární zdroj informací o dané potravine. Měl by tedy mimo samotného názvu potraviny obsahovat v případě potřeby i další informace. Příkladem by mohla být mouka, u které je důležité ji rozlišit (hladká, hrubá polohrubá). Tyto dodatečné informace stávající systém neumí pojmout jinak, než jejich začleněním do názvu, který by měl být co nejkratší.

Kategorie vznikla primárně za účelem výběru ikonky potraviny. Původně jsem chtěl mít ikonku pro každou potravinu, což by vyžadovalo velkou databázi ikon, které by musel uživatel dodávat, což by pro něj nebylo příliš pohodlné. Proto jsem vytvořil parametr kategorie, který má sloužit jako kompromis. Více se ikonkami potravin zabývám v kapitole o grafickém rozhraní.

EAN kód slouží jako identifikátor potraviny pro usnadnění uživateli potraviny v systému nalézt pomocí načtení čárového kódu. Systém jej nemůže použít jako identifikátor potraviny, jelikož v systému může být několik potravin se stejným kódem, ale rozdílnými parametry.

Uchovávání **spotřeby** je jednou z hlavních výhod používání tohoto programu. U některých potravin není však její určení možné, a tak zadání data spotřeby není povinné a program pak danou potravinu nezobrazí při filtrování podle data spotřeby a nebude upozorňovat na její datum expirace.

Množství je povinná položka, udávající množství dané potraviny v systému, které je možné měnit. Neznačí např. velikost balení, které je nutné v případě potřeby uvádět do názvu potraviny. To platí v případě, že chceme potravinu odebírat po jednotlivých kusech,

Jednotky udávají jednotky množství a mají vliv na možnosti odstraňování potravin z databáze kliknutím na křížek v rohu. V případě, že jednotky jsou nastavené na kusy nebo kilogramy, program při kliknutí na křížek odebere pouze jeden kus daného množství (pokud je množství větší než jeden kus). U ostatních jednotek (g, ml, %) dojde k odebrání celé potraviny, jelikož lze předpokládat, že uživatel nebude danou potravinu odebírat po gramech, ale daná hodnota slouží spíše jako informace o velikosti balení dané potraviny.

Okomentoval(a): [f1]: Možná doplnit

Okomentoval(a): [f2R1]: Hotovo



6. Synchronizace


Stěžejní částí celého projektu je zajištění synchronizace s online databází, jelikož jednou z funkcí této aplikace je možnost fungovat i v režimu offline. Pro správu dat v online databázi databáze přiřadí jednotlivým položkám jedinečné vygenerované id typu integer, který začíná hodnotou 1 a po přidání každé další položky se navýší o jedna. K přístupu k potravinám by bylo možné využít i jiné údaje, u kterých ale hrozí, že by mohly být díky funkci upravující parametry zadané potraviny upraveny a potravina by pak nebyla nalezena. Generování univerzálního Id, které by mohlo být následně nahráno s potravinou do databáze tak, aby nedošlo k případné kolizi mezi id, by bylo velmi obtížné. Jedním z možných řešení by mohlo být vést evidenci všech zařízení, které do databáze přispívají a přiřazování id jim rozdělit tak, aby nedocházelo ke kolizím.

Já jsem se rozhodl potravinám přiřazovat lokální id, které se dají od id přiřazených databází rozpoznat tím, že jde o záporná čísla. Aplikace funguje s vlastním offline seznamem potravin, který se ukládá do xml souboru a při spuštění aplikace se načte. V tomto seznamu mají potraviny pokud možno id přiřazené databází, pokud však program vytvoří novou potravinu, přidělí ji vlastní lokální id. Veškeré operace si program zaznamenává do seznamu. V cyklu opakujícím se každých třicet vteřin se pokusí provedené změny zanést do online databáze. U vytvořené potraviny si nechá vrátit přidělené id databází, na které změni id v lokálním seznamu potravin. Problém je, že veškeré další operace s danou potravinou provedené před změnou id používají stále lokálně přidělené id. Proto program uchovává tabulku s lokálními id. Díky tomu si může dohledat aktuální id pro přístup k dané potravíně v databázi. Po uplatnění veškerých změn v online databázi se uchovávání posledních lokálních id restartuje.

Seznam změn na provedení v online databázi je také ukládán do xml souboru, aby záznam o změnách zůstal zachován i po restartu celé databáze. Tabulku s lokálními id si program pamatuje i pro potraviny typu „ulozene“, což je záznam určený k usnadnění zadávání potraviny se stejným čárovým kódem, kterou uživatel již zadával. Toto řešení funguje, ale neřeší některé problémy, které by mohly nastat. Problém může nastat v prioritě změn, kdy může být jedna potravina upravena v prohlížečové verzi evidence a této aplikace. Ideální by bylo, kdyby měla prioritu změna provedená jako poslední. Webová verze si však historii změn neukládá a tak tuto funkci není možné uplatnit. Změny provedené v tomto programu budou mít vyšší prioritu, což vzhledem k zamýšlenému uplatnění tohoto projektu v domácnostech nevnímám jako velký problém.

7. EAN-13 standard

European Article Number (EAN) byl zaveden v roce 1976. Umožňuje dvanácticiferné číslo (většinou ve formátu UPC-A) převést do třinácticiferného kódu EAN-13 přidáním nuly na začátek. V rámci standardu UPC-A se číselný kód skládá ze tří skupin čísel. První šesticiferné předčísí je přidělováno UCC konkrétnímu výrobci, dalších 5 cifer kódu značí kód produktu, který přidělují jednotlivé společnosti. Poslední číslo slouží jako kontrolní prvek, který se vypočítá na základě předchozích čísel. K výpočtu kontrolního čísla sečteme všechny cifry, které se nachází na lichých pozicích (první číslo je na pozici 1). K tomu přičteme součet čísel na sudých pozicích vynásobený třemi. Tyto dva součty sečteme a kontrolní číslici tvoří poslední cifra daného součtu (modulo 10) odečtena od 10. (3) (4)



5 901234 123457 >

Obrázek 11 Ukázka EAN kódu

Výpočet kontrolního čísla by podle kódu na obrázku vypadal takto:

$$(5+0+2+4+2+4) + 3 * (9+1+3+1+3+5) = 83$$

Lichá strana Sudá strana

$$83 \% 10 = 3$$

Zbytek po dělení 10

$$10 - 3 = 7$$

Rozdíl od čísla 10

EAN-13 kód se dělí do následujících skupin:

- První dvě až tři čísla značí kód země (kód České republiky je 859)
Čísla 20 – 29 jsou vyhrazena pro interní použití a nepatří žádné zemi
977 – vyhrazeno pro magazíny (ISSN)
978 – 979 – vyhrazeno pro knihy (ISBN)
980 – 99 – vyhrazeno pro vouchery
- Následujících pět čísel jsou přidělena konkrétnímu výrobci
- Další pět a více čísel slouží jako kód produktu
- Poslední číslo slouží jako kontrolní prvek

7.1. EAN kód potravin

Z kódu na obalu potravin tedy dokážeme vyčíst zemi původu, kód výrobce a kód produktu. Jednoduše se však dá dopátrat jen země původu, jelikož existují centrální databáze obsahující názvy jednotlivých produktů, ale automatizovaný přístup je do nich zpoplatněn. Jediná informace, kterou dokážu z kódu na potravině získat, je fakt, že dané číslo má specifický výrobek. Jako zásadní nevýhodu tohoto systému považuji nemožnost vyčíst z kódu datum spotřeby, což by např. v obchodě, kde se nachází velké množství stejného produktu s různým datem trvanlivosti, umožňovalo monitorovat přesný počet potravin, kterým se krátí doba minimální spotřeby a nebylo by nutné ručně průběžně potraviny procházet.

Hlavním přínosem použití čárových kódů v projektu je rychlost vyhledávání potravin v seznamu načtením jejího kódu. Také si databáze vytváří vlastní seznam čárových kódů, které musel uživatel poprvé zadat. Jde o informaci o názvu potravin, kategorii, EAN kódu a použitých jednotkách pro množství. Tyto informace se uživateli při opětovném přidávání stejné potravin ve formuláři předvyplní.

V PHP verzi tohoto projektu jsem se pokoušel různými způsoby získat název produktu z čárového kódu. Nejlépe mi fungovala metoda, kdy program číselný kód vyhledal pomocí vyhledávače Google a do kolonky vyplnil obsah prvního názvu nalezené stránky. Úskalí této metody byl příliš dlouhý název, který mnohdy obsahoval další informace nebo s danou potravinou vůbec nesouvisel, jelikož šlo o sponzorované příspěvky.

7.2. Generování vlastních kódů

Pro možnost přidání potravin, které nedisponují vlastním kódem, program daným potravinám kód vygeneruje. Vygenerovaný kód by neměl kolidovat s existujícím kódem registrovaných potravin. Dále by bylo vhodné zamezit kolizím mezi účty různých uživatelů, kteří by si mohli potraviny s vygenerovanými kódy vyměňovat.

Na základě těchto požadavků jsem se inspiroval existujícím systémem. Abych zamezil možným kolizím se zaregistrovanými potravinami, využil jsem první předčísli v rozmezí, které je určeno pro interní potřeby (20 – 29). Další část čtyř cifer byla vyhrazena pro id účtu, aby se zamezilo případné kolizi při výměně potravin mezi uživateli. Do kódu jsem také chtěl zahrnout datum spotřeby ve formátu (ddmmyy). Ze třináctimístného kódu není možné využít poslední kontrolní číslo, první číslo musí být 2, aby se zabránilo kolizím s ostatními potravinami, kód měl obsahovat čtyři čísla pro id uživatele (maximální počet uživatelů by bez kolizí byl 10 000) a zbylé cifry by sloužily jako id potraviny.

Bohužel by pro id potraviny zbyla jen jedna cifra, což by znamenalo, že by daný uživatel mohl vytvořit jen 10 potravin se stejným datem spotřeby, což se mi jeví jako velmi pravděpodobné, jelikož u trvanlivějších potravin, jako jsou marmelády, by uživatel pravděpodobně nevybíral náhodná data spotřeby, ale stanovil by rok, do kterého by bylo dobré potravinu spotřebovat, a jako zbytek data by nastavil vždy to stejné (předpokládám 1.1.). Navíc toto id by se velmi obtížně počítalo, jelikož může jeden účet fungovat na několika zařízeních, která by spolu nekomunikovala.

Proto jsem musel svůj přístup přehodnotit. Polovinu kódu zabíralo datum spotřeby, které by bylo užitečné jen v případě, že by potravinu někdo načítal jako novou. V takovém případě by program mohl pole s datem spotřeby předvyplnit, což mi nepřipadá tolik užitečné. Datum spotřeby je navíc vytištěno na etiketě vedle čárového kódu. Jedním z případných řešení generování id by bylo evidovat připojená zařízení. Každému zařízení by program přiřadil vlastní identifikátor, kterým by se bránilo kolizi s jinými zařízeními. Toto řešení by fungovalo, ale vyžadovalo by velké zásahy do již napsaného kódu, který s podobnou evidencí připojených zařízení nepočítal. Byla by ale jistota, že by ke kolizím nedocházelo.

Zvolil jsem řešení, ve kterém se pokouším do kódu zanést otisk potraviny. To se může jevit podobně, jako kód skládat z náhodně zvolených čísel. Výhodou ale je, že při zadání stejných parametrů bude vygenerovaný kód vždy stejný. Případné kolize ale nelze vyloučit. Program vygenerované kódy pro případné předvyplňování neeviduje, kolize tedy budou nejvíce pro uživatele patrné, když se pokusí danu potravinu vyhledat v seznamu pomocí načtení kódu. Namísto předpokládané jedné potraviny by se mu v seznamu zobrazily všechny potraviny, které mají stejný kód, což lze předpokládat, že nastane jen velmi výjimečně. Uživatel by musel ze zobrazených potravin vybrat tu správnou podobným způsobem, jako program zobrazuje při hledání všechny evidované kusy stejného zboží s rozdílným datem expirace, ze kterých také uživatel musí najít ten správný bez další pomoci programu.

Jednotlivé části kódu generuji takto:

- 1. číslice - zabraňuje kolizi s existujícími komerčními produkty
- 2. – 5. číslice – id uživatele
- 6. – 7. číslice – id kategorie potraviny
- 8. – 9. číslice – zahashovaná hodnota spotřeby
- 10. – 12. číslice – zahashovaný název potraviny
- 13. číslice – kontrolní prvek

8. Ukázky kódu

Výpočet kontrolní číslice

```
String ean = "123456789123";
int liche = Character.getNumericValue(ean.charAt(0)) +
Character.getNumericValue(ean.charAt(2)) +
Character.getNumericValue(ean.charAt(4)) +
Character.getNumericValue(ean.charAt(6)) +
Character.getNumericValue(ean.charAt(8)) +
Character.getNumericValue(ean.charAt(10));
int sude = Character.getNumericValue(ean.charAt(1)) +
Character.getNumericValue(ean.charAt(3)) +
Character.getNumericValue(ean.charAt(5)) +
Character.getNumericValue(ean.charAt(7)) +
Character.getNumericValue(ean.charAt(9)) +
Character.getNumericValue(ean.charAt(11));
int kontrolniCislice = 10 - ((liche + 3*sude) % 10);
if (kontrolniCislice == 10) {
    kontrolniCislice = 0;
}
ean = ean + kontrolniCislice;
```

Zjednodušená ukázka funkce synchronizace

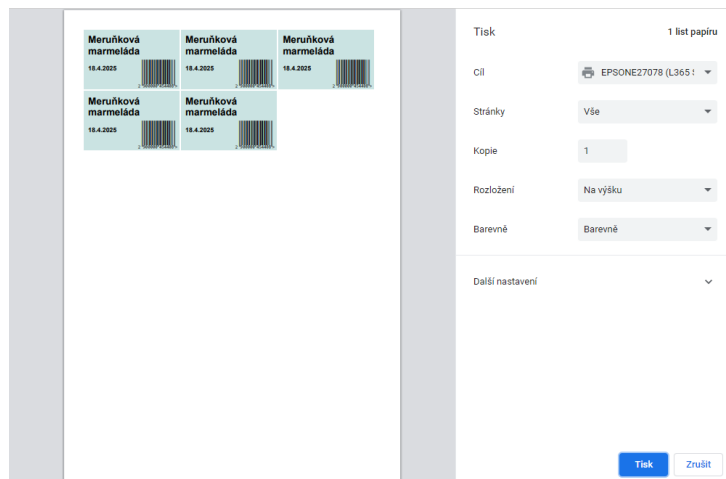
```
if (offlineSQL.size() > 0) {// Kontrola, pokud je co synchronizovat
    try {
        ...Připojení k databázi...
        //procházení záznamů úprav pro nahrání do databáze
        while (offlineSQL.size() != 0) {
            int id = offlineSQL.get(0).id;//získání id dané potraviny
            int aktualizovaneId = id;
            //pokud má potravina lokální id a již máme uložený její id z databáze, pak získej její id z databáze
            if (id < 0 && offlineSQL.get(0).vratitId == 0) {
                aktualizovaneId = (int) tabulkaId[Math.abs(id)];
            }
            //načti sql příkaz, nastav u něj správné id a odešli ho databázi
            String sql = offlineSQL.get(0).sql;
            sql = sql.replace("$ID$", aktualizovaneId + "");
            PreparedStatement updateEXP = connection.prepareStatement(sql);
            // Příklad, kdy do databáze ukládáme potravinu, které se přidělí id z online databáze
            if (offlineSQL.get(0).vratitId == 1) {
                Statement stmt = connection.createStatement();
                ResultSet rs = stmt.executeQuery("SELECT LAST_INSERT_ID();");
                while (rs.next()) {
                    int noveId = rs.getInt(1);
                    tabulkaId[Math.abs(id)] = noveId;//Přidej id do tabulky
                    //Aktualizuj id potraviny na to přidělené databází
                    for (int i = 0; i < potraviny.size(); i++) {
                        if (potraviny.get(i).id == id) {
                            potraviny.get(i).id = noveId;
                            break;
                        }
                    }
                }
            }
        }
    }
}
...odpoj se od databáze...
...restartuj počítání lokálních id...
} catch (SQLException ex) {
    //Zařízení je offline
}
}
```


9. Tisk etiket

Důležitou součástí možnosti přidat do databáze vlastní potravinu, která nemá vlastní čárový kód, je funkce tisku vygenerovaného čárového kódu. Popis generování číselného kódu jsem popsal v předešlé kapitole, ale samotný proces generování se ukázal být složitější, jelikož se mi nepodařilo najít vhodné volně dostupné knihovny, které by se hodily pro tento projekt. Jedním z možných řešení by bylo si vytvořit vlastní program, který by generoval obrázky s kódy. Já jsem se rozhodl pro mnohem jednodušší řešení a použil jsem speciální font, který jsem stáhl z Google Fonts (5).

Dalším krokem bylo najít vhodný způsob, jak vytvořené kódy vytisknout. Jako první jsem se pokusil najít vhodnou knihovnu, která by dokázala vytisknout obsah určitého grafického prvku JavaFx, což způsobovalo spoustu chybových hlášení. K tomu uživatelské rozhraní nastavení tisku nepůsobilo příliš uživatelsky přívětivě. Nezobrazovalo např. počet tištěných stran nebo náhled celého tisku. Proto jsem se pokusil tisk vyřešit vygenerováním pdf souboru, který by si mohl uživatel vytisknout pomocí vlastního programu pro editaci pdf souborů nebo pomocí prohlížeče. I v tomto případě se vyskytly chyby, knihovna nepodporovala kódování UTF-8. Zůstal jsem tedy u použití prohlížeče k tisku. Program vygeneruje html stránku s požadovaným počtem čárových kódů, kterou po vygenerování program zobrazí v preferovaném prohlížeči, kde se navíc pomocí příkazu napsaném v JavaScriptu otevře ihned okno pro tisk. (6)

Toto řešení podle mého kombinuje to nejlepší z předchozích možností. Nevyžaduje připojení k internetu a nabízí dobré uživatelské rozhraní pro tisk včetně náhledu. Navíc toto rozhraní umožňuje daný dokument v případě potřeby místo tisku exportovat do formátu pdf. V prohlížeči Chrome je možné pomocí odškrtnutí políčka Grafika na pozadí vypnout zobrazení barevného obdélníku v pozadí každé etikety, což je užitečná funkce v případě, že by barevné pozadí bránilo správnému načtení čárového kódu čtečkou. Mimo čárový kód jsem na etiketu potraviny umístil její název a datum spotřeby (pokud jej uživatel zadal). V případě příliš dlouhého názvu (Na etiketě je prostor pro 45 znaků) dojde k jeho oříznutí, což není ideální řešení, ale alespoň text nerozhodí formátování zbytku dokumentu.



Obrázek 12 Ukázka exportu etiket

10. Závěr

Mezi hlavní problémy použití tohoto programu považuji obtížnou instalaci, které by bylo možné se vyhnout vytvořením centrální databáze, na kterou by byl již program nakonfigurován. Jinak program obsahuje dostatečné množství funkcí k tomu, aby mohl v domácnostech pro evidenci potravin fungovat s tím, že má potenciál mnoha dalších rozšíření a propojení s dalšími službami, kterými by mohla být funkce filtrovat recepty na základě toho, co se v domácnosti nachází, nebo možnosti jako je automatické plánování nákupních seznamů s přihlédnutím na aktuálně probíhající akce v supermarketech, odkud by mohlo v budoucnu plynout nejvíce příjmů z daného projektu. Model předplatného dané služby se mi nejeví jako vhodný, jelikož by to mělo negativní vliv na rychlost šíření na trhu.

Program v tomto stavu je schopný si zapamatovat přihlášeného uživatele, přidávat a odebírat potraviny, generovat čárový kód pro nové potraviny a vygenerovat tisknutelné etikety pro dané potraviny, upravovat zadané parametry potravin a pamatovat si parametry již zadaných potravin za účelem urychlení zadávání hodnot při opětovném vkládání potraviny do systému. Tyto úkony fungují navíc bez nutnosti připojení k databázi a po navázání spojení dojde k jejich synchronizaci. Dále program umožňuje vyhledávat v seznamu potravin na základě EAN kódu, názvu (k nalezení potraviny stačí jen část názvu), dále program umožňuje potraviny seřadit podle data spotřeby, prošlé potraviny začervenit a ty potraviny, které projdou do čtrnácti dní, obarvit oranžovou barvou. Poslední funkce této aplikace je možnost exportovat potraviny do seznamu seřazeného podle jména potraviny, který je možné v prohlížeči vytisknout nebo uložit jako pdf. Tento seznam obsahuje čárové kódy potravin, které je možné naskenovat a pomocí nich danou potravinu v databázi rychle vyhledat.

10.1. Možnosti rozšíření

Tato aplikace umožňuje základní operace pro správu potravin včetně exportu vygenerovaného seznamu potravin. Oproti webové verzi aplikace chybí možnost změny pevně daného seznamu kategorií kvůli problému s jejich ikonkami, které by se musely synchronizovat v rámci obou verzí programu. Další funkcí, o kterou by bylo možné tento program rozšířit je funkce automatického zálohování a správa záloh nebo funkce nastavení minimálního množství některých potravin, kdy by program upozornil v případě, že by jejich počet překročil minimální limit. Další možné funkce by mohly zahrnovat pokročilejší možnosti exportu potravin z databáze a možnosti jejich importu.

12. Bibliografie

1. **U.S. DEPARTMENT OF AGRICULTURE.** Food Waste FAQs. *U.S. DEPARTMENT OF AGRICULTURE*. [Online] [Citace: 22. 5 2022.] <https://www.usda.gov/foodwaste/faqs#>.
2. **Evropská komise.** Food Safety. *Evropská komise*. [Online] [Citace: 22. 5 2022.] https://ec.europa.eu/food/safety/food-waste_cs.
3. **Appsforlife.** EAN-13 Check Digit Calculator. *boxshot*. [Online] [Citace: 18. 4 2022.] <https://boxshot.com/barcode/tutorials/ean-13-calculator/>.
4. **barcodestalk.** Country codes of the EAN-13. *barcodestalk*. [Online] [Citace: 18. 4 2022.] https://www.barcodestalk.com/countrycodes_ean13.
5. **Fister, Lasse.** Libre Barcode EAN13 Text. [Online] Google. [Citace: 18. 4 2022.] https://fonts.google.com/specimen/Libre+Barcode+EAN13+Text?query=ean&preview.text=213654854545&preview.text_type=custom#license.
6. **Contreras, Giovanni.** Export Stage into PDF. *stackoverflow*. [Online] [Citace: 18. 4 2022.] <https://stackoverflow.com/questions/22789449/export-stage-into-pdf>.
7. **Việt, Hùng Phạm a krzysiek.ste.** How to open the default webbrowser using java. *Stackoverflow*. [Online] 3 2011. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/5226212/how-to-open-the-default-webbrowser-using-java>.
8. **DVarga.** How to listen resize event of Stage in JavaFX? *stackoverflow*. [Online] 2016. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/38216268/how-to-listen-resize-event-of-stage-in-javafx>.
9. **Atherton, Leon.** How to add a window resize listener to JavaFX scene. *IDR SOLUTIONS*. [Online] 20. 11 2012. [Citace: 25. 4 2022.] <https://blog.idrsolutions.com/2012/11/adding-a-window-resize-listener-to-javafx-scene/>.
10. **Javatpoint.** Java Database Connectivity with MySQL. *Javatpoint*. [Online] [Citace: 25. 4 2022.] <https://www.javatpoint.com/example-to-connect-to-the-mysql-database>.
11. **MySQL.** 6.1 Connecting to MySQL Using the JDBC DriverManager Interface. *MySQL*. [Online] [Citace: 25. 4 2022.] <https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-usagenotes-connect-drivermanager.html>.
12. **Owen, Sean.** Connect Java to a MySQL database. *stackoverflow*. [Online] 2010. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/2839321/connect-java-to-a-mysql-database>.
13. **Adam.** How to get the current opened stage in JavaFX? *stackoverflow*. [Online] 2019. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/32922424/how-to-get-the-current-opened-stage-in-javafx>.
14. **Saikat a zhujik.** How to get stage from controller during initialization? *Stackoverflow*. [Online] 2012. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/13246211/javafx-how-to-get-stage-from-controller-during-initialization>.
15. **Abramov, Andrii a invariant.** Platform.runLater and Task in JavaFX. *Stackoverflow*. [Online] 2013. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/13784333/platform-runlater-and-task-in-javafx>.
16. **SleightX a randers.** How to make a Windows Notification in Java. *Stackoverflow*. [Online] 2016. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/34490218/how-to-make-a-windows-notification-in-java>.
17. **LLC, Weber Informatics.** org.bouncycastle.jcajce.provider.digest.SHA3 Maven / Gradle / Ivy. *jar-download*. [Online] [Citace: 25. 4 2022.] <https://jar-download.com/artifacts/org.bouncycastle/bcprov-ext-jdk14/1.49/source-code/org/bouncycastle/jcajce/provider/digest/SHA3.java>.
18. **Lii a James_D.** Updating UI from different threads in JavaFX. *Stackoverflow*. [Online] 2014. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/22772379/updating-ui-from-different-threads-in-javafx>.

19. **jewelsea a CommunityBot**. JavaFX: Updating UI elements in a Controller class from a Thread. *Stackoverflow*. [Online] 2013. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/17873597/javafx-updating-ui-elements-in-a-controller-class-from-a-thread>.
20. **riptutorial**. How to use JavaFX Service. *riptutorial*. [Online] [Citace: 25. 4 2022.] <https://riptutorial.com/javafx/example/23625/how-to-use-javafx-service>.
21. **baeldung**. A Guide to the Java ExecutorService. *baeldung*. [Online] 22. 12 2021. [Citace: 25. 4 2022.] <https://www.baeldung.com/java-executor-service-tutorial>.
22. **Mendoza, Luigi**. Print "hello world" every X seconds. *Stackoverflow*. [Online] 2012. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/12908412/print-hello-world-every-x-seconds>.
23. **Slaw**. JavaFX periodic background task. *Stackoverflow*. [Online] 2012. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/9966136/javafx-periodic-background-task>.
24. **RGO**. How to compare two array lists for similar objects which differ in at least one property in java? *Stackoverflow*. [Online] 2012. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/12853121/how-to-compare-two-array-lists-for-similar-objects-which-differ-in-at-least-one>.
25. **jewelsea**. How to write a KeyListener for JavaFX. *Stackoverflow*. [Online] 2015. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/29962395/how-to-write-a-keylistener-for-javafx>.
26. **James_D**. JavaFX change the font color of text in a tab. *Stackoverflow*. [Online] 2017. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/46354731/javafx-change-the-font-color-of-text-in-a-tab>.
27. **kelunik a Teocali**. How to close all stages when the primary stage is closed? *Stackoverflow*. [Online] 2012. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/12194967/how-to-close-all-stages-when-the-primary-stage-is-closed/12195366>.
28. **ysrb**. The split() method in Java does not work on a dot (.) [duplicate]. *Stackoverflow*. [Online] 2011. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/7935858/the-split-method-in-java-does-not-work-on-a-dot/7935873>.
29. **cole.markham**. Get value from Date picker. *Stackoverflow*. [Online] 2014. [Citace: 25. 4 2022.] <https://stackoverflow.com/questions/20446026/get-value-from-date-picker>.

13. Seznam obrázků

Obrázek 1 Struktura databáze uživatelů	7
Obrázek 2 Struktura databáze potravin	7
Obrázek 3 Vygenerovaná paleta barev pro tento projekt	8
Obrázek 4 Okno přidání potraviny	8
Obrázek 5 Okno přihlašování	8
Obrázek 6 Ukázka hlavního okna a menu	9
Obrázek 7 Ukázka řazení podle spotřeby	9
Obrázek 8 Ukázka exportu seznamu potravin	10
Obrázek 9 Ukázka hledání potravin podle názvu	10
Obrázek 10 Notifikace	10
Obrázek 11 Ukázka EAN kódu	14
Obrázek 12 Ukázka exportu etiket	17