PRAXE

3	**SPŠ Chomutov**	**Novotný Malý**
31.1. 2017	**Přístupový systém RFID**	**V4**

Zadání

Vytvořte program pro skladový systém se skenerem čárových kódů.

Program musí:

- Umět identifikovat uživatele
- Umět kontrolovat dostupnost zboží
- Umět vygenerovat textovou výdejku do adresáře
- Být ovládán hlavně přes numerický blok na klávesnici
- Být přímočarý
- Zaznamenávat historii uživatele, skladu a dostupné či nedostupné položky
- Mít limity pro každého uživatele (za týden, limit na cenu zboží)
- Mít implementovanou funkci podlimitního množství

Teorie

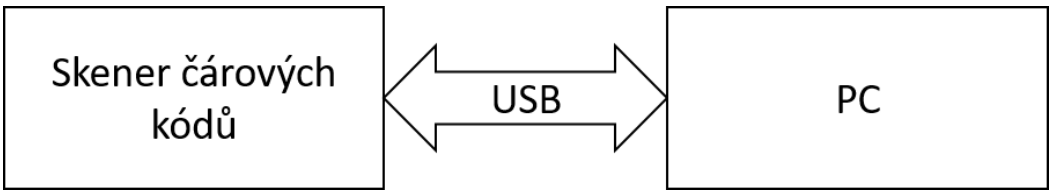
Čárový kód umožňuje rychle číst číselná data z tištěné podoby do počítače. Skládá se ze stejně širokých černých a bílých sloupečků, které podle množství odraženého světla čtečka vyhodnotí jako 1 nebo 0. Kódy mohou být lineární nebo dvourozměrné (ty obsahují více dat, ale vyžadují pokročilejší čtečku). Součástí kódů jsou i části umožňující kontrolovat správnost orientace čtení a integritu přečtených dat.

Čtečky kódů se často pro předání informací počítači chovají jako klávesnice. Přečtený kód vlastně „napíší“ stejně jako by to udělal člověk na klávesnici. Je proto potřeba správně (stejně) nastavit rozložení klávesnice, jak na čtečce, tak na koncovém zařízení.

SQLite je relační databázový systém. Na rozdíl od jiných implementací SQL jazyka není určen k provozu sdíleně na serveru, ale k použití spolu s aplikací jako její úložiště. Nejedná se tedy o samostatně spouštěný proces, ale o knihovnu a databázi obsluhuje sám program. Databáze je uložena v jediném souboru nezávislém na operačním systému, což usnadňuje migrování. Jde o relativně minimální implementaci, která ale podporuje většinu standardu SQL2.

Tady chybí především popis Ado.NET technologie, relace tabulek v SQL, případně reakce na enter při vstupu.

Schéma zapojení



V případě jednoduchého zapojení lze tolerovat, jinak se podle tohoto schématu nedá zapojit nic …

Popis řešení

Interakci s databází obstarává class **db**. Vytvořením její instance dojde k připojení k požadované databázi (nebo jejímu vytvoření) a případnému založení tabulek podle dotazů uložených v nastavení aplikace.

Obsahuje metody pro čtení dat z databáze, které podle dodaných argumentů doplní a vykonají dotaz. Data přečtená z **SQLiteDataReader**u vrátí jako objekt. Vlastnosti tohoto objektu pak odpovídají sloupečkům v databázi.

Podobně fungují i metody pro ukládání dat do databáze, s tím rozdílem, že vrací true nebo false, podle úspěšnosti vykonávání dotazu.

Program nejprve založí instanci class **db** pro obsluhu databáze a přepne rozložení klávesnice na anglické, aby bylo shodné s výchozím nastavením čtečky.

Před otevřením hlavního okna vyvolá metoda **performIdentification** formulář pro naskenování UID. Tato metoda je volána při každém požadavku změnit uživatele. Při každém přihlášení vrací **db** class v objektu uživatele kromě dat z databáze i vypočtený limit pro aktuální výběr. Ten spočítá ze základního limitu uživatele a již vyčerpaného množství, které určí z existujících záznamů akcí.

Otevře se hlavní okno, kde program čeká na stisk klávesy. Podle stisknuté klávesy a dalších okolností vyvodí požadovanou akci:

- Stisk **/** – Přepnutí uživatele
- Stisk **+** – Přepnutí do administrace
- Stisk **-** – Odebrání položky ze seznamu
- Stisk **Enter**u – Naskenování nebo potvrzení zadaného kódu

Z tohoto řešení si úplně vzor neberte …

Při **odebírání položky** se vyvolá formulář pro naskenování kódu produktu, který se má odebrat. Program si vyžádá nalezení produktu v databázi a pokud existuje, vyvolá se formulář pro zadání množství. Program pak prohledá seznam vybraných produktů a pokud ten obsahuje odstraňovaný produkt, odečte požadované množství nebo jej odebere úplně.

Po **naskenování kódu produktu** do hlavního okna program určí, o jaký kód se jedná a podle toho vykoná příslušnou akci:

- Kód má **11 znaků** – kód kategorie, pouze vypíše název kategorie z databáze
- Kód má **10 znaků** – kód produktu, provede se přidání na seznam
- Kód začíná **znaky „UID“** – kód uživatele, dokončí výběr ze skladu

Pro **přidání produktu na seznam** si program vyžádá nalezení produktu v databázi a pokud existuje, vyvolá formulář pro zadání množství (výchozí je 1). Po potvrzení množství program prohled aktuální seznam a pokud ten obsahuje typ přidávaného produktu, pouze přičte množství. Jinak jej přidá jako novou položku. Program také podle dat z databáze kontroluje maximální dostupné množství produktu a případně omezí jeho přidané množství.

Při každé akci, která mění obsah seznamu program obnoví zobrazení limitu uživatele.

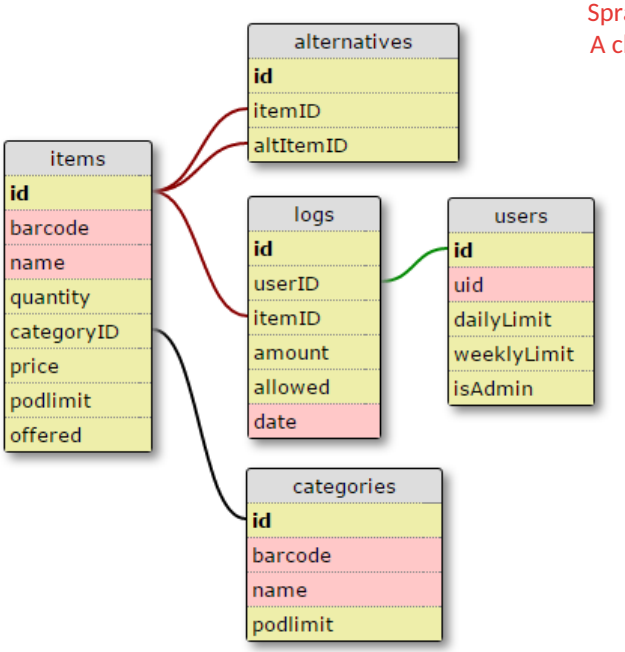
Dokončení výběru ze skladu uživatel provede naskenováním svého UID. Pokud naskenuje cizí, program na to upozorní. Jinak zkontroluje, že uživatel nepřesahuje svůj aktuální limit (vypočtený při přihlášení) a pokud ne, zavolá metody pro uložení akcí do záznamů (**saveAllTransactions**) a vytištění výdejky (**createReceipt**). Nakonec uživatele odhlásí.

saveAllTransactions projde seznam a zavolá pro každou položku metody v **db** class – pro uložení záznamu a pro aktualizaci dostupného množství v databázi.

createReceipt založí textový soubor v adresáři výdejek daného uživatele a vypíše do něj každou položku na seznamu spolu s informací o množství, ceně a další.

Vizualizace databáze

Správně má být pod rozborem proměnných.
A chybí typy datových polí ...



Rozbor funkcí a proměnných

Typ	Proměnná	Účel
	Class **Form1**	
db	**data**	Instance db class
db.user	**currentUser**	Aktuální uživatel
List<chosenItem>	**cart**	Seznam položek
int	**currentSum**	Celková cena
string	**inputText**	Kód ze vstupního pole
db.item	**foundItem**	Položka z databáze
db.category	**cat**	Kategorie z databáze
	Class **db**	
string	**dbname**	Jméno databáze použité pro připojení
DateTime	**currentDate**	Aktuální časové razítko pro práci se záznamy
bool	**initiated**	Indikuje připravenost databáze
různý	**result**	Výsledek čtení z databáze, typ podle tabulky
	Class **scanQueryForm**	
string	**barcode**	Kód zadaný do formuláře
string	**type**	Zjištěný typ kódu
string	**computedResult**	Upravený kód podle typu (např. substring UID)
	Class **numberQueryForm**	
int	**addingAmount**	Množství zadané do formuláře
int	**price**	Cena vypočítaná podle položky
	Class **adminForm**	
db	**data**	Instance db class

Proměnné v objektech pro tabulky databáze odpovídají sloupečkům konkrétní tabulky.

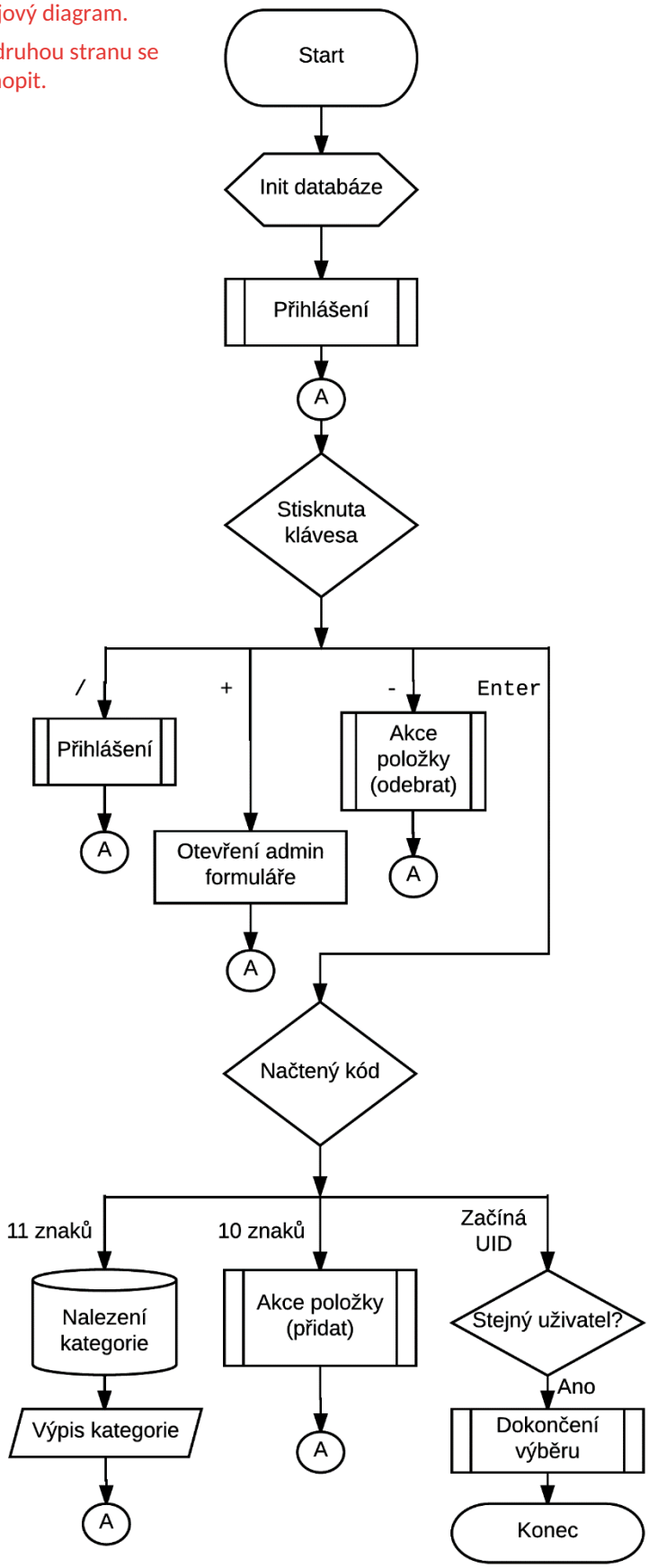
Typ	Argumenty	Metoda	Účel
		Class **Form1**	
void		**performIdentification**	Přihlášení uživatele pomocí UID
void		**identificationFail**	Akce při selhání přihlášení
void		**checkPodlimit**	Kontrola položek pod limitním množstvím
InputLanguage	inputName	**GetInputLanguageByName**	Nalezení rozložení klávesnice

void		**refreshOutput**	Obnovení zobrazení seznamu
void		**refreshAllowance**	Obnovení zobrazení limitů
int		**getSum**	Výpočet celkové ceny
int	foundItem, cart	**findExistingInCart**	Nalezení indexu položky v seznamu
bool		**saveAllTransactions**	Uložení záznamů do databáze
bool		**createReceipt**	Tisk výdejky
void	sender, e	**textBox1_KeyUp**	Zpracování stisknutých kláves
Class **db**			
konstruktor	dbname	**db**	Konstruktor
bool	reset	**init**	Inicializace databáze a spojení
category	categoryID	**getCategoryByID**	Vyhledání kategorie podle ID
category	barcode	**getCategoryByBarcode**	Vyhledání kategorie podle čárového kódu
item	barcode	**getItemByBarcode**	Vyhledání položky podle čárového kódu
user	UID	**getUserByUID**	Vyhledání uživatele podle UID
int	target	**computeSessionLimit**	Výpočet limitu pro aktuální výběr
int	date	**weekOfYear**	Určení čísla týdne
List<item>		**itemsPodlimit**	Kontrola položek pod limitním množstvím
string[]		**getLogArrayPlain**	Získání pole záznamů pro výpis
bool	t	**saveTransactionToLog**	Zápis transakce do záznamů
bool	item, amount	**saveItemQuantity**	Aktualizace dostupného množství
Class **scanQueryForm**			
konstruktor	qLabel	**scanQueryForm**	Konstruktor
void	sender, e	**inputBox_KeyUp**	Zpracování po stisku klávesy
void	sender, e	**cancelButton_Click**	Zrušení dialogu
Class **numberQueryForm**			
konstruktor	itemPrice, itemName	**numberQueryForm**	Konstruktor
void	sender, e	**amountBox_Enter**	Vybrání textu v text. poli
void	sender, e	**amountBox_KeyUp**	Zpracování po stisku klávesy
void		**handleConfirm**	Potvrzení dialogu
void		**handleCancel**	Zrušení dialogu
Class **scanQueryForm**			
konstruktor		**adminForm**	Konstruktor
void	sender, e	**adminForm_KeyUp**	Zpracování po stisku klávesy

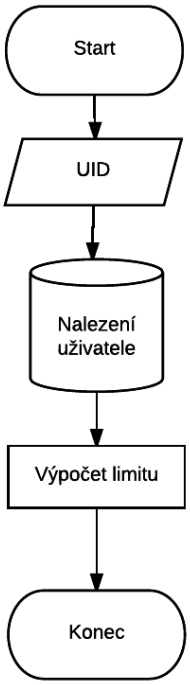
Vývojový diagram

Toto je spíš pokus o vývojový diagram.

Chyb je celá řada, ale na druhou stranu se postup programu dá pochopit.

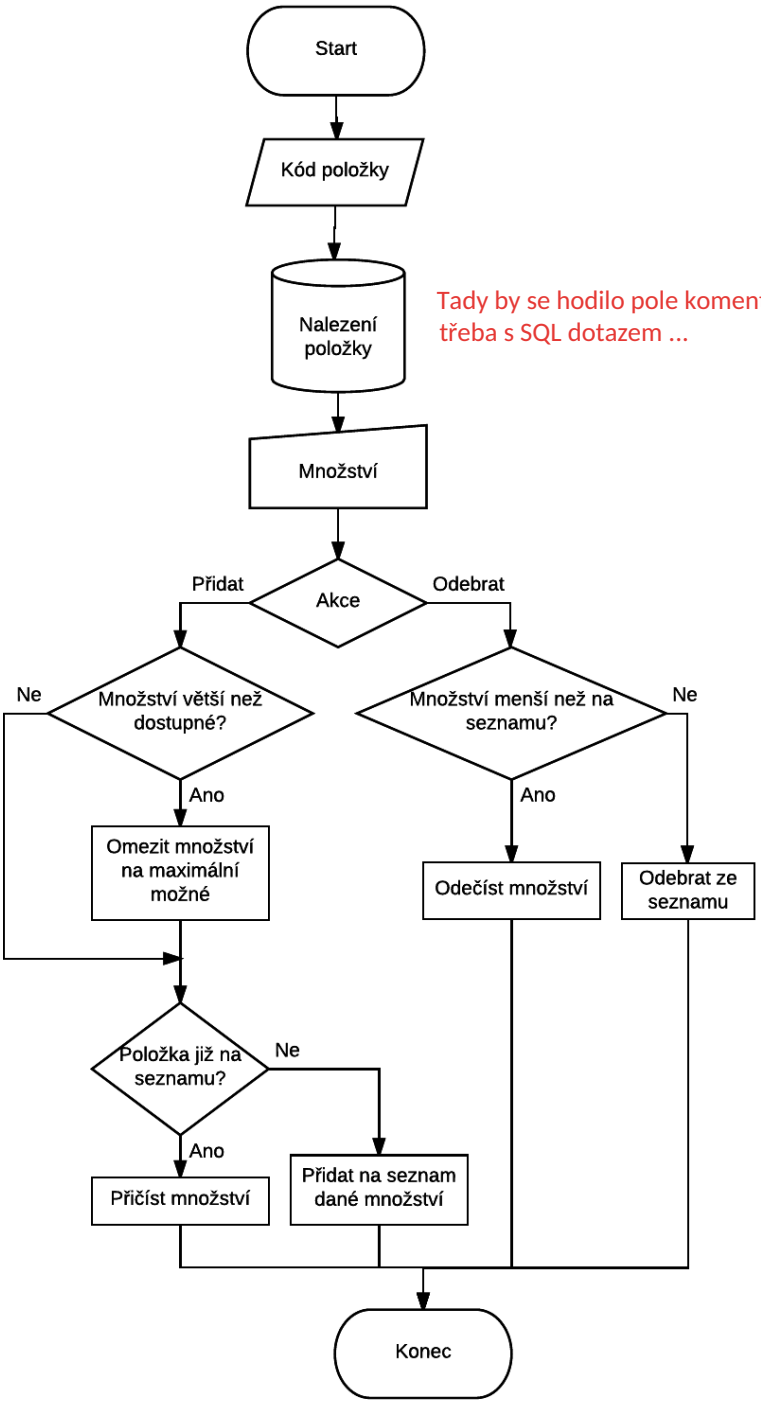


Přihlášení



Dokončení výběru

Akce položky



Tady by se hodilo pole komentáře třeba s SQL dotazem ...

Výpis programu

Form

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Drawing;
4.  using System.IO;
5.  using System.Windows.Forms;
6.
7.  namespace Sklad
8.  {
9.      public partial class Form1 : Form
10.     {
11.         public Form1() {
12.             InitializeComponent();
13.         }
14.
15.         db data;
16.         // Přihlášený uživatel (null = nikdo)
17.         db.user currentUser = null;
18.         // Seznam produktů k odběru
19.         List<chosenItem> cart = new List<chosenItem>();
20.         // Aktuální celková cena
21.         int currentSum = 0;
22.
23.         //
24.         // Login
25.         //
26.         private void performIdentification() {
27.             // Nový skenovací formulář
28.             using (scanQueryForm UIDDialog = new scanQueryForm("Naskenujte své UID")) {
29.                 var result = UIDDialog.ShowDialog();
30.                 // Pokud formulář OK a naskenovaný typ je UID
31.                 if (result == DialogResult.OK && UIDDialog.type == "user") {
32.                     // Přepnout na obrázek IKEA pracovníka
33.                     employeePic.ImageLocation = "http://i.imgur.com/rEth3kA.png";
34.                     // Podle UID najít uživatele v databázi a uložit jako aktuálního
35.                     currentUser = data.getUserByUID(UIDDialog.computedResult);
36.                     // Pokud úspěšně přihlášen...
37.                     if (currentUser != null) {
38.                         // Zkontrolovat položky pod limitním množstvím
39.                         checkPodlimit();
40.                         statusBox.Text = "Můžete začít. Tady najdete informace o stavu.";
41.                         // Oznámit uživatele
42.                         usernameLabel.Text = currentUser.uid;
43.                         // Aktualizovat zobrazené limity uživatele
44.                         refreshAllowance();
45.                       // Jinak...
46.                     } else {
47.                         // Oznámit selhání
48.                         identificationFail();
49.                     }
50.                   // Zpracování zrušení dialogu
51.                 } else if (result == DialogResult.Cancel && currentUser != null) {
52.                     return;
53.                 } else {
54.                     // Oznámit selhání
55.                     identificationFail();
56.                 }
57.             }
58.         }
59.
60.         //
61.         // Selhání přihlášení
62.         //
63.         private void identificationFail() {
64.             // Přepnout na obrázek selhání
65.             employeePic.ImageLocation = "http://i.imgur.com/7XAXvB6.png";
66.             // Oznámit neplatné UID
```

... atd, pro představu co má být v kódu asi stačí ...

```
                                // Obnovení zobrazení seznamu
                                refreshOutput();
                                // Oznámit přidané položky, popř. že došly (max)
                                statusBox.Text = String.Format("Přidáno {0} produktů {1}{2}. Položky o
    deberete klávesou [Mínus].", adding, foundItem.name, maxed ? ", protože došly" : "");
                            }
                        //
                        // Délka vstupu 11 zn.: Kód kategorie
                        //
                    } else if (inputText.Length == 11) {
                        // Nalezení v databázi
                        db.category cat = data.getCategoryByBarcode(inputText);
                        // Neexistující kategorie
                        if (cat == null) {
                            statusBox.Text = "Nenalezena kategorie s tímto kódem.";
                          // Oznámení názvu kategorie
                        } else {
                            statusBox.Text = String.Format("Kategorie '{0}'", cat.name);
                        }
                    }
                    //
                    // Neplatný vstup
                    //
                    else {
                        statusBox.Text = "Neplatný kód!";
                    }

                    inputBox.Text = "";
                }
            }

            // Struktura pro položky na seznamu: Položka a množství
            class chosenItem
            {
                public db.item item { get; set; }
                public int amount { get; set; }
            }
        }
    }
```

Db.cs

```
using System;
using System.Collections.Generic;
using System.Data.SQLite;
using System.Globalization;

namespace Sklad
{
    class db
    {
        SQLiteConnection conn;
        string dbname;
        DateTime currentDate = DateTime.Now;
        public bool initiated;

        public db(string dbname) {
            this.dbname = dbname;
            this.conn = new SQLiteConnection(String.Format("Data Source = {0}.db", this.dbname));
            this.initiated = this.init(false);
        }

        // Inicializace připojení
        bool init(bool reset) {
            this.conn.Open();
            SQLiteCommand cmd = new SQLiteCommand();
            // Smazat db, pokud resetujeme
            if (reset) System.IO.File.Delete(String.Format("{0}.db", this.dbname));
            // Vytvořit db, pokud neexistuje
            if (!System.IO.File.Exists(String.Format("{0}.db", this.dbname))) SQLiteConnection.Create
    File(String.Format("{0}.db", this.dbname));
```

… zbývající třídy z ukázky vyjmuty …

Odpovědi na otázky

1. Pro řízení rozsáhlých skladů se používají tzv. warehouse management systémy. Jaká je základní podmínka pro nasazení tohoto typu systému pro efektivní řízení skladu?

   – Je to efektivní řízení skladu pro velké sklady s velkým počtem položek.

2. S využitím vhodného generátoru QR kódu vygenerujte a vložte použitelný kód obsahující vaše jméno a emailovou adresu.[1]

3. Ne vždy je možné se ve skladovém hospodářství spolehnout na unikátní EAN kód skladové položky – např. pro jeho nedostupnost či velikost. Navrhněte efektivní řešení této situace.

   – Na položky by se lepily i nálepky s RFID, které se dají skenovat z větší dálky. Dále by se vedl seznam položek pro případ vážné nouze, ve kterém by se skladník vyznal. (Seznam položek by byl optimální asi pouze pro malé či středně velké sklady)

Odpovědi nejsou úplně dokonalé, poprosím o lepší formulace ať se nemusí tolik ptát při hodnocení ...

Závěr

Úlohu se nám podařilo zcela dokončit. Sklad je funkční.

... a venku je hezky, ptáčci zpívají a nebe je modré ... :-D

[1] http://generator-uzsi-najdete ...

Zdrojů je také méně než málo, na druhou stranu odpovědi další zdroje možná nepotřebovaly