RESTAURANT ORDER SYSTEM

Éttermi rendelési rendszer a dolgozók számára

INFORMATIKA SZAKDOLGOZAT DOKUMENTÁCIÓ

Diák:
Szakács-Kádár Norbert

Tanintézmény:
Mikes Kelemen Elméleti Líceum - Sepsiszentgyörgy

Felkészítő tanárok:
Erdőközi Enikő Anna, Gábor Béla

Elkészítés éve:
2023

Tartalom

Témaválasztás indoklása

Azért választottam egy ilyen alkalmazás fejlesztését, mert szerettem volna valami gyakorlatiasabbat, hasznosat alkotni. Hozzám legközelebb a vendéglátás szakma állt, ahová el is tudtam volna képzelni valamilyen rendszert. Így jutottam el egy egyszerű, de mégis hatékony rendelési rendszer fejlesztéséhez.

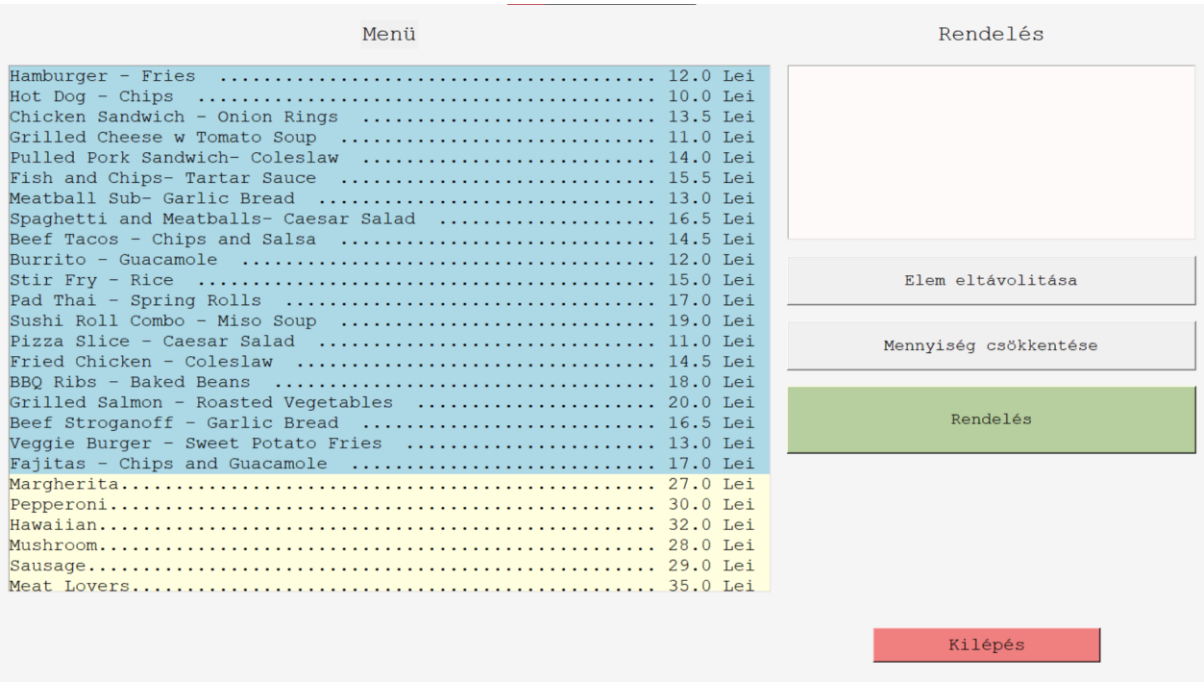
Úgy az alkalmazottak, mint az étterem számára egy ilyen rendszer hasznos lehet a hatékonyabb és gyorsabb üzleti folyamatok biztosítása érdekében. Az alkalmazás lehetővé teszi, hogy az alkalmazottak könnyedén és gyorsan rögzíthessék a megrendeléseket. Továbbá az alkalmazás használata segíthet csökkenteni az emberi hibák és tévedések számát, mivel az adatokat elektronikus formában rögzíti és tárolja, ezenkívül az alkalmazás segíthet csökkenteni a papíralapú folyamatok számát, mivel a számlákat elektronikusan „txt” file-okban tárolja, ami egy jóval környezetbarátabb megoldás.

Az alkalmazás további előnye, hogy lehetőséget biztosít a megrendelések pontosabb nyomon követésére és elemzésére, ami segíthet az üzleti folyamatok hatékonyabbá tételében, mivel minden számlát megőriz a rendszer, így a könyvelés és a készletellenőrzés is gyorsabbá válhat.

Tehát összességében egy rendelési alkalmazás használata javíthatja az alkalmazottak és a vállalat hatékonyságát és teljesítményét, ezáltal növelve az ügyfél elégedettségét és a vállalat versenyképességét.

A teljes kezelőfelület rövid bemutatása

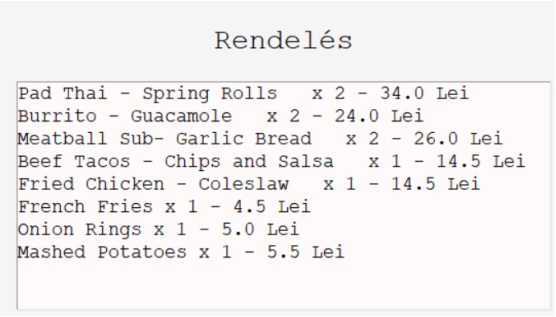
Az alkalmazás futtatása után a felhasználó egyből találkozik a teljes felülettel.(1. ábra)



(1.ábra)

Az alkalmazás két megjelenítési listadobozt tartalmaz, a **Menü** listadobozt és a **Rendelés** listadobozt. A menü listadoboz értelem szerűen a rendelhető tételeket tartalmazza, és elemei kategoriák szerint kapták meg színüket, hogy könnyebb legyen az alkalmazásban a navigáció. (Például: Menük-kék, Pizzák – sárga, stb...). A rendelés listadobozan pedig a rendeléshez hozzáadott tételek fognak megjelenni.(2.ábra)

A hozzáadás módja a **dubla kattintás**, és bármely elem végtelen sokszor hozzáadható a rendeléshez.

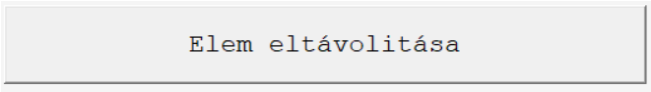


(2.ábra)

Továbbá a felületen található még négy felirattal, esetenként színnel ellátott gomb. (3.ábra)

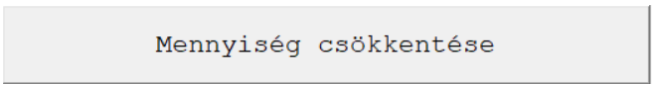
(3.ábra)

Az **Elem eltávolítása** (3.1 ábra) gomb a rendelésből a törölni kívánt termék kijelölése után eltávolítja az összes darabot abból a termékből, tehát törli a rendelésből.



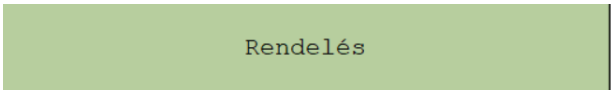
(3.1 ábra)

A **Mennysigég csökkentése** (3.2 ábra) gomb a rendelésből a csökkenteni kívánt termék kijelölése után eltávolít pontosan egy darabot, tehát ha több mint egy van csökkenti a darabszámot, ha pedig egy van, teljesen törli a rendelésből.



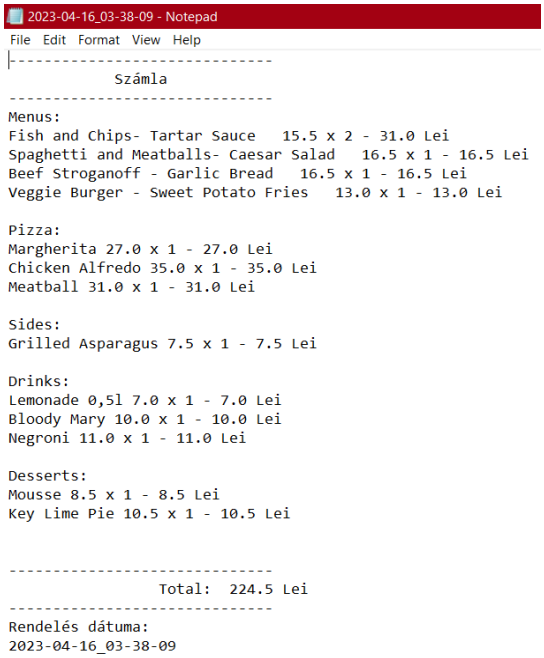
(3.2 ábra)

A **Rendelés** (3.3 ábra) gomb véglegesíti, azaz leadja a rendelést, amiután felugró ablakban megjelenik a rendelési nyugta (4.ábra), ugyanez a nyugta mentésre kerül egy txt kiterjsztésű file-ba (5.ábra).



(3.3 ábra)

(4. ábra)



(5. ábra)

A felugró ablak bezárása után a felület visszáll a kezdeti állapotba, és készen várja az új rendeléseket.

A **Kilépés** (3.4 ábra) gomb bezárja az applikációt és megszakít minden folyamatban lévő metódust.

(3.4 ábra)

Az alkalmazás működéséhez szükséges állományok ismertetése

A main.py állomány

Az alkalmazás python nyelven lett fejlesztve, python 3.11 verzió alatt, Windows 10 operációs rendszeren. A fejlesztéshez a PyCharm Community Editon fejlesztői környezetet használtam.

A program Tkinter modult használ a felhasználói felület megjelenítésére és a Python beépített JSON modulját a menü adatainak betöltésére a **menu.json** fájlból, továbbá használja a Python **datetime** modulját a dátumok és idők kezelésére. (6. ábra) Az alkalmazás tartalmaz egy **OrderApp** osztályt (7.1 ábra), amely a program fő komponenseit tartalmazza, beleértve az elemek hozzáadását és eltávolítását a rendelési listából, valamint a rendelés véglegesítését. Az osztályban definiált gombokat a felhasználói interakciók kezelésére használják és a gombok mögött a megírt metódusok dolgoznak.

Az alkalmazás két fő keretet használ: az egyik a menü megjelenítéséhez (7.2 ábra), a másik pedig a rendelési listához (7.3 ábra). A menü keretében egy **Listbox** (7.4 ábra) objektum jeleníti meg a menüt, amely a különböző kategóriákhoz színezett elemeket tartalmaz. A rendelési lista keretében egy másik **Listbox** (7.5 ábra) objektum jeleníti meg a rendelési listát, valamint három gombot, amelyek lehetővé teszik a felhasználók számára, hogy eltávolítsák az elemeket, vagy csökkentsék a mennyiségüket. A véglegesített rendelési összeget a "Rendelés" gombra kattintva jeleníti meg.

```
1  import tkinter as tk
2  import datetime
3  import json
```

(6.ábra)

```
6  class OrderApp:
7      def __init__(self, master):
8          self.master = master
9          self.master.title("2Rendelés")
10         self.master.attributes("-fullscreen", True)
```

(7.1 ábra)

```
26         menu_frame = tk.Frame(self.master, bg="white smoke", width=400)
27         menu_frame.grid(row=0, column=0, rowspan=4, sticky="nsew", padx=10, pady=10)
28         menu_label = tk.Label(menu_frame, text="Menü", font=("Courier New", 21))
29         menu_label.pack(pady=10)
```

(7.2 ábra)

```
54         self.order_listbox = tk.Listbox(order_frame, font=("Courier New", 14), width=50, bg="snow")
55         self.order_listbox.pack(pady=10)
```

(7.3 ábra)

```
49         order_frame = tk.Frame(self.master, bg="white smoke", width=300)
50         order_frame.grid(row=0, column=1, rowspan=4, sticky="nsew", padx=10, pady=10)
51         order_label = tk.Label(order_frame, text="Rendelés", font=("Courier New", 21), bg="white smoke")
52         order_label.pack(pady=10)
```

(7.4 ábra)

```
81         self.menu_listbox = tk.Listbox(menu_frame, font=("Courier New", 17), width=70, bg="snow")
```

(7.5 ábra)

A kód részlet (8. ábra) a menü listadoboz feltöltését viszi véghez. A **category_colors** nevű szótárban eltárolják a kategóriákhoz tartozó színeket (hexadecimális kódokként), például a **Menus** kategóriához az "#ADD8E6" színt. A **self.menu_listbox** változó egy listát tartalmaz, amely minden ételt és annak árát tartalmazza. A **for-ciklus** végigiterál az összes ételen, és meghatározza az étel kategóriáját. Ezután a kategória színét (amelyet a **category_colors** szótárból nyernek) eltárolják a **color** változóban. Az étel nevét és árát hozzáadják a listához, és a hozzáadott elem hátterének színét beállítják a **color** változóban tárolt értékre. Végül a listadoboz beállításai (méret, elhelyezkedés) megtörténnek.

```
    category_colors = {
        "Menus": "#ADD8E6",
        "Sides": "#90EE90",
        "Pizza": "#FFFFE0",
        "Desserts": "#FFB6C1",
        "Drinks": "#E6E6FA",
    }
    self.menu_listbox = tk.Listbox(menu_frame, font=("Courier New", 17), width=70, bg="snow")
    for item in self.menu_items:
        category = item['category']
        color = category_colors.get(category, "white")
        self.menu_listbox.insert(tk.END, f"{item['name']:.<59}{item['price']: >5} Lei")
        self.menu_listbox.itemconfig(tk.END, bg=color)
        self.menu_listbox.pack(pady=10)
    self.menu_listbox.config(height=50)
```

(8. ábra)

A gombok definiálása, tulajdonságainak megadása és a megfelelő függvénnyekkel való társítása. (9. ábra)

```
remove_button = tk.Button(order_frame, text="Elem eltávolítása", font=("Courier New", 16), width=40,
                          height=2, command=self.delete_item)
remove_button.pack(pady=10)
reduce_button = tk.Button(order_frame, text="Mennyiség csökkentése", font=("Courier New", 16), width=40,
                          height=2, command=self.decrease_quantity)
reduce_button.pack(pady=10)
place_order_button = tk.Button(order_frame, text="Rendelés", font=("Courier New", 16), width=40, height=3,
                               bg="#B7CE9E", command=self.place_order)
place_order_button.pack(pady=10)
exit_button = tk.Button(self.master, text="Kilépés", font=("Courier New", 17), width=20, height=1,
                        bg="light coral", command=self.master.destroy)
exit_button.grid(row=4, column=1, sticky="w", padx=120, pady=25)
```

(9. ábra)

A metódusok

Az add_item függvény (10. ábra) lehetővé teszi az új elemek hozzáadását a rendelési listához.

A függvény működése a következő:

- Először ellenőrzi, hogy megtörtént-e a dupla kattintás a menülista bármely elemére, és az **event** változóban tárolja a kiválasztott elemeket.
- A **selection** változóban eltárolja a kiválasztott elemek listáját.Az **item** változóban eltárolja az első kiválasztott menüelemet, azaz annak nevét és árát.
- A függvény ellenőrzi, hogy az új elem már szerepel-e a rendelési listában.
- Ha igen, akkor a **quantity** értékét növeli, frissíti az árakat, majd frissíti a rendelési listát.
- Ha az új elem még nem szerepel a rendelési listában, akkor hozzáadja az új elemet a rendelési listához, a **quantity** értékét 1-re állítja, és frissíti az árakat.
- Az új elemeket a rendelési listában a **order_listbox** listába adja, amely a kezelőfelületen jelenik meg.

```
def add_item(self, event=None):
    if event:
        selection = event.widget.curselection()
    else:
        selection = self.menu_listbox.curselection()

    if selection:
        item = self.menu_items[selection[0]]
        name = item['name']
        price = item['price']
        item_found = False
        for i in range(self.order_listbox.size()):
            if self.order_listbox.get(i).startswith(name):
                item['quantity'] += 1
                self.total_price += price
                self.order_listbox.delete(0, tk.END)
                for item in self.order:
                    name = item['name']
                    price = item['price']
                    quantity = item['quantity']
                    new_price = quantity * price
                    self.order_listbox.insert(tk.END, f"{name} x {quantity} - {new_price} Lei")
                item_found = True
                break
        if not item_found:
            item['quantity'] = 1
            self.order_listbox.insert(tk.END, f"{name} x 1 - {price} Lei")
            self.order.append(item)
            self.total_price += price
```

(10. ábra)

A delete_item függvény (11. ábra) segítségével a felhasználók törölhetnek a rendelési listából egyes elemeket.
A függvény működése a következő:

- Először ellenőrzi, hogy a felhasználó kiválasztott-e egy elemet a rendelési listából.
- A **selection** változóban eltárolja a kiválasztott elem indexét a rendelési listában.Az **item** változóban eltárolja az adott elemet, amelyet a felhasználó törölni kíván.
- Az **item** változó segítségével kiszámítja az eltávolított elem árát és mennyiségét, majd levonja ezt az összeget a **total_price** változóból.
- Az eltávolítandó elemet törli a rendelési listából.
- Végül az eltávolított elem **quantity** értékét nullára állítja.
- Ez az ***add_item*** függvénnyel ellentétben eltávolítja a dolgozók által hozzáadott elemeket a rendelési listából. Összességében tehát a ***delete_item*** függvény lehetővé teszi a dolgozók számára, hogy korrigálják a rendelésüket, ha szükséges, és eltávolítsák az olyan tételeket, amelyeket esetleg véletlenül adtak hozzá a rendelési listához.

```
def delete_item(self):
    selection = self.order_listbox.curselection()
    if selection:
        item = self.order[selection[0]]
        self.total_price -= item['price'] * item['quantity']
        self.order_listbox.delete(selection[0])
        self.order.remove(item)
        item['quantity'] = 0
```

(11. ábra)

A decrease_quantity függvény (12. ábra) segítségével a dolgozók csökkenthetik az egyes elemek mennyiségét a rendelési listában.
A függvény működése a következő:

- Először ellenőrzi, hogy a felhasználó kiválasztott-e egy elemet a rendelési listából.
- A **selection** változóban eltárolja a kiválasztott elem indexét a rendelési listában.

- Az **item** változóban eltárolja az adott elemet, amelynek a mennyiségét a felhasználó csökkenteni kívánja.
- Az **item** változó segítségével csökkenti az adott elem **quantity** értékét 1-gyel.Ha az elem **quantity** értéke 0, akkor törli az elemet a rendelési listából és az order listából is, majd levonja az elem árát a **total_price** változóból.
- Ha az elem **quantity** értéke nem 0, akkor kiszámítja az új árat, majd eltávolítja az eredeti elemet a rendelési listából, és újra beszúrja az elemet a rendelési listába az új mennyiség és ár információkkal.
- Végül levonja az elem eredeti árát a **total_price** változóból.
- Ez a függvény lehetővé teszi a dolgozók számára, hogy korrigálják a rendelésüeket, ha véletlenül túl sokat adtak hozzá az egyes tételekből a rendelési listához, vagy ha a kliensek úgy döntenek, hogy kevesebb mennyiséget kérnek az adott tételből.

```
def decrease_quantity(self):
    selection = self.order_listbox.curselection()
    if selection:
        item = self.order[selection[0]]
        name = item['name']
        price = item['price']
        item['quantity'] -= 1
        quantity = item['quantity']
        if quantity == 0:
            self.order_listbox.delete(selection[0])
            self.order.remove(item)
            self.total_price -= price
        else:
            new_price = quantity * price
            self.order_listbox.delete(selection[0])
            self.order_listbox.insert(selection[0], f"{name} x {quantity} - {new_price} Lei")
            self.total_price -= price
```

(12. ábra)

A place_order függvény (13. ábra) segítségével a dolgozók leadhatják a felvett rendelést és elkészül a számla. ***A függvény működése a következő:***

- Először ellenőrzi, hogy van-e rendelés az **order** listában. Ha nincs, akkor nem megy tovább.
- Különben összegyűjti az egyes elemeket a rendelési listában csoportokba, aszerint, hogy melyik kategóriához tartoznak. Erre egy **grouped_order** nevű dictionary-t használ.

- Létrehoz egy **timestamp** változót, amely a rendelés időpontját tartalmazza.Létrehoz egy **order_items** és egy **order_receipt** stringet.
- Végigmegy a **grouped_order** dictionary-n, és az egyes csoportok nevét és elemeit hozzáadja az **order_items** stringhez.
- Az **order_receipt** string tartalmazza a számlaüzenetet, amely az összesített megrendelést, az árakat és az időbélyeget tartalmazza.
- Megjeleníti a számlát egy felugró ablakban.Létrehoz egy szöveges fájlt, amelyben elmenti a számlát.
- Törli a rendelést az **order** listából, az egyes elemek mennyiségét visszaállítja nullára, és törli azokat a megrendelési lista felületéről.

```
def place_order(self):
    if not self.order:
        return
    grouped_order = {}
    for item in self.order:
        if item['category'] not in grouped_order:
            grouped_order[item['category']] = []
        grouped_order[item['category']].append(item)

    timestamp = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    order_items = ""
    order_receipt = ""
    for category, items in grouped_order.items():
        order_items += f"{category.capitalize()}:\n"
        for item in items:
            order_items += f"{item['name']} " \
                           f"{item['price']} x {item['quantity']} - {item['price'] * item['quantity']} Lei\n"
        order_items += "\n"
        order_receipt = f"{'-' * 30}\n{'Számla':^30}\n{'-' * 30}\n{order_items}\n{'-' * 30}\n{'Total:':>23}" \
                        f"{self.total_price:>7} Lei\n{'-' * 30}\n{'Rendelés dátuma:':<17}" \
                        f"\n{timestamp:>13}\n{'-' * 30}"
    receipt_window = tk.Toplevel(self.master)
    receipt_window.title("Rendelési nyugta")
    receipt_window.geometry("400x500")
    receipt_label = tk.Label(receipt_window, text=order_receipt)
    receipt_label.pack(fill=tk.BOTH, expand=True)
    with open(f"{timestamp}.txt", "w", encoding="utf-8") as f:
        f.write(order_receipt)
    self.order = []
    self.total_price = 0
    for item in self.menu_items:
        item['quantity'] = 0
    self.order_listbox.delete(0, tk.END)
```

(13. ábra)

A program futásáért felelős kódrészlet
(14. ábra)

- ***Elindítja az*** alkalmazást, amelynek felülete egy Tkinter ablakon jelenik meg.
- Az if name == "main": egy olyan blokk kezdete, amely csak akkor fut le, ha ezt a fájlt közvetlenül futtatjuk, és nem importáljuk egy másik Python fájlból.
- Ezután a kód létrehoz egy Tkinter Tk objektumot, ami az ablakunkat jel***enti***. A root.configure(bg="white smoke") metódus hátteret állít be ***az*** ablakunk számára.
- ***Az*** *OrderApp* ***objektum*** kapcsolódik az ablakhoz, amit a root változó tartalmaz.
- Végül a root.mainloop() metódus elindítja a Tkinter ablakot, és megjeleníti az alkalmazást, majd a program ***aktív*** marad, amíg az ablak bezárása nem történik meg.

```
if __name__ == "__main__":
    root = tk.Tk()
    root.configure(bg="white smoke")
    app = OrderApp(root)
    root.mainloop()
```

(14. ábra)

A menu.json állomány

A menu.json (15. ábra) fájl egy adatstruktúrát reprezentál, amely menüelemeket tartalmaz egy étteremben. A fájl tartalmazza a menüpontok kategóriáját, nevét és árát. A Python programozási nyelv lehetővé teszi számunkra, hogy ezt az adatstruktúrát könnyen feldolgozzuk és változtassuk.


```
[
  { "category": "Menus", "name": "Hamburger - Fries  ", "price": 12.0 },
  { "category": "Menus", "name": "Hot Dog - Chips  ", "price": 10.0 },
  { "category": "Menus", "name": "Chicken Sandwich - Onion Rings  ", "price": 13.5 },
  { "category": "Menus", "name": "Grilled Cheese w Tomato Soup  ", "price": 11.0 },
  { "category": "Menus", "name": "Pulled Pork Sandwich- Coleslaw  ", "price": 14.0 },
  { "category": "Menus", "name": "Fish and Chips- Tartar Sauce  ", "price": 15.5 },
  { "category": "Menus", "name": "Meatball Sub- Garlic Bread  ", "price": 13.0 },
  { "category": "Menus", "name": "Spaghetti and Meatballs- Caesar Salad  ", "price": 16.5 },
  { "category": "Menus", "name": "Beef Tacos - Chips and Salsa  ", "price": 14.5 },
  { "category": "Menus", "name": "Burrito - Guacamole  ", "price": 12.0 },
  { "category": "Menus", "name": "Stir Fry - Rice  ", "price": 15.0 },
  { "category": "Menus", "name": "Pad Thai - Spring Rolls  ", "price": 17.0 },
  { "category": "Menus", "name": "Sushi Roll Combo - Miso Soup  ", "price": 19.0 },
  { "category": "Menus", "name": "Pizza Slice - Caesar Salad  ", "price": 11.0 },
  { "category": "Menus", "name": "Fried Chicken - Coleslaw  ", "price": 14.5 },
  { "category": "Menus", "name": "BBQ Ribs - Baked Beans  ", "price": 18.0 },
  { "category": "Menus", "name": "Grilled Salmon - Roasted Vegetables  ", "price": 20.0 },
  { "category": "Menus", "name": "Beef Stroganoff - Garlic Bread  ", "price": 16.5 },
  { "category": "Menus", "name": "Veggie Burger - Sweet Potato Fries  ", "price": 13.0 },
  { "category": "Menus", "name": "Fajitas - Chips and Guacamole  ", "price": 17.0 },
  { "category": "Pizza", "name": "Margherita", "price": 27.0 },
  { "category": "Pizza", "name": "Pepperoni", "price": 30.0 },
  { "category": "Pizza", "name": "Hawaiian", "price": 32.0 },
  { "category": "Pizza", "name": "Mushroom", "price": 28.0 },
  { "category": "Pizza", "name": "Sausage", "price": 29.0 },
  { "category": "Pizza", "name": "Meat Lovers", "price": 35.0 },
  { "category": "Pizza", "name": "BBQ Chicken", "price": 32.0 },
  { "category": "Pizza", "name": "Veggie", "price": 30.0 },
```


(15. ábra)

A JSON fájlt betöltjük egy Python scriptbe és eltároljuk az adatokat egy listában. Az adatok elérése érdekében lehetőségünk van a lista elemeinek indexelésére, vagy akár az adatok szűrésére is a kategóriák vagy az árak alapján.(16. ábra)

```python
with open("menu.json") as f:
    self.menu_items = json.load(f)
    for item in self.menu_items:
        item['quantity'] = 0
```

(16. ábra)

Bibliográfia

W3schools.com:
https://www.w3schools.com/python/
https://www.w3schools.com/python/python_json.asp
https://www.w3schools.com/python/python_datetime.asp
Studytonight.com:
https://www.studytonight.com/tkinter/introduction-to-python-tkinter-module#google_vignette
Tutorialspoint.com:
https://www.tutorialspoint.com/python/python_gui_programming.htm
Docs.python.org:
https://docs.python.org/3/library/tkinter.html
https://docs.python.org/3/library/datetime.html
Copyassingment.com:
https://copyassignment.com/restaurant-management-system-project-in-python/
W3resource.com:
https://www.w3resource.com/python-exercises/class-exercises/python-class-real-life-problem-2.php
Github.com:
https://github.com/amark23/Restaurant-Management-System-Python-