Содержание

Введение..............................................................................................................2

1. Постановка задачи..........................................................................................3

2. Описание используемого стека технологий................................................4

2.1 PYTHON……………………………………………………………..4

2.1.1 ДЛЯ ЧЕГО НУЖЕН………………………………………..4

2.1.2 ОСОБЕННОСТИ ПРОГРАММИРОВАНИЯ…………….5

2.1.3 ПРЕИМУЩЕСТВА………………………………………..5

2.1.4 НЕДОСТАТКИ…………………………………………….6

2.1.5 ИСТОРИЯ…...……………………………………………...6

2.2 PYTHON TKINTER………………………………………………...7

3. Описание приложения...................................................................................12

3.1 Добавление блюд в меню…………………………………………..13

3.2 Добавление блюд в заказ…………………………………………...14

3.3 Удаление блюда из заказа……………………………….................15

3.4 Обновление списка заказа………………………………………….15

3.5 Выбор категории блюд……………………………………………..15

4. Графический интерфейс…………………………………………………….17

4.1 Окно приложения……………………………………………………17

4.2 Панель Меню………………………………………………………..17

4.3 Панель Заказа………………………………………………………..19

Заключение..........................................................................................................22

Список использованной литературы.................................................................23

Исходный код консольного приложения.........................................................

ВВЕДЕНИЕ

Современные технологии играют все более значимую роль в различных сферах жизни, включая индустрию общественного питания. Создание программного обеспечения для автоматизации и оптимизации работы ресторанов стало важной задачей для разработчиков. В данной работе рассматривается процесс разработки программы на языке Python, предназначенной для симуляции работы ресторана.

После появление интернета нам больше не нужно лично ходить и бронировать места и меню в ресторанах, теперь требуется лишь хороший интернет и устройство, позволяющее выйти в сеть. Это позволит не выходя из дома заказать себе в любом доступном ресторане понравившееся блюдо.

По заданию требуется написать программу, реализующую простую работу ресторана.

В отличии от программ, где будет указываться время заказа, столик который заказывал, и прочих нюансов, у нас будет только меню, блюдо и заказ.

В качестве стека был выбран следующий состав:

1. Бэкенд: язык программирования Python

2. Библиотека “tkinter” для реализации графического интерфейса

1. ПОСТАНОВКА ЗАДАЧИ

Цель учебной практики – разработать программу на Python для симуляции работы ресторана с классами “Меню”, “Блюдо”, “Заказ”.

В ходе разработки программы должны быть выполнены несколько основных задач:

1. Определение меню.

Данная задача подразумевает определение доступных блюд и их цен в ресторане.

2. Принятие заказа.

Данная задача подразумевает, что пользователь делает заказ, указывая количество каждого выбранного блюда.

3. Расчет стоимости.

Данная задача подразумевает умножение цен каждого блюда на количество заказанных порций, чтобы определить общую стоимость заказа.

4. Формирование отчета.

Данная задача подразумевает генерирование отчета, который включает в себя информацию о заказанных блюдах, количестве каждого из них и общей стоимости заказа.

5. Разработка графического интерфейса

2. ОПИСАНИЕ ИСПОЛЬЗУЕМОГО СТЕКА ТЕХНОЛОГИЙ

2.1 PYTHON

Для того, чтобы создавать программное обеспечение, требуется создавать специальные коды. Они пишутся на языках программирования. Одним из вариантов является Python. Выступает в качестве высокоуровневого «способа общения» с ПО и устройствами. Помогает решать задачи различной сложности и назначения. Изучить его с самого начала способен даже тот, кто далек от IT-сферы.

Python – язык программирования высокого уровня общего назначения. Обладает типизацией динамического строгого характера. Имеет автоматическое управление памятью, за счет чего осуществляется повышение производительности контента, написанного на нем.

Python – объектно-ориентированный язык программирования, пользующийся спросом у большинства современных разработчиков. Коды, написанные на нем, достаточно легко читать.

2.1.1 ДЛЯ ЧЕГО НУЖЕН

Python – язык, который является универсальным. С его помощь можно писать:

1. «быстрые» коды – когда нужно, чтобы что-то заработало [1] в кратчайшие сроки;

2. контент для коммерции, науки или анализа информации;

3. игры;

4. веб-утилиты;

5. прототипы.

Подойдет и для сложного программного обеспечения, и для небольших утилит/виджетов. Главное освоить Python с нуля.

2.1.2 ОСОБЕННОСТИ ПРОГРАММИРОВАНИЯ

Изучить Python сможет каждый, кто изъявит такое желание. Этот «способ общения» пользуется спросом, предлагая немало [1] возможностей и библиотек.

Потенциальный программист должен запомнить следующие особенности создания программного кода на Python:

1. Наличие динамической типизации;

2. Возможность удобного возврата нескольких значений функций;

3. Использование сборщика мусора;

4. Автоматическое выделение памяти в устройстве – за счет этого программа будет работать быстрее;

5. Наличие привязки типа информации;

6. Удобство применения цикла For;

7. Возможность интерпретации – не требует компиляции для получения того или иного результата.

Стоит обратить внимание на то, что Python – это простой способ создания программного кода в сочетании с весьма мощными инструментами разработчика.

2.1.3 ПРЕИМУЩЕСТВА

Программировать на Python не слишком трудно, если изучить основы данного варианта. Язык выделяется на фоне остальных следующими преимуществами:

- компактность;

- высокая читаемость получающегося программного кода;

- схожесть синтаксиса с английским языком;

- отсутствие необходимости применения фигурных скобок и точки с запятой в кодификации;

- универсальность;

- наличие ООП;

- легкое обозначение вложенности блоков утилиты, которое повышает читаемость.

В Google можно отыскать информацию о том, насколько Python популярен. Его выбирают не только новички, но и продвинутые разработчики. Особенно на первых порах, когда требуется изучить основы создания программного обеспечения [1].

2.1.4 НЕДОСТАТКИ

Перед тем, как начинать изучение Python, стоит уточнить его минусы. В Google полно полезной информации на данный счет. Python обладает следующими недостатками:

- относительно медленная работа по сравнению с СИ-семейством;

- отсутствие строгости – это и плюс, и минус (не всем легко сразу «перестроиться» под Python).

Программировать с его помощью легко и просто. Если выучить Python, удастся в будущем не только создавать универсальные сложные утилиты, но и достаточно быстро осваивать СИ-семейство.

2.1.5 ИСТОРИЯ

История языка программирования Python началась в конце 1980-х. Гвидо ван Россум задумал Python в 1980-х годах, а приступил к его созданию в декабре 1989 года в центре математики и информатики в Нидерландах. Язык Python был задуман как потомок языка программирования ABC, способный к обработке исключений и взаимодействию с операционной системой Amoeba. Ван Россум является основным автором Python и продолжал выполнять центральную роль в принятии решений относительно развития языка вплоть до 12 июля 2018 года.

Версия Python 2.0 была выпущена 16 октября 2000 года и включала в себя много новых крупных функций — таких как полный сборщик мусора и поддержка Unicode. Однако наиболее важным из всех изменений было изменение самого процесса развития языка и переход на более прозрачный процесс его создания.

Первая обратно-несовместимая версия Python 3.0 была выпущена 3 декабря 2008 года после длительного периода тестирования. Многие её функции были портированы и обратно совместимы с Python 2.6 и Python 2.7 [2].

2.2 PYTHON TKINTER

В Python имеется достаточно большое количество GUI фреймворков, тем не менее, лишь Tkinter включен в состав стандартной библиотеки языка. Tkinter обладает целым рядом достоинств. Он является кроссплатформенным, и это означает, что один и тот же код может быть использован на Windows, macOS и Linux. Визуальные компоненты могут отображаться через собственный набор элементов текущей операционной систем, то есть, программные приложения, которые созданы при помощи Tkinter, будут выглядеть так, как будто они принадлежат той платформе, на которой они работают [3].

Хотя Tkinter является популярным GUI фреймворком на Python, у него есть свои недостатки. Один из них заключается в том, что графические интерфейсы, созданные с использованием Tkinter, выглядят устаревшими. Если пользователю нужен современный, броский интерфейс, то Tkinter может оказаться не совсем тем, для этого есть PyQt5 который развивается сильнее в данном плане [3].

Тем не менее в плане использования, Tkinter является относительно легким по сравнению с другими библиотеками. Это отличный выбор для создания GUI приложений в Python, особенно если современный облик не в приоритете для программы, а большую роль играет функциональность и кроссплатформенная скорость.

Python Tkinter - это библиотека, предоставляющая интерфейс для создания графического пользовательского интерфейса (GUI) в Python. С помощью Tkinter можно создавать окна, кнопки, ползунки, текстовые поля и другие виджеты, которые позволяют взаимодействовать с пользователем. Tkinter предоставляет простые и интуитивно понятные методы для управления виджетами и обработки событий.

Простой пример использования Tkinter представлен на рисунке 1:

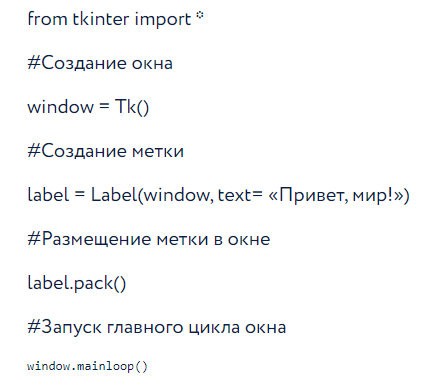


Рисунок 1 – пример использования Tkinter

В этом примере выполнено создание окна с помощью конструктора Tk(). Затем создан виджет Label с текстом «Привет, мир!» и размещаем его в окне с помощью метода pack(). Наконец, выполнен запуск главного цикла окна с помощью метода mainloop(), чтобы окно оставалось видимым и пользователь мог с ним взаимодействовать.

Tkinter также предоставляет возможность обрабатывать события, такие как нажатие кнопки, ввод текста и другие. Для этого можно использовать методы привязки событий к функциям обратного вызова. Tkinter является стандартной библиотекой Python, поэтому она обычно уже установлена вместе с Python. Она довольно проста в использовании и хорошо подходит для создания простых GUI-приложений и прототипирования интерфейсов.

В Tkinter также есть возможность создавать различные виджеты, такие, как кнопки, ползунки, текстовые поля и другие. Ниже на рисунке 1.2 приведены примеры использования некоторых из них:

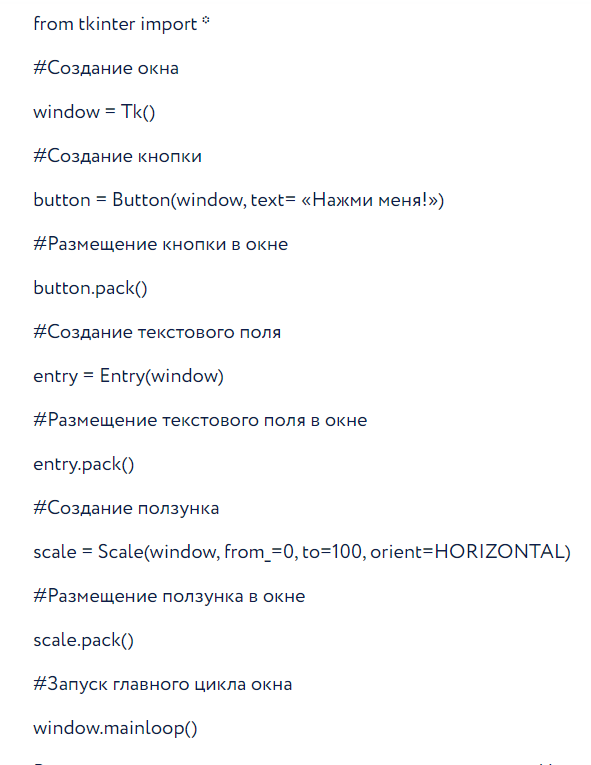


Рисунок 1.2 – пример создания виджетов

В этом примере выполнено создание кнопки с текстом «Нажми меня!», текстового поля и ползунка. Затем осуществлено размещение их в окне и выполнен запуск главного цикла окна.

Кнопки могут иметь функции обратного вызова, которые выполняются при нажатии на кнопку. На рисунке 1.3 приведен пример использования функции обратного вызова:

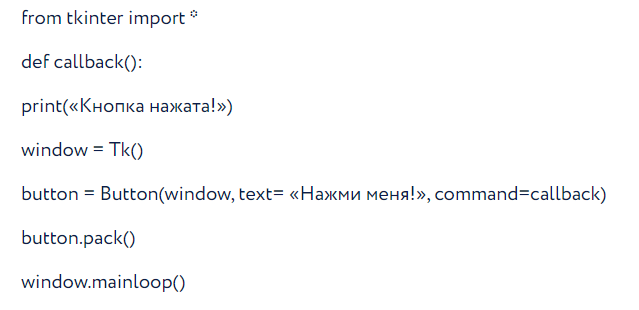


Рисунок 1.3 – пример использования функции обратного вызова

В этом примере функция callback() будет вызвана при нажатии на кнопку. В данном случае, функция просто выводит сообщение в консоль. Tkinter также предоставляет возможность создавать меню, диалоговые окна, таблицы и многое другое.

3. ОПИСАНИЕ ПРОГРАММЫ

Программа представляет собой симуляцию работы ресторана с использованием графического интерфейса на базе библиотеки “tkinter” в Phyton.

В программе реализованы основные компоненты ресторана: меню с различными категориями блюд, возможность добавления блюд в заказ и отображение текущего заказа с подсчетом общей стоимости.

Данная программа включает в себя несколько классов: класс “Меню” (Menu), класс “Блюдо” (Dish) и класс “Заказ” (Order).

Класс “Меню” (Menu) отвечает за хранение и управление меню ресторана, разбитого на категории блюд: Первое, Второе, Салаты, Напитки и Десерты.

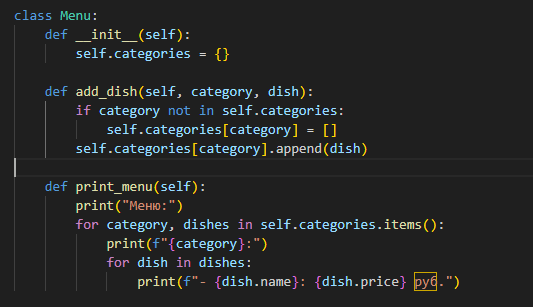


Рис. 1 -код класса “Меню”

Класс “Блюдо” представляет отдельное блюдо в меню.

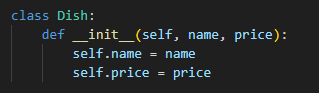


Рис. 2 – код класса “Блюдо”

Класс “Заказ” отвечает за хранение и управление текущим заказом.

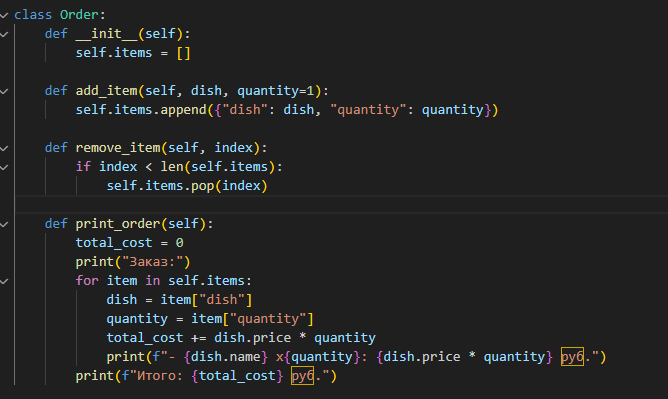


Рис. 3 – код класса “Заказ”

Разработанная программа также имеет следующие функции:

– Добавление блюд в меню

– Добавление блюд в заказ

– Удаление блюда из заказа

– Обновление списка заказа

– Выбор категории блюд

3.1 ДОБАВЛЕНИЕ БЛЮД В МЕНЮ

Добавление блюд в меню производится вручную в коде программы



Рис. 4 – добавление блюд в меню

3.2 ДОБАВЛЕНИЕ БЛЮД В ЗАКАЗ

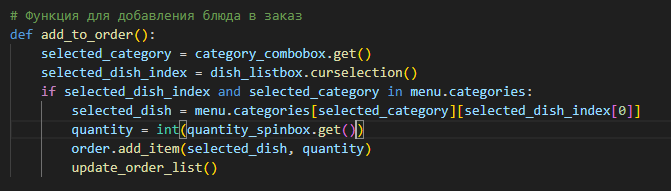


Рис. 4 – код функции добавления блюда в заказ

3.3 УДАЛЕНИЕ БЛЮДА ИЗ ЗАКАЗА

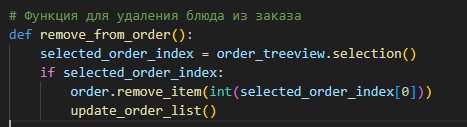


Рис. 5 – код функции удаления блюда из заказа

3.4 ОБНОВЛЕНИЕ СПИСКА ЗАКАЗА

Обновление списка происходит автоматически при добавлении или удалении блюда в заказе.

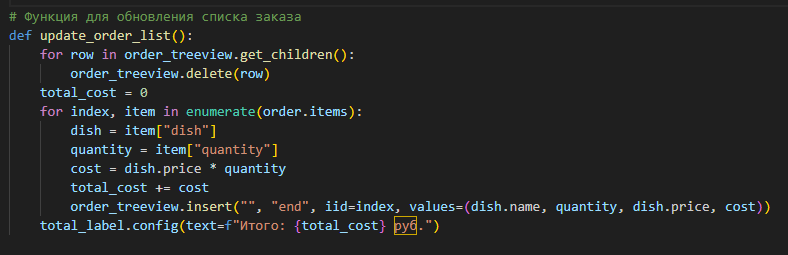


Рис. 6 – код функции обновления списка заказа

3.5 ВЫБОР КАТЕГОРИИ БЛЮД

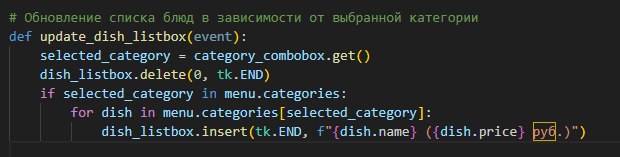


Рис. 7 – код функции выбора категории блюд

4. ГРАФИЧЕСКИЙ ИНТЕРФЕЙС

Графический интерфейс создан с использованием библиотеки “tkinter” в Python. Он позволяет пользователям просматривать меню, выбирать блюда, указывать их количество и добавлять в заказ. Также имеется возможность удалить выбранные блюда из заказа и видеть общую стоимость заказа.

Интерфейс разделен на три основные части: окно приложения, панель Меню и панель Заказа.

4.1 Окно приложения

Главное окно приложения имеет заголовок “Меню ресторана”.

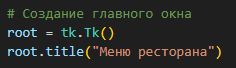


Рис. 8 – код главного окна приложения

4.2 Панель Меню

Панель Меню расположена в левой части окна и предназначена для отображения доступных категорий блюд и самих блюд в выбранной категории. Данная панель содержит следующие основные элементы:

1. Combobox – выпадающий список, который позволяет пользователю выбрать определенную категорию блюд.

2. Listbox – отображает список блюд из выбранной категории.

3. Spinbox – позволяет пользователю указать количество выбранного блюда

4. Кнопка добавления в заказ

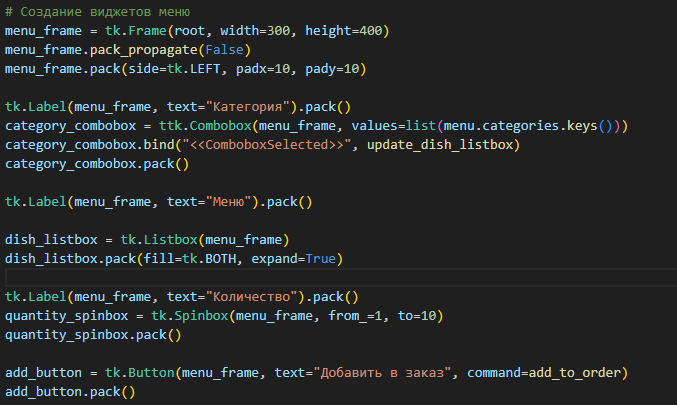


Рис. 9 – код панели Меню

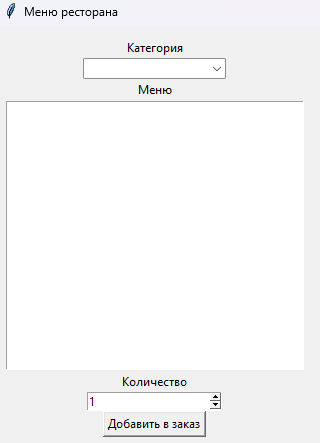


Рис. 10 – панель Меню

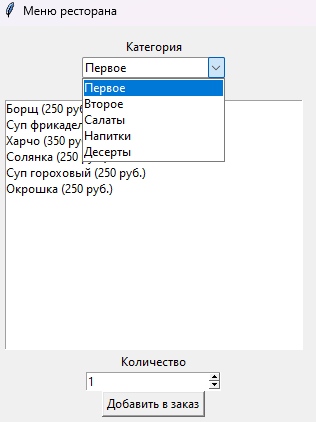


Рис. 11 – панель Меню с выбором категории

4.3 Панель Заказа

Панель заказа расположена в правой части окна и предназначена для отображения текущего заказа пользователя. Данная панель содержит следующие основные элементы:

1. Treeviev – виджет, который предназначен для отображения текущего заказа в виде таблицы. Таблица состоит из четырех столбцов: Наименование, Количество, Цена за единицу и Итоговая цена.

2. Строка для общей стоимости

3. Кнопка для удаления из заказа

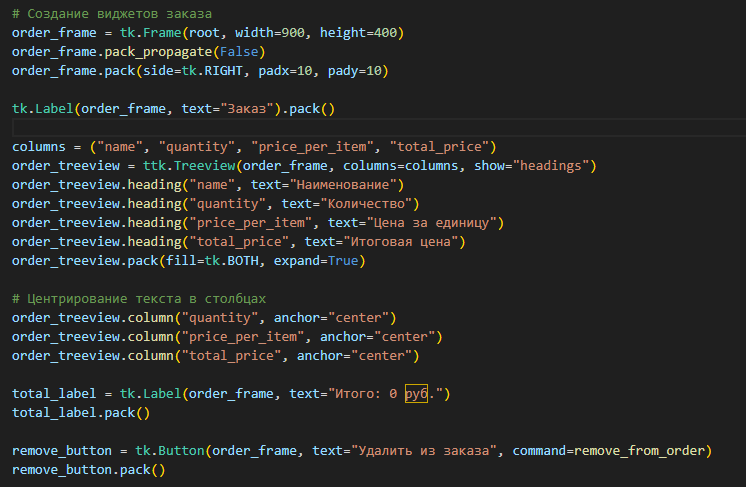


Рис. 12 – код панели Заказ

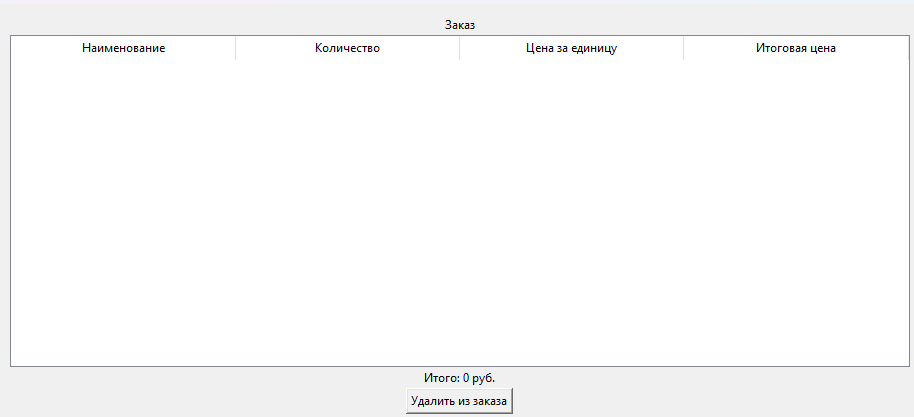


Рис. 13 – панель Заказа

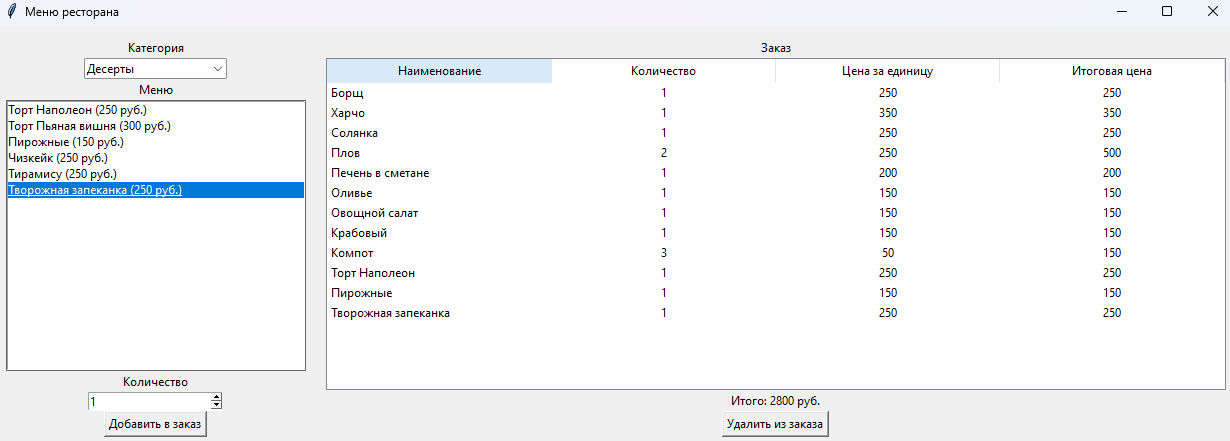


Рис. 14 – вид приложения Меню ресторана

ЗАКЛЮЧЕНИЕ

В результате работы была разработана программа на языке программирования Python, позволяющая представить работу ресторана с использованием графического интерфейса.

Все поставленные задачи были выполнены. Приложение позволяет:

– Добавлять блюда в заказ;

– Выбирать категорию блюд;

– Удалять некоторые блюда из заказа.

В дальнейшем возможно:

– Улучшение интерфейса;

– Добавление возможности изменения меню в окне приложения;

– Добавление возможности изменения количества заказанных блюд без удаления блюд;

– Добавление функции выбора “Забрать заказ с собой” и “Остаться в ресторане”.

Полные исходные тексты приложения опубликованы в качестве проекта с открытым исходным кодом, и доступны в Git-репозитории по адресу: <https://github.com/Ivan856-bit/->

Список используемой литературы

1. Информация о Python: <https://otus.ru/journal/python-vse-o-yazyke-programmirovaniya/#Определение> (дата обращения 18.07.24).

2. История Python: <https://ru.wikipedia.org/wiki/История_языка_программирования_Python>.

3. Библиотека tkinter:

<https://spravochnick.ru/informatika/python_tkinter/#:~:text=Python%20Tkinter%20-%20это%20библиотека%2C,управления%20виджетами%20и%20обработки%20событий>

ИСХОДНЫЙ КОД ПРОГРАММЫ

Здесь расположен листинг программы

Листинг 1. Код программы

import tkinter as tk

from tkinter import messagebox, ttk

class Menu:

def \_\_init\_\_(self):

self.categories = {}

def add\_dish(self, category, dish):

if category not in self.categories:

self.categories[category] = []

self.categories[category].append(dish)

def print\_menu(self):

print("Меню:")

for category, dishes in self.categories.items():

print(f"{category}:")

for dish in dishes:

print(f"- {dish.name}: {dish.price} руб.")

class Dish:

def \_\_init\_\_(self, name, price):

self.name = name

self.price = price

class Order:

def \_\_init\_\_(self):

self.items = []

def add\_item(self, dish, quantity=1):

self.items.append({"dish": dish, "quantity": quantity})

def remove\_item(self, index):

if index < len(self.items):

self.items.pop(index)

def print\_order(self):

total\_cost = 0

print("Заказ:")

for item in self.items:

dish = item["dish"]

quantity = item["quantity"]

total\_cost += dish.price \* quantity

print(f"- {dish.name} x{quantity}: {dish.price \* quantity} руб.")

print(f"Итого: {total\_cost} руб.")

# Создание меню

menu = Menu()

menu.add\_dish("Первое", Dish("Борщ", 250))

menu.add\_dish("Первое", Dish("Суп фрикадельковый", 250))

menu.add\_dish("Первое", Dish("Харчо", 350))

menu.add\_dish("Первое", Dish("Солянка", 250))

menu.add\_dish("Первое", Dish("Суп гороховый", 250))

menu.add\_dish("Первое", Dish("Окрошка", 250))

menu.add\_dish("Второе", Dish("Паста", 250))

menu.add\_dish("Второе", Dish("Овощное рагу", 300))

menu.add\_dish("Второе", Dish("Запеканка", 250))

menu.add\_dish("Второе", Dish("Ребрышки", 350))

menu.add\_dish("Второе", Dish("Плов", 250))

menu.add\_dish("Второе", Dish("Печень в сметане", 200))

menu.add\_dish("Салаты", Dish("Оливье", 150))

menu.add\_dish("Салаты", Dish("Цезарь", 200))

menu.add\_dish("Салаты", Dish("Крабовый", 150))

menu.add\_dish("Салаты", Dish("Овощной салат", 150))

menu.add\_dish("Напитки", Dish("Вода", 50))

menu.add\_dish("Напитки", Dish("Кола", 80))

menu.add\_dish("Напитки", Dish("Компот", 50))

menu.add\_dish("Напитки", Dish("Кофе", 50))

menu.add\_dish("Напитки", Dish("Молочный коктейль", 50))

menu.add\_dish("Десерты", Dish("Торт Наполеон", 250))

menu.add\_dish("Десерты", Dish("Торт Пьяная вишня", 300))

menu.add\_dish("Десерты", Dish("Пирожные", 150))

menu.add\_dish("Десерты", Dish("Чизкейк", 250))

menu.add\_dish("Десерты", Dish("Тирамису", 250))

menu.add\_dish("Десерты", Dish("Творожная запеканка", 250))

# Функция для добавления блюда в заказ

def add\_to\_order():

selected\_category = category\_combobox.get()

selected\_dish\_index = dish\_listbox.curselection()

if selected\_dish\_index and selected\_category in menu.categories:

selected\_dish = menu.categories[selected\_category][selected\_dish\_index[0]]

quantity = int(quantity\_spinbox.get())

order.add\_item(selected\_dish, quantity)

update\_order\_list()

# Функция для удаления блюда из заказа

def remove\_from\_order():

selected\_order\_index = order\_treeview.selection()

if selected\_order\_index:

order.remove\_item(int(selected\_order\_index[0]))

update\_order\_list()

# Функция для обновления списка заказа

def update\_order\_list():

for row in order\_treeview.get\_children():

order\_treeview.delete(row)

total\_cost = 0

for index, item in enumerate(order.items):

dish = item["dish"]

quantity = item["quantity"]

cost = dish.price \* quantity

total\_cost += cost

order\_treeview.insert("", "end", iid=index, values=(dish.name, quantity, dish.price, cost))

total\_label.config(text=f"Итого: {total\_cost} руб.")

# Обновление списка блюд в зависимости от выбранной категории

def update\_dish\_listbox(event):

selected\_category = category\_combobox.get()

dish\_listbox.delete(0, tk.END)

if selected\_category in menu.categories:

for dish in menu.categories[selected\_category]:

dish\_listbox.insert(tk.END, f"{dish.name} ({dish.price} руб.)")

# Создание главного окна

root = tk.Tk()

root.title("Меню ресторана")

# Создание виджетов меню

menu\_frame = tk.Frame(root, width=300, height=400)

menu\_frame.pack\_propagate(False)

menu\_frame.pack(side=tk.LEFT, padx=10, pady=10)

tk.Label(menu\_frame, text="Категория").pack()

category\_combobox = ttk.Combobox(menu\_frame, values=list(menu.categories.keys()))

category\_combobox.bind("<<ComboboxSelected>>", update\_dish\_listbox)

category\_combobox.pack()

tk.Label(menu\_frame, text="Меню").pack()

dish\_listbox = tk.Listbox(menu\_frame)

dish\_listbox.pack(fill=tk.BOTH, expand=True)

tk.Label(menu\_frame, text="Количество").pack()

quantity\_spinbox = tk.Spinbox(menu\_frame, from\_=1, to=10)

quantity\_spinbox.pack()

add\_button = tk.Button(menu\_frame, text="Добавить в заказ", command=add\_to\_order)

add\_button.pack()

# Создание виджетов заказа

order\_frame = tk.Frame(root, width=900, height=400)

order\_frame.pack\_propagate(False)

order\_frame.pack(side=tk.RIGHT, padx=10, pady=10)

tk.Label(order\_frame, text="Заказ").pack()

columns = ("name", "quantity", "price\_per\_item", "total\_price")

order\_treeview = ttk.Treeview(order\_frame, columns=columns, show="headings")

order\_treeview.heading("name", text="Наименование")

order\_treeview.heading("quantity", text="Количество")

order\_treeview.heading("price\_per\_item", text="Цена за единицу")

order\_treeview.heading("total\_price", text="Итоговая цена")

order\_treeview.pack(fill=tk.BOTH, expand=True)

# Центрирование текста в столбцах

order\_treeview.column("quantity", anchor="center")

order\_treeview.column("price\_per\_item", anchor="center")

order\_treeview.column("total\_price", anchor="center")

total\_label = tk.Label(order\_frame, text="Итого: 0 руб.")

total\_label.pack()

remove\_button = tk.Button(order\_frame, text="Удалить из заказа", command=remove\_from\_order)

remove\_button.pack()

# Создание заказа

order = Order()

# Запуск главного цикла программы

root.mainloop()