

Колледж космического машиностроения и технологий

КУРСОВОЙ ПРОЕКТ

По МДК.01.02 «Прикладное программирование»

Тема: «Разработка приложения «Клиент спортивного магазина»

Выполнил студент

Морозов Алексей Андреевич

Группа П2-19

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Проверил преподаватель

Гусятинер Леонид Борисович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

Королев 2022 г.

**Оглавление**

[**Введение** 3](#_Toc107348718)

[**1.** **Теоретическая часть** 4](#_Toc107348719)

[**1.1.** **Описание предметной области** 4](#_Toc107348720)

[**1.2.** **Описание существующих разработок** 9](#_Toc107348721)

[**1.2.1.** **Спортмастер.** 9](#_Toc107348722)

[**1.2.2.** **Кант** 11](#_Toc107348723)

[**2.1.** **Диаграмма прецедентов** 13](#_Toc107348724)

[**2.2.** **Выбор инструментов** 14](#_Toc107348725)

[**2.3.** **Проектирование сценария** 15](#_Toc107348726)

[**2.4.** **Диаграмма классов** 16](#_Toc107348727)

[**2.5.** **Описание главного модуля** 18](#_Toc107348728)

[**2.6.** **Описание спецификаций к модулям** 23](#_Toc107348729)

[**2.7.** **Описание модулей** 24](#_Toc107348730)

[**2.8.** **Описание тестовых наборов модулей** 25](#_Toc107348731)

[**2.9.** **Описание применения средств отладки** 28](#_Toc107348732)

[**3.** **Эксплуатационная часть** 29](#_Toc107348733)

[**3.1.** **Руководство оператора** 29](#_Toc107348734)

[**3.2.** **To-Do лист** 38](#_Toc107348735)

[**Заключение** 39](#_Toc107348736)

[**Список использованной литературы и интернет-ресурсов** 40](#_Toc107348737)

[**Приложения** 41](#_Toc107348738)

# **Введение**

Целью данного курсового проекта является разработка приложения «Клиент спортивного магазина» для упрощения работы спортивных магазинов и удобства покупателей. Эта тема является актуальной на данный момент, потому что в наше время все стремятся все автоматизировать или упростить. Данный курсовой проект позволит облегчить работу персонала магазина и облегчить совершение покупок покупателям. Так же в данном проекте будет простой для понимания интерфейс, корзина покупок и чек покупки.

В первой части будет рассмотрена предметная область данной темы, а также несколько продуктов по данной теме.

Во второй части будут рассмотрены инструменты и модули, которые были разработаны, структура программной части и листинги ключевых частей программных модулей.

В третьей части будет рассмотрено руководство для пользователей.

В заключительной части будет приведен To-do лист с планами по доработки программы, а также сделаны общие выводы о получившемся проекте.

# **Теоретическая часть**

## **Описание предметной области**

Спортивный магазин – универсальный магазин, в котором имеется свой перечень товаров (спортивная одежда, экипировка спортсменов, спортивное питание), которые разделены на виды, эти товары поставляют ему поставщики, причем назначение каждого поставщика уникально.

Один поставщик может поставлять разные виды товаров.

В большинстве магазинов предпочитают пользоваться специальной программой. Поставленные товары заносятся на склад, должно указываться их количество, а также должна храниться информация о поставщиках, сведения о проданных товарах, также должно храниться в соответствующей таблице.

**(Она позволяет хранить информацию о совершенных покупках клиента, данные о клиенте и о ассортименте товаров в магазине.)**

Существует большое количество спортивных магазинов [4]:

1. Интернет-магазин «Спортмастер».

Спортмастер — это доступ в мир спорта с рекордным ассортиментом! Это настоящий бренд-эксперт в спорте и активном отдыхе, который вдохновляет своих покупателей, мотивирует их и помогает регулярно заниматься спортом, чтобы радоваться жизни и менять себя к лучшему день за днем!

Первый магазин сети открылся в 1992 году. С тех пор их главной и неустанной заботой является удовлетворенность потребителей качеством, разнообразием и доступностью товаров и сервисов.

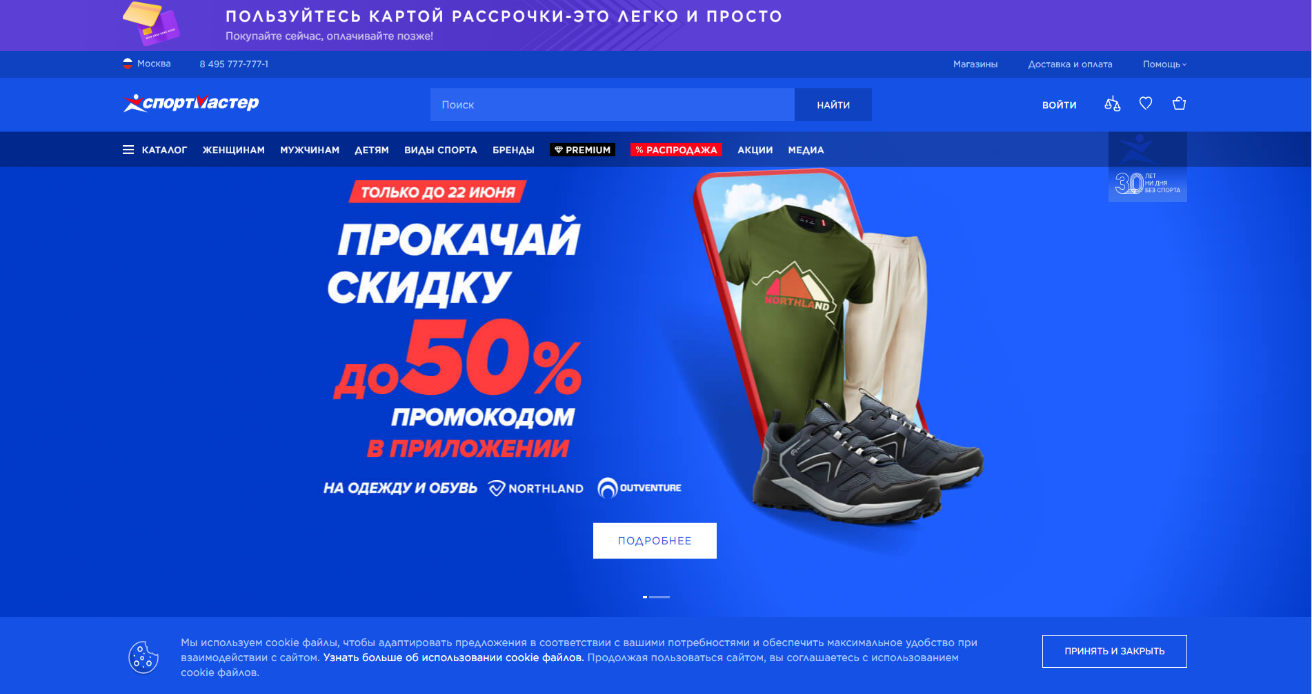


Рисунок 1. «Спортмастер»

1. Интернет-магазин «Decathlon».

Decathlon — это не только классический ритейлер, который закупает и перепродает товары. Decathlon приступил к производству и дизайну своей собственной продукции еще в 80-х годах. Их основатели хотели устанавливать цену сами, чтобы сделать спортивные товары доступными.

Безопасность является их приоритетом. Вся их продукция отвечает требованиям безопасности и прошла испытания в реальных условиях. Если их клиенты оценивают продукт менее 2 звезд из 5-ти, то они останавливают производство и снимаем товар с продаж во всех странах, даже если товар есть в большом количестве в магазинах.



Рисунок 2. «Decathlon»

1. Интернет-магазин «Дом Спорта».

Компания “Дом Спорта” основана в 2009 году. За одиннадцать лет работы более 200 тысяч клиентов приобрели их товары. Они следят за трендами в сфере спортивного оборудования и предоставляют вам ассортимент из 50 тысяч позиций. В их магазине есть не только простые вещи (например, мячи или штанги), но и оборудование для обустройства целых тренажерных залов.

Они работают с проверенными временем производителями и дистрибьюторами. Все товары в магазине сертифицированы и соответствуют санитарным нормам. А еще они предоставляют гарантию не только в собственном сервисном центре, но и в сервисных центрах производителей, поэтому вы можете не беспокоиться о дорогостоящем ремонте.

Их интернет-магазин доставляет товары по всей стране. Простая и удобная система заказа позволяет приобрести желаемый тренажер, не выходя из дома. Они заботятся о вашем комфорте, поэтому предлагают доставку до ближайшего к вам пункта самовывоза или могут привезти заказ к вам домой.

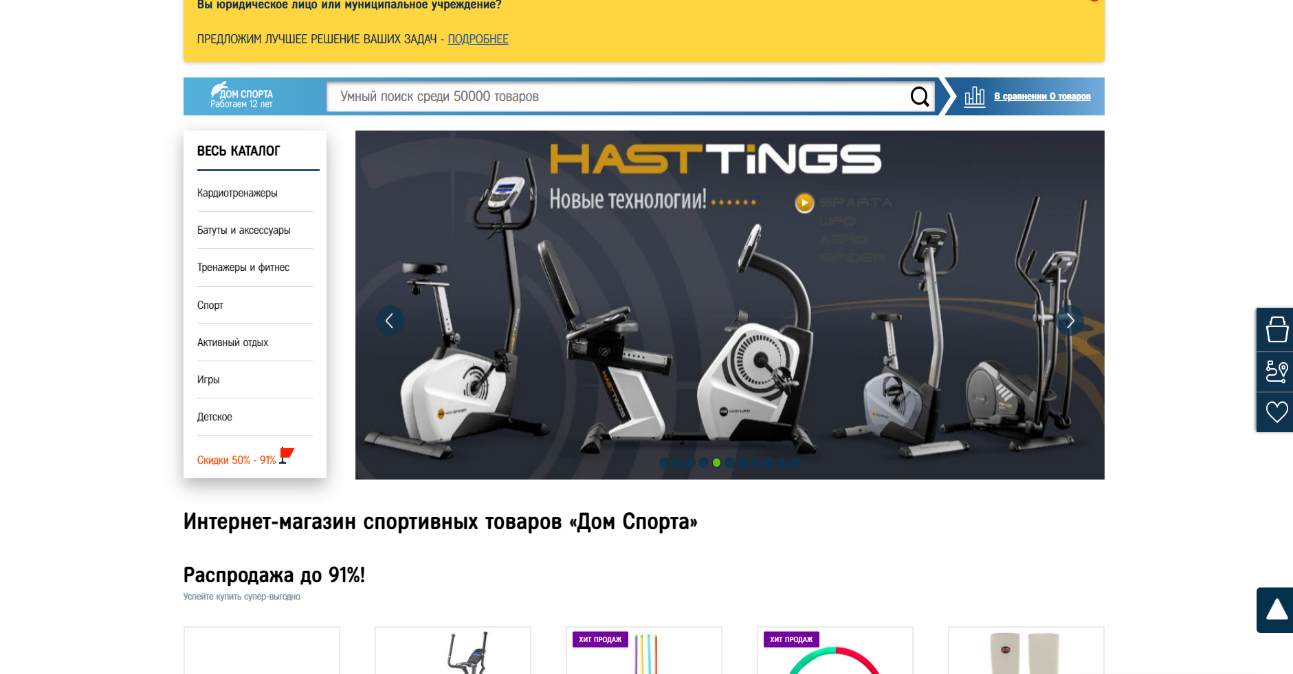


Рисунок 3. «Дом Спорта»

1. Интернет-магазин «Кант».

У КАНТа 20 спортивных магазинов, которые специализируются на 6 видах спорта: горные лыжи, сноуборд, беговые лыжи, туризм, роликовые коньки и велоспорт. Сеть спортивных специализированных магазинов КАНТ отличает от других высокий профессионализм продавцов-консультантов, прошедших специальное обучение и специально подготовленных к каждой коллекции нового сезона, возможность приобрести в одном месте полный ассортимент правильно подобранной экипировки и наличие в магазинах лучшего горнолыжного и сноубордического оборудования.

Компания КАНТ уверенно лидирует на рынке спортивных товаров в Москве и в регионах России. За много лет работы они собрали под своими знамёнами ведущих производителей спортивного инвентаря, что позволяет удовлетворить запросы любого потребителя различных видов спорта, туризма и отдыха.

КАНТ является эксклюзивным дистрибьютором следующих известных мировых брендов: Elan, Black fire, Jones, Now, FTWO, Focus, Rocky Mountain, WELT, Hamax, Alpina, Buff, Kama, Howl, Matt, Romp, Reima

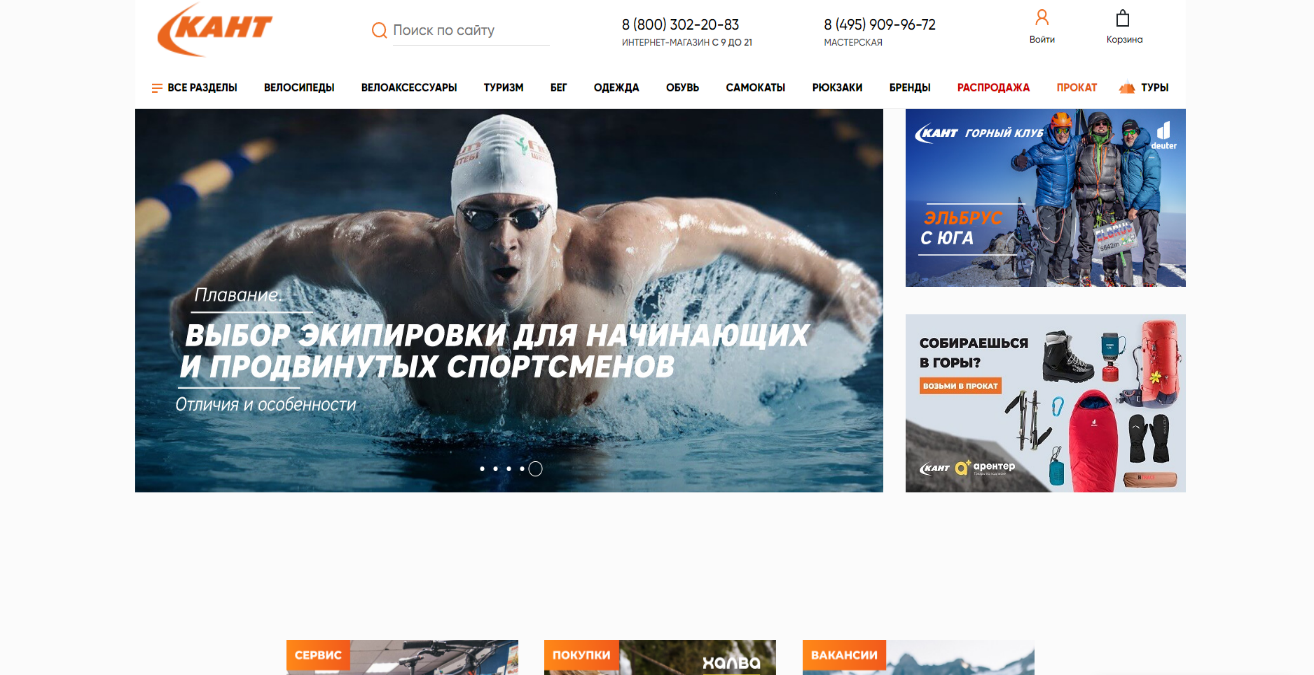


Рисунок 4. «Кант»

1. Интернет-магазин «Sport Point».

Sport Point — российская мультибрендовая сеть магазинов спортивной одежды, обуви и аксессуаров для тренировок, соревнований и повседневной жизни.

Интернет-магазин Sport Point — крупный портал, который объединяет товары более 10 спортивных брендов, а также интересные подборки статей в мире спортивного ритейла.

Первый магазин был открыт в 2016 году. С тех пор открыто более 10 магазинов в Москве и Подмосковье, Санкт-Петербурге и на территории СНГ.

Опыт, накопленный за эти годы, они трансформируем в постоянные улучшения всех аспектов онлайн и офлайн торговли, от состава брендов, до сервиса и индивидуального подхода к каждому покупателю.

Их ассортимент – это не только хорошо зарекомендовавшие себя бренды, такие как Nike, Adidas, Reebok, Puma, Asics, New Balance, но и уникальные для России бренды – Under Armour, 4F, LIVE UP.

В их магазинах Вы можете найти продукт на любой вкус, от новых коллекций и обувных франшиз, до классических силуэтов и бюджетных моделей. Они стремятся объединить модные тенденции и умеренную цену в рамках одного магазина.

Так же у них есть программа лояльности с кэшбеком.

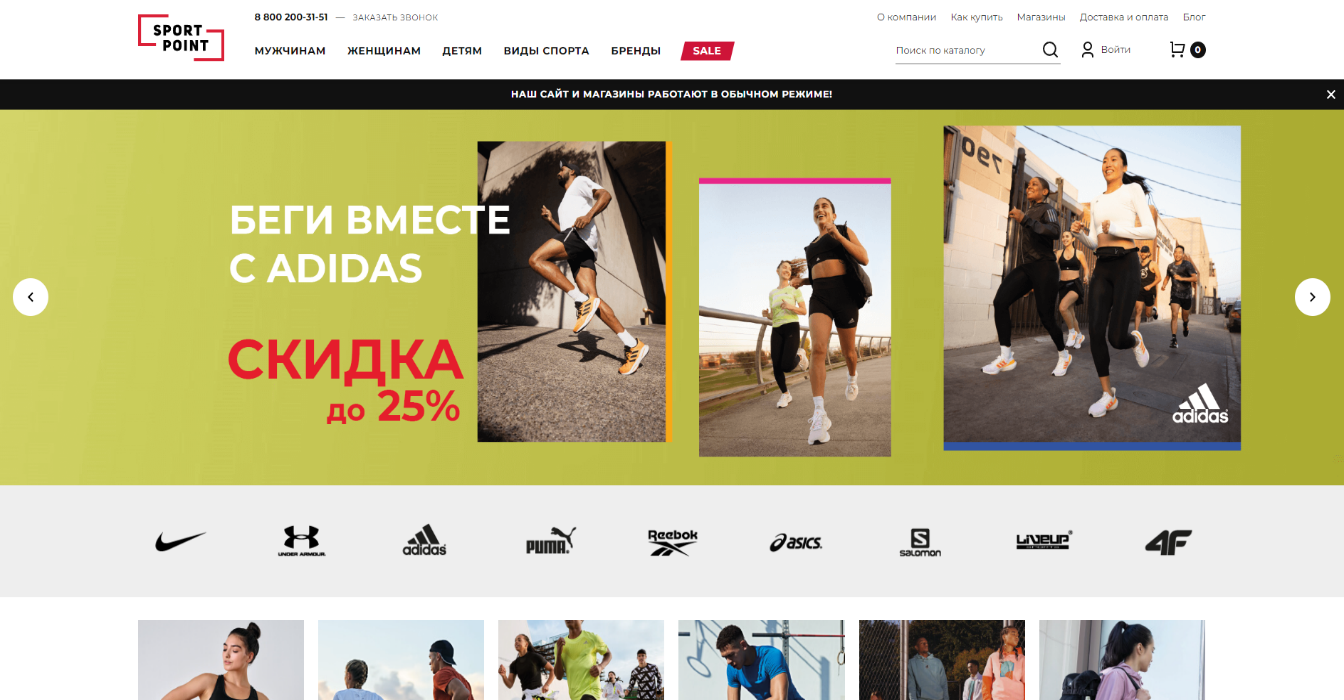


Рисунок 5. «Sport Point»

## **Описание существующих разработок**

В это разделе рассмотрены некоторые уже имеющиеся клиенты спортивных магазинов, а также цели таких программ и требования.

Интерфейс таких программ чаще всего простой и понятный для обычных пользователей.

Цели использования:

1. Поиск товара по каталогу
2. Карта магазинов
3. Программа лояльности
4. Онлайн-доставка

Основные требования:

1. Обратная связь
2. Персонализация
3. Система бонусов и наград
4. Удобный интерфейс

### **Спортмастер.**

Полное название - Спортмастер: интернет-магазин [2].

Одно из самых популярных приложений, в интернете есть множество положительных отзывов о нём.

Достоинства, которые подтверждают отзывы:

1. Быстрая работа.
2. Дополнительные бонусы и виртуальная бонусная карта.
3. Дизайн и простота.
4. Большой выбор.
5. Имеет богатый и удобный функционал.

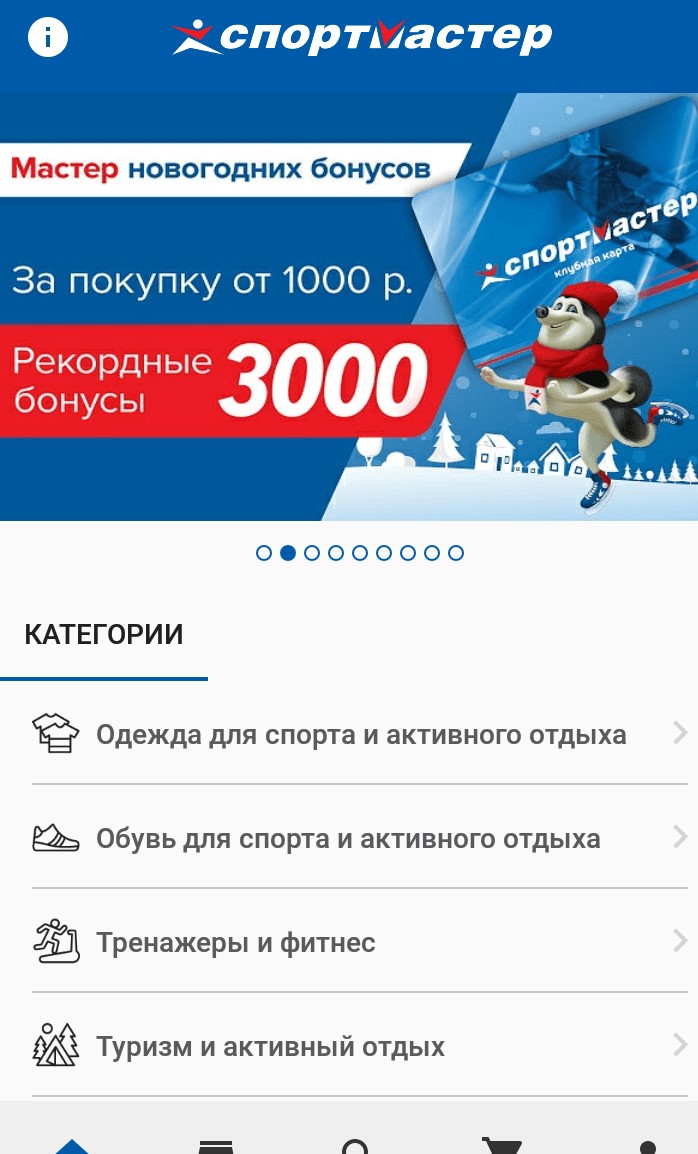


Рисунок 6. Интерфейс Спортмастера

Он включает в себя:

1. Личный кабинет.  
   Проверяйте баланс Клубной карты и отслеживайте заказы в личном кабинете.
2. Рекомендации.  
   Созданием персональных подборок занимается искусственный интеллект, поэтому в них будет только то, что вам интересно.
3. Список желаний.  
   Добавляйте понравившиеся товары в избранное, коснувшись сердечка на фото.
4. Быстрый поиск.
5. Умная система фильтров поможет найти нужный товар по заданным вами параметрам.
6. Дополнительные сервисы.  
   Бесплатные онлайн-тренировки, афиша мероприятий Спортмастер.

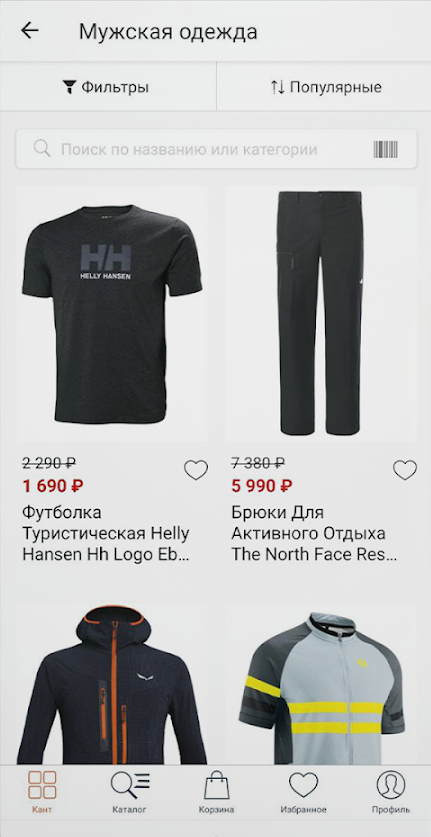
### **Кант**

Полное название - Кант-одежда и обувь для спорта [3].

Ещё одно популярное приложение, в интернете есть множество положительных отзывов о нём.

Достоинства, которые подтверждают отзывы:

1. Быстрая работа.
2. Бесплатная доставка.
3. Дизайн и простота.
4. Бесплатная доставка.
5. Широкий ассортимент.



Он включает в себя:

1. Поиск товара по штрихкоду.  
   Вы можете использовать поиск по названию или категории, а также отсканировать шрихкод, чтобы узнать характеристики продукта и прочитать описание и отзывы.
2. Распродажи.  
   У них есть раздел с акциями!
3. Избранное.  
   Добавляйте понравившиеся товары в список желаний, чтобы не потерять!
4. Быстрый поиск.
5. Умная система фильтров поможет найти нужный товар по заданным вами параметрам.
6. Бонусная программа.  
   Для покупателей разработали бонусную программу.
7. **Проектная часть**

## **Диаграмма прецедентов**

В этом разделе представлена диаграммы прецедентов. На ней показаны все возможные функциональные и поведенческие отношения.

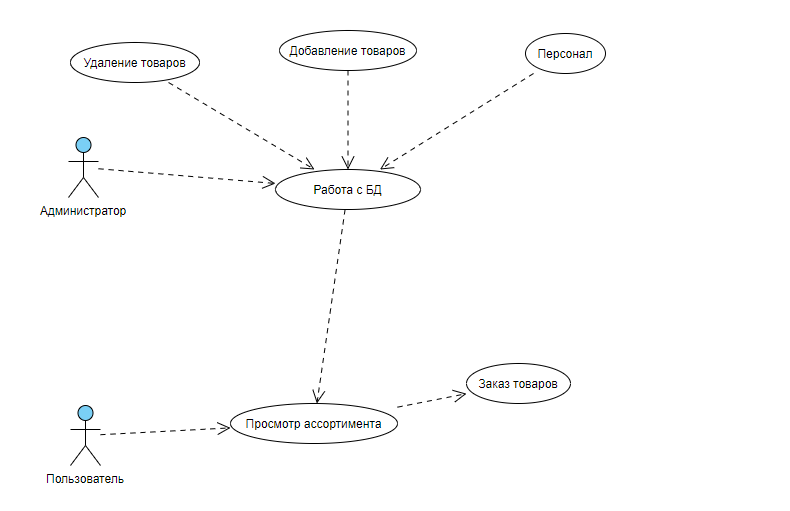


Рисунок 7. Диаграмма прецедентов

## **Выбор инструментов**

При выборе инструментов было проведено сравнение по критериям, представленных в таблице 1.

Степень важности критерия выбиралась из: низкая, ниже средней, средняя, ниже высокой, высокая.

Таблица 1. Критерии выбора инструмента.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерий | Участие в корпоративном проекте | Простота сопровождения | Наличие библиотек | Наличие документации на русском языке | Скорость разработки |
| Важность критерия | Высокая | Средняя | Высокая | Ниже средней | Ниже высокой |

Исходя из этих критериев, я сравнил 3 языка программирования от 0 до 10 баллов за критерий.

Таблица 2. Оценка языков программирования.

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий** | **C++** | **Python** | **Kotlin** |
| Участие в корпоративном проекте | 10 | 8 | 7 |
| Простота сопровождения | 7 | 10 | 5 |
| Наличие библиотек | 6 | 10 | 8 |
| Наличие документации на русском языке | 8 | 6 | 5 |
| Скорость разработки | 6 | 10 | 5 |
| Итого баллов | 37 | 44 | 30 |

По результатам сравнения был выбран язык программирования Python.

## **Проектирование сценария**

В данном разделе приведен сценарий использования программы пользователем приложения.

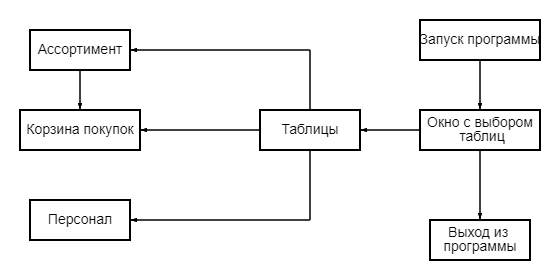


Рисунок 8. Сценарий использования

Пользователь после запуска программы может выполнить 2 действия: выйти из программы или начать работу с таблицами.

При выборе выхода программа заканчивает свою работу, при выборе работы с таблицами пользователю открываются таблицы, где может выбрать с какой таблицей работать и какие действия совершать, а из таблицы «Ассортимент» можно добавить товар в таблицу «Корзина покупок».

## **Диаграмма классов**

В данном разделе представлены все классы, использующиеся в проекте, а также их отношения между собой.

Класс «DB1» содержит следующие методы:

* insertDataAssortment – Добавление данных для «Assortment»
* insertDataWorkers – Добавление данных для «Workers»
* records1 – Ввод новых данных «Assortment»
* updateRecord1 – Редактирование данных «Assortment»
* deleteRecords1 – Удаление результата «Assortment»
* addcart – Добавление товара в корзину
* records2 – Ввод новых данных «Shopping cart»
* updateRecord2 – Редактирование данных «Shopping cart»
* deleteRecords2 – Удаление результата «Shopping cart»
* records3 – Ввод новых данных «Workers»
* updateRecord3 – Редактирование данных «Workers»
* deleteRecords3 – Удаление результата «Workers»

Класс «Assortment» содержит следующие методы:

* viewRecords – Вывод данных
* tableAssortment – Создание таблицу с помощью ttk.Treeview()
* editAssortment – Редактирование таблицы
* idgive – Получение строки по id из БД
* idDelete – Удаление строки по id из БД
* lineDelete – Удаление выбранной строки из таблицы tkinter

Класс «Shopping\_cart» содержит следующие методы:

* viewRecords – Вывод данных
* tableShopping\_cart– Создание таблицу с помощью ttk.Treeview()
* editShopping\_cart– Редактирование таблицы
* idDelete – Удаление строки по id из БД
* lineDelete – Удаление выбранной строки из таблицы tkinter

Класс «Workers» содержит следующие методы:

* viewRecords – Вывод данных
* tableWorkers – Создание таблицу с помощью ttk.Treeview()
* editWorkers– Редактирование таблицы
* idDelete – Удаление строки по id из БД
* lineDelete – Удаление выбранной строки из таблицы tkinter

Класс «tkAssortment» содержит следующие методы:

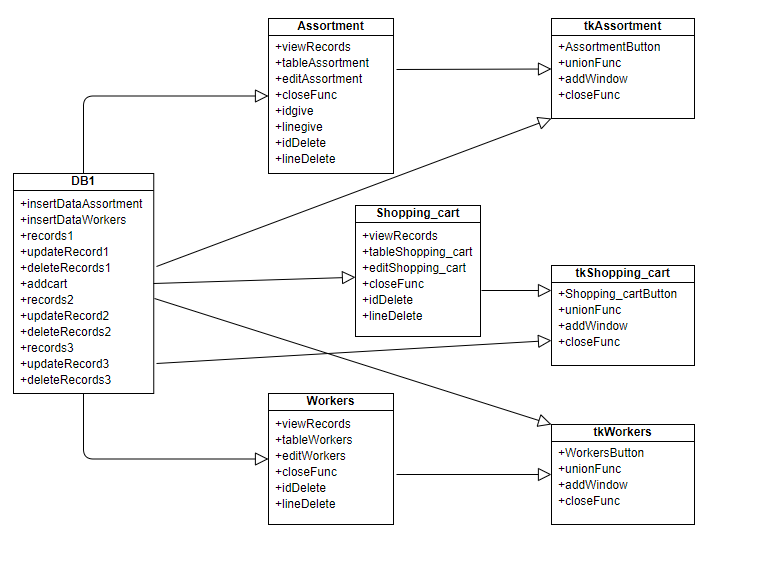
* AssortmentButton – Кнопка для открытия таблицы и её функций
* unionFunc – Компонует все кнопки функции выше
* addWindow – Создание нового окна для кнопки <Добавить>

Класс «tkShopping\_cart» содержит следующие методы:

* Shopping\_cartButton – Кнопка для открытия таблицы и её функций
* unionFunc – Компонует все кнопки функции выше
* addWindow – Создание нового окна для кнопки <Добавить>

Класс «tkWorkers» содержит следующие методы:

* WorkersButton – Кнопка для открытия таблицы и её функций
* unionFunc – Компонует все кнопки функции выше
* addWindow – Создание нового окна для кнопки <Добавить>



## **Описание главного модуля**

В главный модуль входят классы «Assortment», «Shopping\_cart», «Workers», «tkAssortment», «tkShopping\_cart», «tkWorkers», которые отвечают за функционал программы.

Также в главный модуль входит код, который запускает саму программу при ее запуске. Полный код главного модуля находится в «Приложение 1».

**Листинг 1. Класс «tkAssortment» отвечающий за интерфейс таблицы «Assortment»**

class tkAssortment():

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

self.sale = sale

self.db = db

def AssortmentButton(self):

def unionFunc():

def addWindow():

def closeFunc():

s.destroy()

s = Toplevel()

s.geometry('250x200')

s.title('Add')

s.resizable(False, False)

Name\_label = Label(s, text='Наименование').place(x=10, y=10, width=85, height=30)

Price\_label = Label(s, text='Цена').place(x=10, y=60, width=80, height=30)

Quantity\_label = Label(s, text='Количество').place(x=10, y=100, width=80, height=30)

name\_1 = StringVar()

name\_2 = StringVar()

name\_3 = StringVar()

Name\_entry = Entry(s, width=50, textvariable=name\_1).place(x=130, y=10, width=100, height=30)

Price\_entry = Entry(s, width=50, textvariable=name\_2).place(x=130, y=60, width=100, height=30)

Quantity\_entry = Entry(s, width=50, textvariable=name\_3).place(x=130, y=100, width=100, height=30)

add\_button = Button(s, text='Добавить', command=lambda:

(self.db.records1(name\_1.get(), name\_2.get(), name\_3.get()), self.sale.viewRecords(), self.sale.tableAssortment()))

add\_button.place(x=10, y=160, width=115, height=30)

close\_button = Button(s, text='Закрыть', command=closeFunc)

close\_button.place(x=130, y=160, width=80, height=30)

self.add\_image = tk.PhotoImage(file='pictures/add.png')

Assortment\_add\_button = Button(root, text='Добавить', image=self.add\_image, compound='top', command=addWindow)

Assortment\_add\_button.place(x=10, y=50, width=100, height=80)

self.edit\_image = tk.PhotoImage(file='pictures/edit.png')

Assortment\_edit\_button = Button(root, text='Редактировать', image=self.edit\_image, compound='top', command=lambda:

self.sale.editAssortment())

Assortment\_edit\_button.place(x=120, y=50, width=100, height=80)

self.delete\_image = tk.PhotoImage(file='pictures/delete.png')

Assortment\_edit\_button = Button(root, text='Удалить', image=self.delete\_image, compound='top', command=lambda:

(self.sale.idDelete(), self.sale.lineDelete()))

Assortment\_edit\_button.place(x=230, y=50, width=100, height=80)

self.give\_image = tk.PhotoImage(file='pictures/give.png')

Assortment\_addcarts\_button = Button(root, text='Добавление в корзину', image=self.give\_image, compound='top', command=lambda:

(self.sale.idgive(), self.sale.linegive()))

Assortment\_addcarts\_button.place(x=400, y=50, width=140, height=80)

Assortment\_button = Button(root, text='Assortment', command=lambda:

(unionFunc(), self.sale.viewRecords(), self.sale.tableAssortment()))

Assortment\_button.place(x=10, y=10, width=100, height=30)

**Листинг 2. Класс «tkShopping\_cart» отвечающий за интерфейс таблицы «Shopping\_cart»**

class tkShopping\_cart():

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

self.cust = cust

self.db = db

def Shopping\_cartButton(self):

def unionFunc():

def addWindow():

def closeFunc():

s.destroy()

s = Toplevel()

s.geometry('370x150')

s.title('Add')

s.resizable(False, False)

Quantity\_label = Label(s, text='Добавлять в корзину можно только в таблице ассортимента').place(x=10, y=10, width=350, height=30)

close\_button = Button(s, text='Закрыть', command=closeFunc)

close\_button.place(x=60, y=100, width=250, height=30)

self.add\_image = tk.PhotoImage(file='pictures/add.png')

Shopping\_cart\_add\_button = Button(root, text='Добавить', image=self.add\_image, compound='top', command=addWindow)

Shopping\_cart\_add\_button.place(x=10, y=50, width=100, height=80)

self.edit\_image = tk.PhotoImage(file='pictures/edit.png')

Shopping\_cart\_edit\_button = Button(root, text='Редактировать', image=self.edit\_image, compound='top', command=lambda:

self.cust.editShopping\_cart())

Shopping\_cart\_edit\_button.place(x=120, y=50, width=100, height=80)

self.delete\_image = tk.PhotoImage(file='pictures/delete.png')

Shopping\_cart\_edit\_button = Button(root, text='Удалить', image=self.delete\_image, compound='top', command=lambda:

(self.cust.idDelete(), self.cust.lineDelete()))

Shopping\_cart\_edit\_button.place(x=230, y=50, width=100, height=80)

self.give\_image = tk.PhotoImage(file='pictures/give.png')

Shopping\_cart\_addcarts\_button = Button(root, text='В этой таблице нельзя!', image=self.give\_image, compound='top', command=lambda:

())

Shopping\_cart\_addcarts\_button.place(x=400, y=50, width=140, height=80)

Shopping\_cart\_button = Button(root, text='Shopping\_cart', command=lambda:

(unionFunc(), self.cust.viewRecords(), self.cust.tableShopping\_cart()))

Shopping\_cart\_button.place(x=120, y=10, width=100, height=30)

**Листинг 3. Класс «tkWorkers» отвечающий за интерфейс таблицы «Workers»**

class tkWorkers():

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

self.orde = orde

self.db = db

def WorkersButton(self):

def unionFunc():

def addWindow():

def closeFunc():

s.destroy()

s = Toplevel()

s.geometry('280x270')

s.title('Add')

s.resizable(False, False)

Surname\_label = Label(s, text='Фамилия').place(x=20, y=10, width=70, height=30)

Name\_label = Label(s, text='Имя').place(x=20, y=60, width=70, height=30)

Post\_label = Label(s, text='Должность').place(x=20, y=100, width=70, height=30)

name\_1 = StringVar()

name\_2 = StringVar()

name\_3 = StringVar()

Surname\_entry = Entry(s, width=50, textvariable=name\_1).place(x=130, y=10, width=120, height=30)

Name\_entry = Entry(s, width=50, textvariable=name\_2).place(x=130, y=60, width=120, height=30)

Post\_entry = Entry(s, width=50, textvariable=name\_3).place(x=130, y=100, width=120, height=30)

add\_button = Button(s, text='Добавить', command=lambda:

(self.db.records3(name\_1.get(), name\_2.get(), name\_3.get()), self.orde.viewRecords(), self.orde.tableWorkers()))

add\_button.place(x=10, y=230, width=100, height=30)

close\_button = Button(s, text='Закрыть', command=closeFunc)

close\_button.place(x=115, y=230, width=100, height=30)

self.add\_image = tk.PhotoImage(file='pictures/add.png')

Spec\_Sections\_add\_button = Button(root, text='Добавить', image=self.add\_image, compound='top', command=addWindow)

Spec\_Sections\_add\_button.place(x=10, y=50, width=100, height=80)

self.edit\_image = tk.PhotoImage(file='pictures/edit.png')

Spec\_Sections\_edit\_button = Button(root, text='Редактировать', image=self.edit\_image, compound='top', command=lambda:

self.orde.editWorkers())

Spec\_Sections\_edit\_button.place(x=120, y=50, width=100, height=80)

self.delete\_image = tk.PhotoImage(file='pictures/delete.png')

Spec\_Sections\_edit\_button = Button(root, text='Удалить', image=self.delete\_image, compound='top', command=lambda:

(self.orde.idDelete(), self.orde.lineDelete()))

Spec\_Sections\_edit\_button.place(x=230, y=50, width=100, height=80)

self.give\_image = tk.PhotoImage(file='pictures/give.png')

Shopping\_cart\_addcarts\_button = Button(root, text='В этой таблице нельзя!', image=self.give\_image, compound='top', command=lambda:

())

Shopping\_cart\_addcarts\_button.place(x=400, y=50, width=140, height=80)

Workers\_button = Button(root, text='Workers', command=lambda:

(unionFunc(), self.orde.viewRecords(), self.orde.tableWorkers()))

Workers\_button.place(x=230, y=10, width=100, height=30)

## **Описание спецификаций к модулям**

В данном разделе описаны публичные члены модулей курсового проекта.

В главном модуле «main.py» содержатся публичные методы классов «Assortment», «Shopping\_cart», «Workers», «tkAssortment», «tkShopping\_cart», «tkWorkers», которые взаимодействуют с методами класса «DB1» из модуля «db.py».

В данном модуле также есть пара объектов:

1. root - объект, который создает системное окно и объект самого приложения.
2. db – объект, которые принимает экземпляр класса «DB1» для вызова функций.
3. sale – объект, которые принимает экземпляр класса «Assortment» для вызова функций.
4. cust – объект, которые принимает экземпляр класса «Shopping\_cart» для вызова функций.
5. orde – объект, которые принимает экземпляр класса «Workers» для вызова функций.

## **Описание модулей**

Кроме главного модуля программа содержит еще один модуль который реализует работу с базой данных «db.py».

Полный код модуля приведен в «Приложение 2». В этом разделе приведены все методы, которые содержаться в класс «DB1» из модуля «db.py».

**Листинг 4. Методы класса «DB1»**

#**Функции DB (Ассортимент)**

def records1(self, Name, Price, Quantity):

''' Ввод новых данных '''

def updateRecord1(self, Name, Price, Quantity, ID\_1):

''' Редактирование данных '''

def deleteRecords1(self, ID\_1):

''' Удаление результата '''

def addcart(self, ID\_1):

''' Добавление товара в корзину '''

**#Функции DB (Корзина покупок)**

def records2(self, Name, Price, Quantity):

''' Ввод новых данных '''

def updateRecord2(self, Quantity, ID\_2):

''' Редактирование данных '''

def deleteRecords2(self, ID\_2):

''' Удаление результата '''

**#Функции DB (Работники)**

def records3(self, Surname, Name, Post):

''' Ввод новых данных '''

def updateRecord3(self, Surname, Name, Post, ID\_3):

''' Редактирование данных '''

def deleteRecords3(self, ID\_3):

''' Удаление результата '''

## **Описание тестовых наборов модулей**

В этом разделе будут продемонстрированы результаты тестирования программы.

Тест 1. Смена таблицы.

Действия: Нажать на кнопку с названием таблицы.

Ожидаемый результат: Смена таблицы «Assortment» на таблицу «Shopping\_cart».

Результат теста:

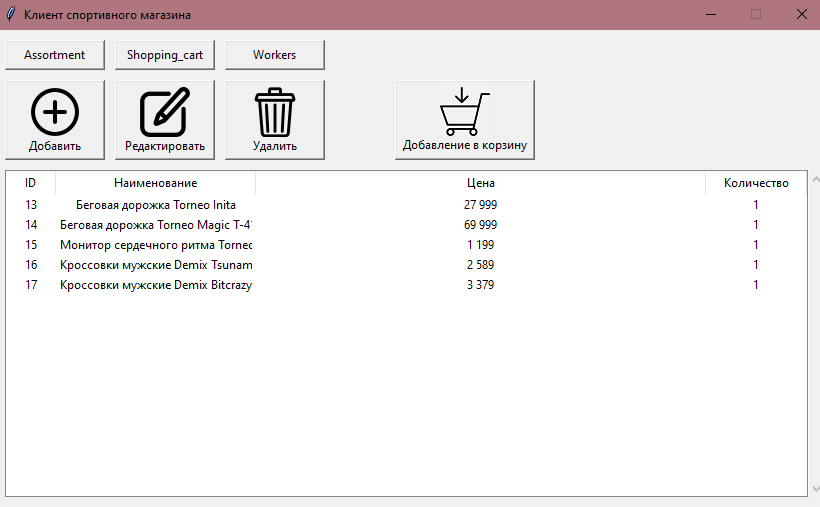


Рисунок 10. Первая таблица

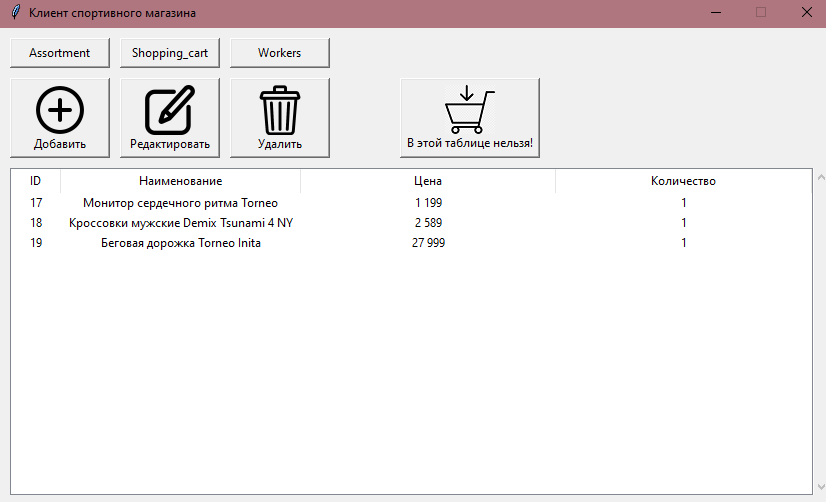


Рисунок 11. Новая таблица

Тест 2. Добавление позиции в таблицу.

Действия: Нажать на кнопку «Добавить», в появившейся новой строке таблицы ввести необходимую информацию и нажать на кнопку «Добавить».

Ожидаемы результат: Добавление строки в таблицу, добавление в нее информации и сохранение редактированной таблицы.

Результат тесте:

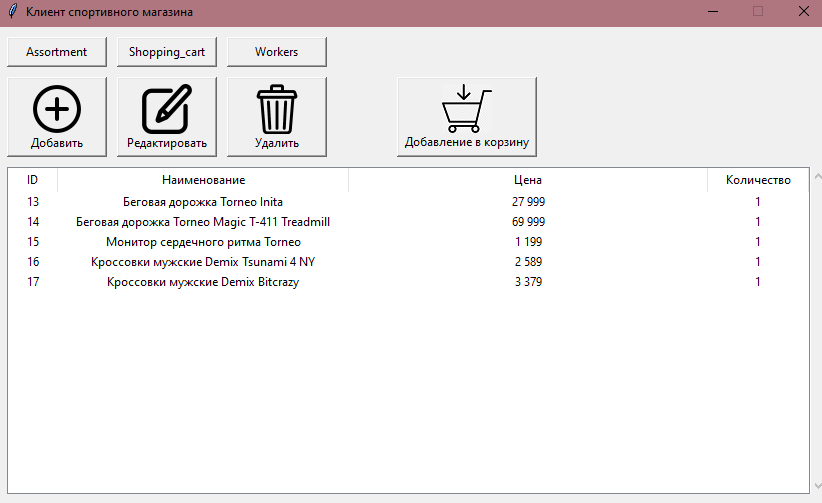


Рисунок 12. Таблица ассортимента

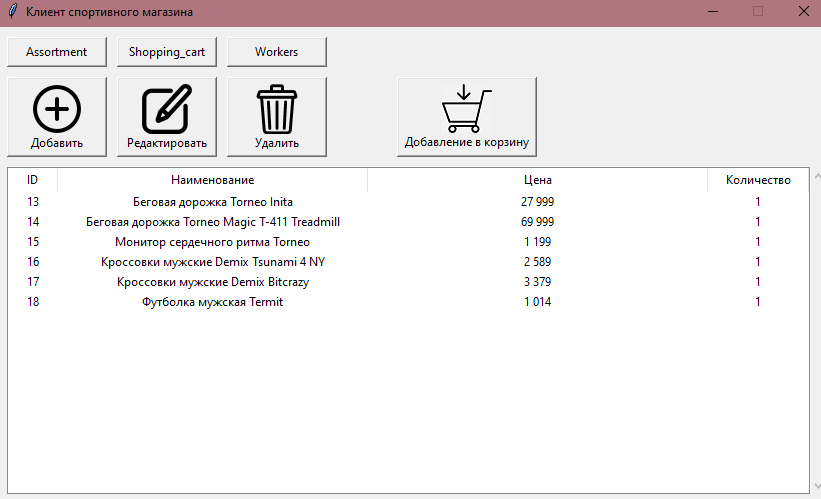


Рисунок 13. Таблица ассортимента после добавления позиции

Тест 3. Удаление позиции из таблицы.

Действия: Нажать на строку в таблице, после выделения строки нажать кнопку «Удалить».

Ожидаемый результат: Удаление выбранной строки в таблице.

Результат теста:

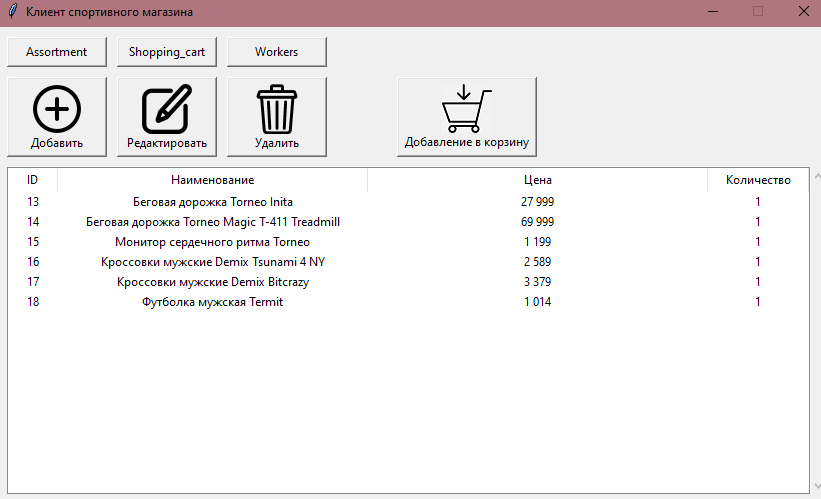


Рисунок 14. Таблица ассортимента до удаления позиции

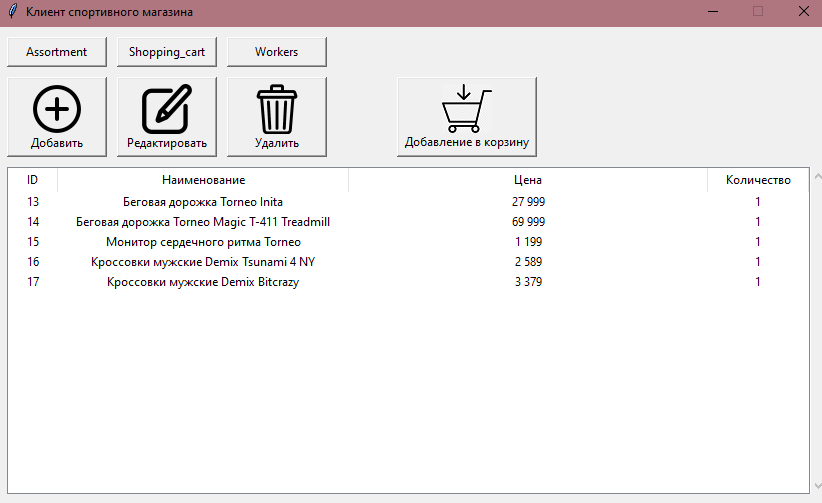
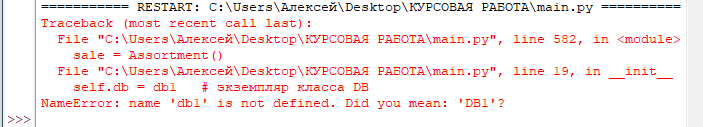


Рисунок 15. Таблица ассортимента после удаления позиции

## **Описание применения средств отладки**

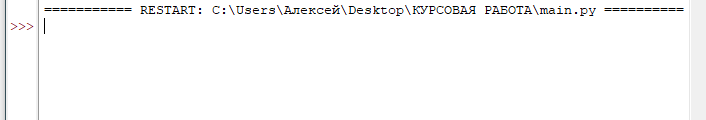
В этом разделе показано умение применять средства отладки. Так как я работаю в IDLE[1], отладчик уже встроен в программу.

Для проверки была искусственно создана ошибка в коде. Вот что показал отладчик:

****

**Рисунок 16. Ошибка**

После получения данных из отладчика, была просмотрена 19 строка модуля «main.py» и выявлена ошибка в работе класса «Assortment», объявление экземпляра класса.



**Рисунок 17. Исправлено**

# **Эксплуатационная часть**

## **Руководство оператора**

**АННОТАЦИЯ**

В данном программном документе приведено руководство оператора по применению и эксплуатации программы «Клиент спортивного магазина».

В данном программном документе, в разделе «Назначение программы» указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

В разделе «Условия выполнения программы» указаны условия, необходимые для выполнения программы (минимальный состав аппаратных и программных средств и т.п.).

В данном программном документе, в разделе «Выполнение программы» указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды.

Оформление программного документа «Руководство оператора» произведено по требованиям ЕСПД (ГОСТ 19.101-77 [[1]](#footnote-1)1), ГОСТ 19.103-77 [[2]](#footnote-2)2), ГОСТ 19.104-78\* [[3]](#footnote-3)3), ГОСТ 19.105-78\* [[4]](#footnote-4)4), ГОСТ 19.106-78\* [[5]](#footnote-5)5), ГОСТ 19.505-79\* [[6]](#footnote-6)6), ГОСТ 19.604-78\* [[7]](#footnote-7)7)).

1. **Назначение программы**
   1. **Функциональное назначение программы**

Специальное программное обеспечение «Клиент спортивного магазина» используется для добавления товаров в ассортимент приложения, добавления товаров в корзину покупок и хранения информации о персонале.

* 1. **Эксплуатационное назначение программы**

Специальное программное обеспечение «Клиент спортивного магазина» может эксплуатироваться на объектах любого масштаба в сфере спортивных магазинов для облегчения работы персонала.

* 1. **Состав функций**
     1. **Функция смены таблицы**

Эта функция позволяет менять таблицу в зависимости от необходимости.

* + 1. **Функция добавления элемента в таблицу.**

Эта функция позволяет добавлять нужную информацию в таблицу.

* + 1. **Функция удаления элемента из таблицы.**

Эта функция позволяет удалить ненужные элементы в таблице.

* + 1. **Функция добавления элемента в Корзину покупок.**

Эта функция позволяет добавлять выбранный элемент в таблицу «Корзина покупок».

1. **Условия выполнения программы**
   1. **Минимальный состав аппаратных средств**

ОС: Windows 10

Процессор: Как минимум 1 ГГц или SoC.

ОЗУ: 1 ГБ (для 32-разрядных систем) или 2 ГБ (для 64-разрядных систем).

Место на жестком диске: 16 ГБ (для 32-разрядных систем) или 20 ГБ (для 64-разрядных систем).

Видеоадаптер: DirectX версии не ниже 9 с драйвером WDDM 1.0.

Дисплей: 800 x 600.

* 1. **Минимальный состав программных средств**

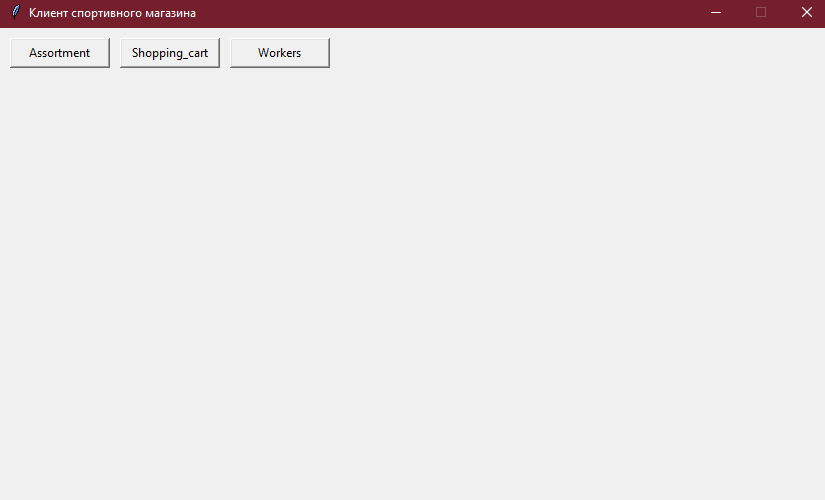
Дополнительные программные средства не требуются.

* 1. **Требование к пользователю**

Конечный пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

1. **Выполнение программы**
   1. **Загрузка и запуск программы**

Запустите программу «Клиент спортивного магазина», после чего откроется окно:

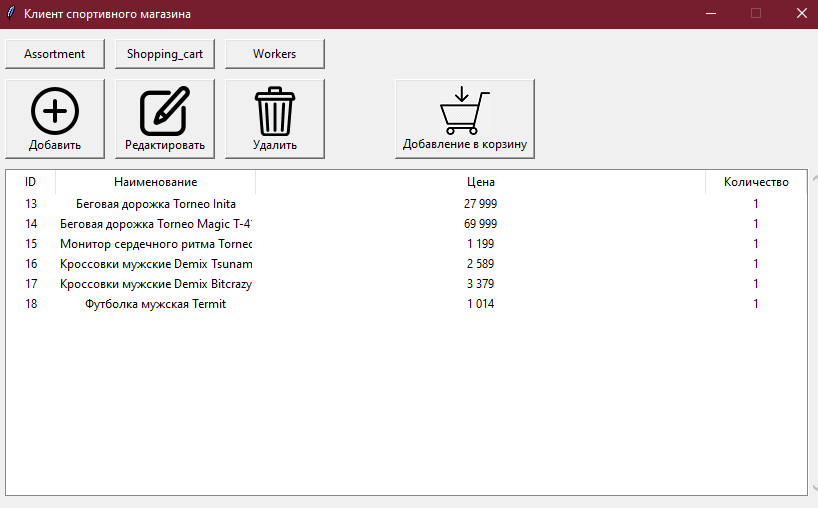
****

**Рисунок 18. Окно при запуске**

Кнопки меню имеют следующие функции:

* Assortment – Открытие таблицы с ассортиментом
* Shopping-cart – Открытие таблицы с корзиной покупкой
* Workers – Открытие таблицы с работниками  
  1. **Выполнение программы**
     1. **Выполнение функции открытия таблицы.**

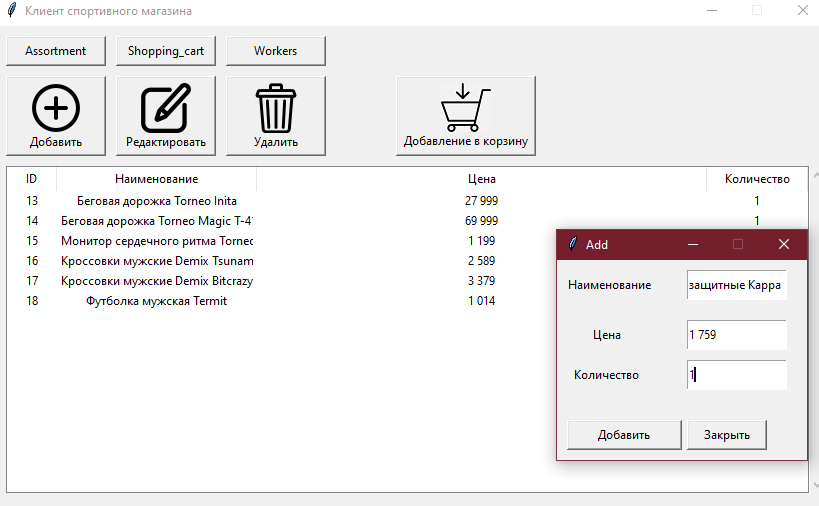
Выберите кнопку «Assortment», после этого откроется окно с таблицей.



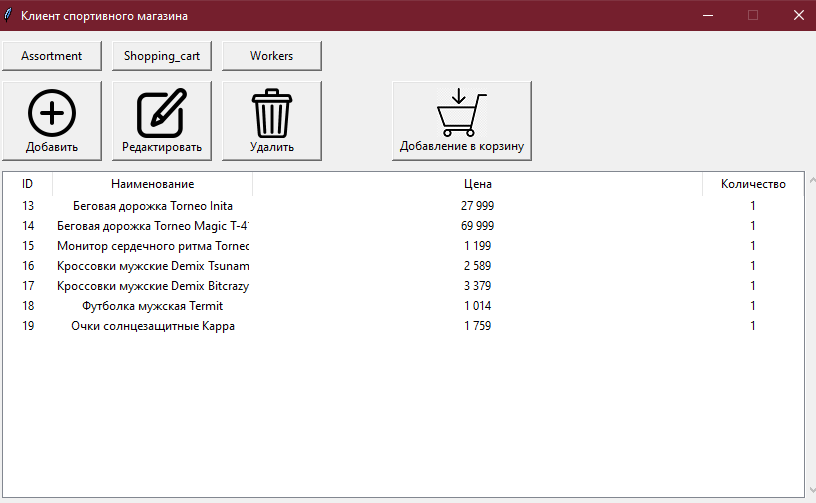
**Рисунок 19. Окно таблицы «Assortment»**

* + 1. **Выполнение функции добавления элемента в таблицу.**

Нажмите на кнопку «Добавить», появится новое поле в выбранной таблице, введите всю необходимую информацию в таблицу и нажмите на кнопку «Сохранить», если этого не сделать, то все что вы ввели не будет сохранено.



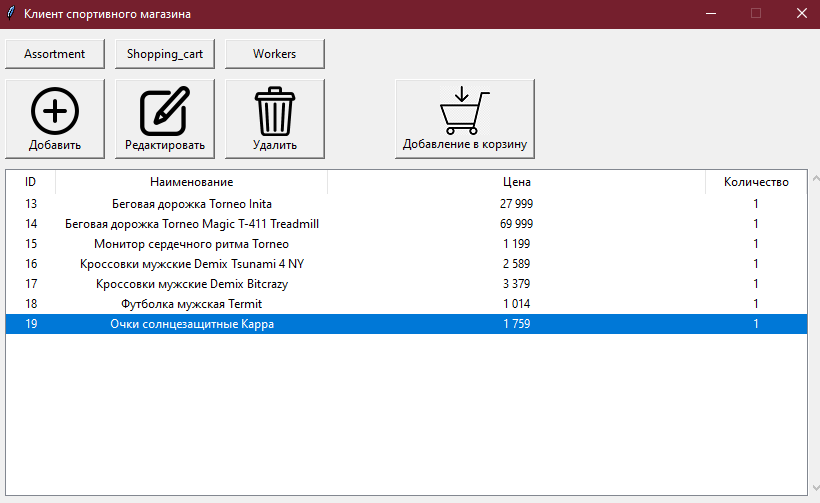
**Рисунок 20. Таблица до нажатия «Сохранить»**



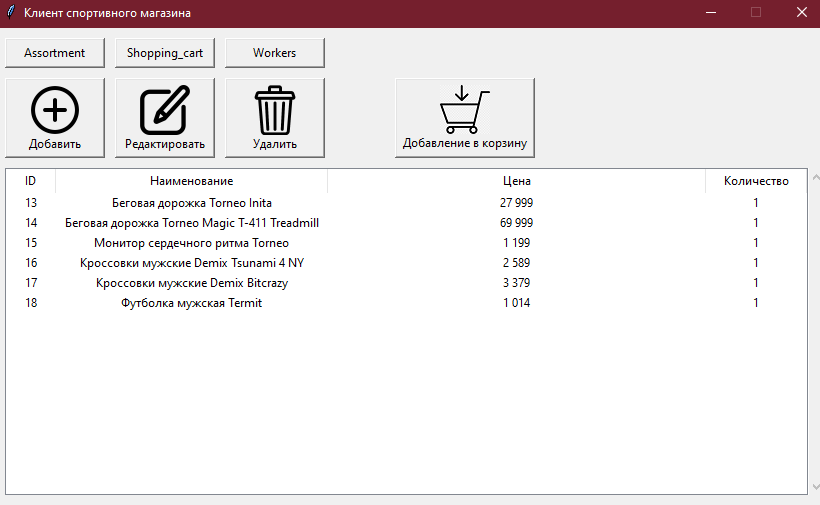
**Рисунок 21. Таблица после нажатия кнопки «Сохранить»**

* + 1. **Выполнение функции удаления элемента из таблицы.**

Нажмите на строку, которую вы хотите удалить и нажмите на кнопку «Удалить». Будьте внимательны, удаленные данные нельзя восстановить!



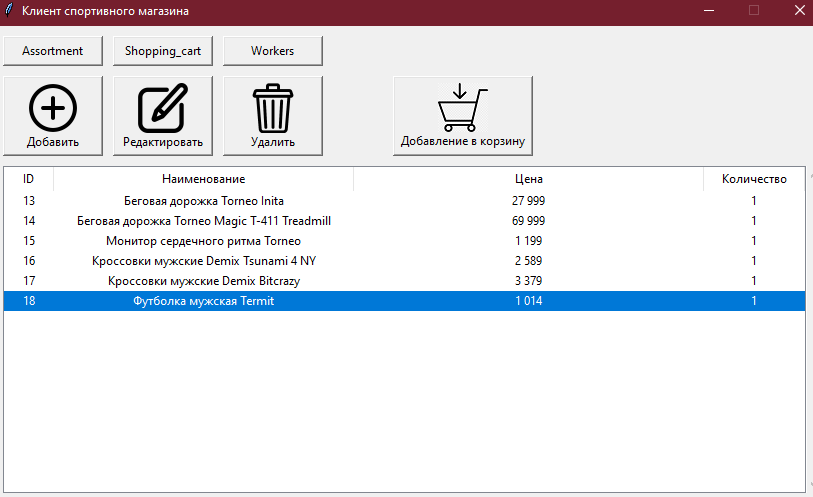
**Рисунок 22. Элемент до удаления**

****

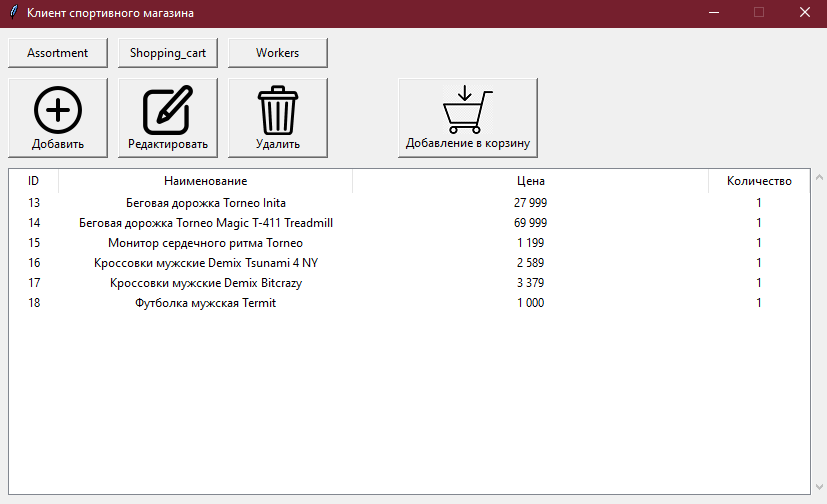
**Рисунок 23. Таблица после удаления элемента**

* + 1. **Выполнение функции редактирования элемента из таблицы.**

Нажмите на строку, которую вы хотите редактировать и нажмите на кнопку «Редактировать».



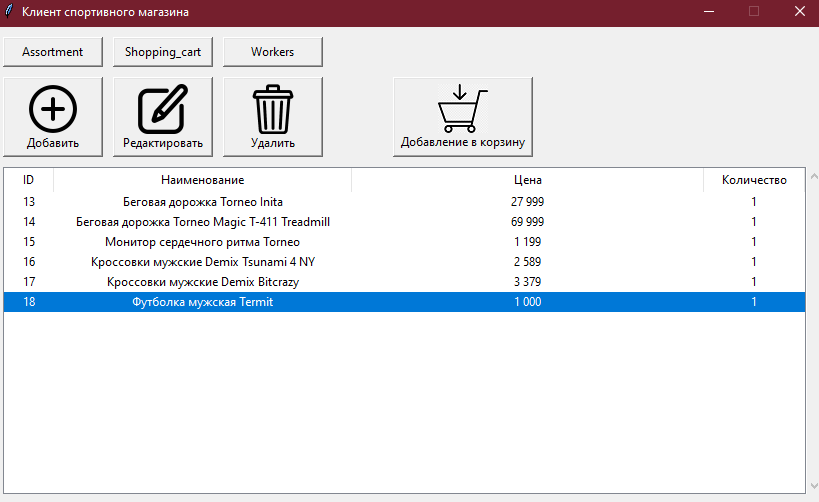
**Рисунок 24. Элемент до редактирования**



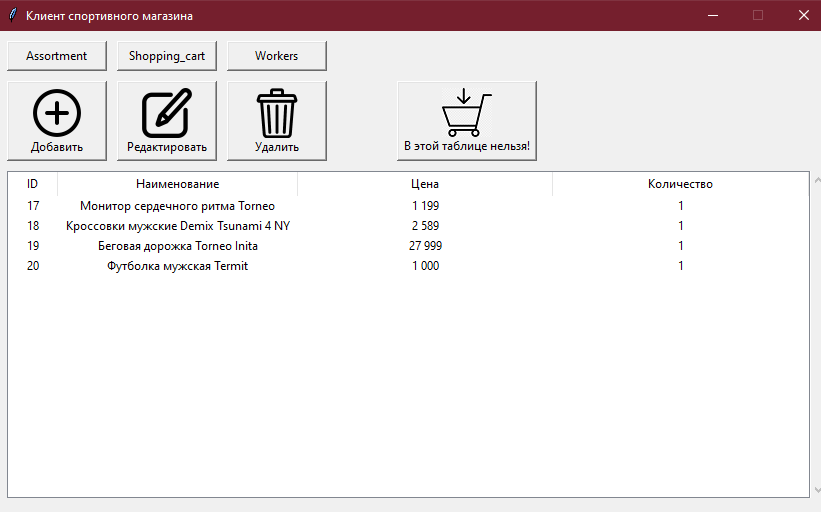
**Рисунок 25. Элемент после редактирования**

* + 1. **Выполнение функции добавление элемента из таблицы в таблицу «Корзина покупок».**

Нажмите на строку, которую вы хотите редактировать и нажмите на кнопку «Добавить в корзину».



**Рисунок 26. Выбор товара из таблицы**

****

**Рисунок 27. Обновленная таблица «Корзина покупок»**

* 1. **Завершение работы программы**

Чтобы завершить работу программы нажмите на кнопку «Закрыть» в верхнем правом углу окна.

## **To-Do лист**

1. Переписать интерфейс на PyQt.
2. Добавить вывод чека (Завершение покупки)
3. Сделать авторизацию пользователя
4. Добавить сортировку товара по виду предназначения

# **Заключение**

В результате выполнения курсового проекта была написана программа «Клиент спортивного магазина» для упрощения и удобства просмотра и покупки.

В ходе работы были проанализированы: предметная область, существующие разработки, получены навыки по созданию интерфейса с помощью библиотеки tkinter и базами данных с помощью библиотеки sqlite.

Также планируется продолжение работы над данным проектом с целью расширения возможностей и удобства приложения для пользователей. Планы по доработкам представлены в «3.2. To-Do лист».

# **Список использованной литературы и интернет-ресурсов**

1. Среда разработки – IDLE:

<https://docs.python.org/3/library/idle.html>

1. Клиент спортивного магазина – Спортмастер:

<https://apps.apple.com/ru/app/%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BC%D0%B0%D1%81%D1%82%D0%B5%D1%80-%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD/id1081336048>

1. Клиент спортивного магазина – Кант:

<https://apps.apple.com/ru/app/%D0%BA%D0%B0%D0%BD%D1%82-%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD/id1536569064>

1. Спортивные магазины:

<https://infoselection.ru/infokatalog/pokupki-shopping/pokupki-po-temam/item/902-luchshie-sportivnye-magaziny-spiski-i-obzory>

1. Мой проект на GitHub:

<https://github.com/Mist03/Coursework>

# **Приложения**

**Приложение 1. «main.py»**

import tkinter as tk

from tkinter import \*

from tkinter import ttk

import sqlite3

from db import \*

class Assortment():

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

self.db = db # экземпляр класса DB

self.variable = []

self.selected\_item = 0

self.viewRecords()

#self.funcs()

def viewRecords(self):

''' Вывод данных '''

self.db.cur.execute(

'''SELECT \* FROM Assortment''')

self.variable.clear() # очищаем прошлые данные (чтобы не дублировались)

[self.variable.append(row) for row in self.db.cur.fetchall()] # записываем новый результат

def tableAssortment(self):

''' Создание таблицу с помощью ttk.Treeview() '''

# Задаем расположение таблицы

frame = tk.Frame(root, width=100, height=100)

frame.place(x=10, y=140)

# Создаем заголовоки для таблицы

headers = ['ID', 'Наименование', 'Цена', 'Количество']

self.table = ttk.Treeview(frame, columns=headers, height=15, show='headings')

self.table.column('ID', width=50, anchor='center')

self.table.column('Наименование', width=200, anchor='center')

self.table.column('Цена', width=450, anchor='center')

self.table.column('Количество', width=100, anchor='center')

for header in headers: # заполняем заголовоки

self.table.heading(header, text=header)

for i in self.variable: # заполняем значения

self.table.insert('', tk.END, values=i)

# Создаем скролл для таблицы

scroll = ttk.Scrollbar(frame, command=self.table.yview)

self.table.configure(yscrollcommand=scroll.set)

scroll.pack(side=tk.RIGHT, fill=tk.Y)

self.table.pack(expand=tk.YES, fill=tk.BOTH)

def editAssortment(self):

''' Редактирование таблицы '''

def closeFunc():

z.destroy() # закрываем окно

# Создаем новое окно

z = Toplevel()

z.geometry('225x200')

z.title('Edit')

z.resizable(False, False)

# Получаем данные строки по клику

selected\_item = self.table.selection()[0]

values = self.table.item(selected\_item, option='values')

# Заносим данные строки в переменные

id = values[0]

Name = values[1]

Price = values[2]

Quantity = values[3]

# Создаем метки и их расположение

Name\_label = Label(z, text='Наименование').place(x=10, y=10, width=90, height=30)

Price\_label = Label(z, text='Цена').place(x=10, y=60, width=70, height=30)

Quantity\_label = Label(z, text='Количество').place(x=10, y=100, width=70, height=30)

# Переменные для ввода значений

Name\_1 = StringVar()

Price\_2 = StringVar()

Quantity\_3 = StringVar()

# Строки для ввода значений

Name\_entry = Entry(z, width=50, textvariable=Name\_1)

Price\_entry = Entry(z, width=50, textvariable=Price\_2)

Quantity\_entry = Entry(z, width=50, textvariable=Quantity\_3)

# Записываем значения из строк ввода

Name\_entry.insert(0, str(Name))

Price\_entry.insert(0, str(Price))

Quantity\_entry.insert(0, str(Quantity))

# Задаем расположение строк ввода

Name\_entry.place(x=100, y=10, width=110, height=30)

Price\_entry.place(x=100, y=60, width=110, height=30)

Quantity\_entry.place(x=100, y=100, width=110, height=30)

# Создаем кнопку "Редактировать"

edit\_button = Button(z, text='Редактировать', command=lambda:

(self.db.updateRecord1(Name\_1.get(), Price\_2.get(), Quantity\_3.get(), id), self.viewRecords(), self.tableAssortment()))

edit\_button.place(x=10, y=160, width=100, height=30)

# Создаем кнопку "Закрыть"

close\_button = Button(z, text='Закрыть', command=closeFunc)

close\_button.place(x=115, y=160, width=100, height=30)

def idgive(self):

''' строки по id из БД '''

self.selected\_item = self.table.selection()[0] # получаем строку

values = self.table.item(self.selected\_item, option='values') # получаем значения строки

give\_id = values[0] # получаем id строки

self.db.addcart(give\_id) # добавляем строку по полученному id

def linegive(self):

''' Удаление выбранной строки из таблицы tkinter '''

self.db.selected\_item # добавляем полученную строку

def idDelete(self):

''' Удаление строки по id из БД '''

self.selected\_item = self.table.selection()[0] # получаем строку

values = self.table.item(self.selected\_item, option='values') # получаем значения строки

delete\_id = values[0] # получаем id строки

self.db.deleteRecords1(delete\_id) # удаляем строку по полученному id

def lineDelete(self):

''' Удаление выбранной строки из таблицы tkinter '''

self.table.delete(self.selected\_item) # удаляем полученную строку

class Shopping\_cart():

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

self.db = db # экземпляр класса DB

self.variable = []

self.selected\_item = 0

self.viewRecords()

def viewRecords(self):

''' Вывод данных '''

self.db.cur.execute(

'''SELECT \* FROM Shopping\_cart''')

self.variable.clear() # очищаем прошлые данные (чтобы не дублировались)

[self.variable.append(row) for row in self.db.cur.fetchall()] # записываем новый результат

def tableShopping\_cart(self):

''' Создание таблицу с помощью ttk.Treeview() '''

# Задаем расположение таблицы

frame = tk.Frame(root, width=100, height=100)

frame.place(x=10, y=140)

# Создаем заголовоки для таблицы

headers = ['ID', 'Наименование', 'Цена', 'Количество']

self.table = ttk.Treeview(frame, columns=headers, height=15, show='headings')

self.table.column('ID', width=50, anchor='center')

self.table.column('Наименование', width=240, anchor='center')

self.table.column('Цена', width=255, anchor='center')

self.table.column('Количество', width=255, anchor='center')

for header in headers: # заполняем заголовоки

self.table.heading(header, text=header)

for i in self.variable: # заполняем значения

self.table.insert('', tk.END, values=i)

# Создаем скролл для таблицы

scroll = ttk.Scrollbar(frame, command=self.table.yview)

self.table.configure(yscrollcommand=scroll.set)

scroll.pack(side=tk.RIGHT, fill=tk.Y)

self.table.pack(expand=tk.YES, fill=tk.BOTH)

def editShopping\_cart(self):

''' Редактирование таблицы '''

def closeFunc():

z.destroy() # закрываем окно

# Создаем новое окно

z = Toplevel()

z.geometry('270x100')

z.title('Edit')

z.resizable(False, False)

# Получаем данные строки по клику

selected\_item = self.table.selection()[0]

values = self.table.item(selected\_item, option='values')

# Заносим данные строки в переменные

id = values[0]

Quantity = values[3]

# Создаем метки и их расположение

Quantity\_label = Label(z, text='Количество').place(x=10, y=10, width=80, height=30)

# Переменные для ввода значений

Quantity\_3 = StringVar()

# Строки для ввода значений

Quantity\_entry = Entry(z, width=50, textvariable=Quantity\_3)

# Записываем значения из строк ввода

Quantity\_entry.insert(0, str(Quantity))

# Задаем расположение строк ввода

Quantity\_entry.place(x=130, y=10, width=130, height=30)

# Создаем кнопку "Редактировать"

edit\_button = Button(z, text='Редактировать', command=lambda:

(self.db.updateRecord2(Quantity\_3.get(), id), self.viewRecords(), self.tableShopping\_cart()))

edit\_button.place(x=10, y=50, width=100, height=30)

# Создаем кнопку "Закрыть"

close\_button = Button(z, text='Закрыть', command=closeFunc)

close\_button.place(x=115, y=50, width=100, height=30)

def idDelete(self):

''' Удаление строки по id из БД '''

self.selected\_item = self.table.selection()[0] # получаем строку

values = self.table.item(self.selected\_item, option='values') # получаем значения строки

delete\_id = values[0] # получаем id строки

self.db.deleteRecords2(delete\_id) # удаляем строку по полученному id

def lineDelete(self):

''' Удаление выбранной строки из таблицы tkinter '''

self.table.delete(self.selected\_item) # удаляем полученную строку

class Workers():

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

self.db = db # экземпляр класса DB

self.variable = []

self.selected\_item = 0

self.viewRecords()

def viewRecords(self):

''' Вывод данных '''

self.db.cur.execute(

'''SELECT \* FROM Workers''')

self.variable.clear() # очищаем прошлые данные (чтобы не дублировались)

[self.variable.append(row) for row in self.db.cur.fetchall()] # записываем новый результат

def tableWorkers(self):

''' Создание таблицу с помощью ttk.Treeview() '''

# Задаем расположение таблицы

frame = tk.Frame(root, width=100, height=100)

frame.place(x=10, y=140)

# Создаем заголовоки для таблицы

headers = ['ID', 'Фамилия', 'Имя','Должность']

self.table = ttk.Treeview(frame, columns=headers, height=15, show='headings')

self.table.column('ID', width=50, anchor='center')

self.table.column('Фамилия', width=250, anchor='center')

self.table.column('Имя', width=250, anchor='center')

self.table.column('Должность', width=250, anchor='center')

for header in headers: # заполняем заголовоки

self.table.heading(header, text=header)

for i in self.variable: # заполняем значения

self.table.insert('', tk.END, values=i)

# Создаем скролл для таблицы

scroll = ttk.Scrollbar(frame, command=self.table.yview)

self.table.configure(yscrollcommand=scroll.set)

scroll.pack(side=tk.RIGHT, fill=tk.Y)

self.table.pack(expand=tk.YES, fill=tk.BOTH)

def editWorkers(self):

''' Редактирование таблицы '''

def closeFunc():

z.destroy() # закрываем окно

# Создаем новое окно

z = Toplevel()

z.geometry('260x260')

z.title('Edit')

z.resizable(False, False)

# Получаем данные строки по клику

selected\_item = self.table.selection()[0]

values = self.table.item(selected\_item, option='values')

# Заносим данные строки в переменные

id = values[0]

Surname = values[1]

Name = values[2]

Post = values[3]

# Создаем метки и их расположение

Surname\_label = Label(z, text='Фамилия').place(x=20, y=10, width=70, height=30)

Name\_label = Label(z, text='Имя').place(x=20, y=60, width=70, height=30)

Post\_label = Label(z, text='Должность').place(x=20, y=100, width=70, height=30)

# Переменные для ввода значений

Surname\_1 = StringVar()

Name\_2 = StringVar()

Post\_3 = StringVar()

# Строки для ввода значений

Surname\_entry = Entry(z, width=50, textvariable=Surname\_1)

Name\_entry = Entry(z, width=50, textvariable=Name\_2)

Post\_entry = Entry(z, width=50, textvariable=Post\_3)

# Записываем значения из строк ввода

Surname\_entry.insert(0, str(Surname))

Name\_entry.insert(0, str(Name))

Post\_entry.insert(0, str(Post))

# Задаем расположение строк ввода

Surname\_entry.place(x=100, y=10, width=120, height=30)

Name\_entry.place(x=100, y=60, width=120, height=30)

Post\_entry.place(x=100, y=100, width=120, height=30)

# Создаем кнопку "Редактировать"

edit\_button = Button(z, text='Редактировать', command=lambda:

(self.db.updateRecord3(Surname\_1.get(), Name\_2.get(), Post\_3.get(), id), self.viewRecords(), self.tableWorkers()))

edit\_button.place(x=10, y=220, width=100, height=30)

# Создаем кнопку "Закрыть"

close\_button = Button(z, text='Закрыть', command=closeFunc)

close\_button.place(x=115, y=220, width=100, height=30)

def idDelete(self):

''' Удаление строки по id из БД '''

self.selected\_item = self.table.selection()[0] # получаем строку

values = self.table.item(self.selected\_item, option='values') # получаем значения строки

delete\_id = values[0] # получаем id строки

self.db.deleteRecords3(delete\_id) # удаляем строку по полученному id

def lineDelete(self):

''' Удаление выбранной строки из таблицы tkinter '''

self.table.delete(self.selected\_item) # удаляем полученную строку

class tkAssortment():

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

self.sale = sale # экземпляр класса Assortment()

self.db = db

def AssortmentButton(self):

''' Кнопка для открытия таблицы и её функций '''

def unionFunc():

''' Компонует все кнопки функции выше '''

def addWindow():

''' Создание нового окна для кнопки <Добавить> '''

def closeFunc():

s.destroy() # закрываем окно

# Создаем новое окно

s = Toplevel()

s.geometry('250x200')

s.title('Add')

s.resizable(False, False)

Name\_label = Label(s, text='Наименование').place(x=10, y=10, width=85, height=30)

Price\_label = Label(s, text='Цена').place(x=10, y=60, width=80, height=30)

Quantity\_label = Label(s, text='Количество').place(x=10, y=100, width=80, height=30)

name\_1 = StringVar()

name\_2 = StringVar()

name\_3 = StringVar()

Name\_entry = Entry(s, width=50, textvariable=name\_1).place(x=130, y=10, width=100, height=30)

Price\_entry = Entry(s, width=50, textvariable=name\_2).place(x=130, y=60, width=100, height=30)

Quantity\_entry = Entry(s, width=50, textvariable=name\_3).place(x=130, y=100, width=100, height=30)

add\_button = Button(s, text='Добавить', command=lambda:

(self.db.records1(name\_1.get(), name\_2.get(), name\_3.get()), self.sale.viewRecords(), self.sale.tableAssortment()))

add\_button.place(x=10, y=160, width=115, height=30)

close\_button = Button(s, text='Закрыть', command=closeFunc)

close\_button.place(x=130, y=160, width=80, height=30)

self.add\_image = tk.PhotoImage(file='pictures/add.png')

Assortment\_add\_button = Button(root, text='Добавить', image=self.add\_image, compound='top', command=addWindow)

Assortment\_add\_button.place(x=10, y=50, width=100, height=80)

self.edit\_image = tk.PhotoImage(file='pictures/edit.png')

Assortment\_edit\_button = Button(root, text='Редактировать', image=self.edit\_image, compound='top', command=lambda:

self.sale.editAssortment())

Assortment\_edit\_button.place(x=120, y=50, width=100, height=80)

self.delete\_image = tk.PhotoImage(file='pictures/delete.png')

Assortment\_edit\_button = Button(root, text='Удалить', image=self.delete\_image, compound='top', command=lambda:

(self.sale.idDelete(), self.sale.lineDelete()))

Assortment\_edit\_button.place(x=230, y=50, width=100, height=80)

self.give\_image = tk.PhotoImage(file='pictures/give.png')

Assortment\_addcarts\_button = Button(root, text='Добавление в корзину', image=self.give\_image, compound='top', command=lambda:

(self.sale.idgive(), self.sale.linegive()))

Assortment\_addcarts\_button.place(x=400, y=50, width=140, height=80)

Assortment\_button = Button(root, text='Assortment', command=lambda:

(unionFunc(), self.sale.viewRecords(), self.sale.tableAssortment()))

Assortment\_button.place(x=10, y=10, width=100, height=30)

class tkShopping\_cart():

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

self.cust = cust # экземпляр класса Shopping\_cart()

self.db = db

def Shopping\_cartButton(self):

''' Кнопка для открытия таблицы и её функций '''

def unionFunc():

''' Компонует все кнопки функции выше '''

def addWindow():

''' Создание нового окна для кнопки <Добавить> '''

def closeFunc():

s.destroy() # закрываем окно

# Создаем новое окно

s = Toplevel()

s.geometry('370x150')

s.title('Add')

s.resizable(False, False)

Quantity\_label = Label(s, text='Добавлять в корзину можно только в таблице ассортимента').place(x=10, y=10, width=350, height=30)

close\_button = Button(s, text='Закрыть', command=closeFunc)

close\_button.place(x=60, y=100, width=250, height=30)

self.add\_image = tk.PhotoImage(file='pictures/add.png')

Shopping\_cart\_add\_button = Button(root, text='Добавить', image=self.add\_image, compound='top', command=addWindow)

Shopping\_cart\_add\_button.place(x=10, y=50, width=100, height=80)

self.edit\_image = tk.PhotoImage(file='pictures/edit.png')

Shopping\_cart\_edit\_button = Button(root, text='Редактировать', image=self.edit\_image, compound='top', command=lambda:

self.cust.editShopping\_cart())

Shopping\_cart\_edit\_button.place(x=120, y=50, width=100, height=80)

self.delete\_image = tk.PhotoImage(file='pictures/delete.png')

Shopping\_cart\_edit\_button = Button(root, text='Удалить', image=self.delete\_image, compound='top', command=lambda:

(self.cust.idDelete(), self.cust.lineDelete()))

Shopping\_cart\_edit\_button.place(x=230, y=50, width=100, height=80)

self.give\_image = tk.PhotoImage(file='pictures/give.png')

Shopping\_cart\_addcarts\_button = Button(root, text='В этой таблице нельзя!', image=self.give\_image, compound='top', command=lambda:

())

Shopping\_cart\_addcarts\_button.place(x=400, y=50, width=140, height=80)

Shopping\_cart\_button = Button(root, text='Shopping\_cart', command=lambda:

(unionFunc(), self.cust.viewRecords(), self.cust.tableShopping\_cart()))

Shopping\_cart\_button.place(x=120, y=10, width=100, height=30)

class tkWorkers():

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

self.orde = orde # экземпляр класса Workers()

self.db = db

def WorkersButton(self):

''' Кнопка для открытия таблицы и её функций '''

def unionFunc():

''' Компонует все кнопки функции выше '''

def addWindow():

''' Создание нового окна для кнопки <Добавить> '''

def closeFunc():

s.destroy() # закрываем окно

# Создаем новое окно

s = Toplevel()

s.geometry('280x270')

s.title('Add')

s.resizable(False, False)

Surname\_label = Label(s, text='Фамилия').place(x=20, y=10, width=70, height=30)

Name\_label = Label(s, text='Имя').place(x=20, y=60, width=70, height=30)

Post\_label = Label(s, text='Должность').place(x=20, y=100, width=70, height=30)

name\_1 = StringVar()

name\_2 = StringVar()

name\_3 = StringVar()

Surname\_entry = Entry(s, width=50, textvariable=name\_1).place(x=130, y=10, width=120, height=30)

Name\_entry = Entry(s, width=50, textvariable=name\_2).place(x=130, y=60, width=120, height=30)

Post\_entry = Entry(s, width=50, textvariable=name\_3).place(x=130, y=100, width=120, height=30)

add\_button = Button(s, text='Добавить', command=lambda:

(self.db.records3(name\_1.get(), name\_2.get(), name\_3.get()), self.orde.viewRecords(), self.orde.tableWorkers()))

add\_button.place(x=10, y=230, width=100, height=30)

close\_button = Button(s, text='Закрыть', command=closeFunc)

close\_button.place(x=115, y=230, width=100, height=30)

self.add\_image = tk.PhotoImage(file='pictures/add.png')

Spec\_Sections\_add\_button = Button(root, text='Добавить', image=self.add\_image, compound='top', command=addWindow)

Spec\_Sections\_add\_button.place(x=10, y=50, width=100, height=80)

self.edit\_image = tk.PhotoImage(file='pictures/edit.png')

Spec\_Sections\_edit\_button = Button(root, text='Редактировать', image=self.edit\_image, compound='top', command=lambda:

self.orde.editWorkers())

Spec\_Sections\_edit\_button.place(x=120, y=50, width=100, height=80)

self.delete\_image = tk.PhotoImage(file='pictures/delete.png')

Spec\_Sections\_edit\_button = Button(root, text='Удалить', image=self.delete\_image, compound='top', command=lambda:

(self.orde.idDelete(), self.orde.lineDelete()))

Spec\_Sections\_edit\_button.place(x=230, y=50, width=100, height=80)

self.give\_image = tk.PhotoImage(file='pictures/give.png')

Shopping\_cart\_addcarts\_button = Button(root, text='В этой таблице нельзя!', image=self.give\_image, compound='top', command=lambda:

())

Shopping\_cart\_addcarts\_button.place(x=400, y=50, width=140, height=80)

Workers\_button = Button(root, text='Workers', command=lambda:

(unionFunc(), self.orde.viewRecords(), self.orde.tableWorkers()))

Workers\_button.place(x=230, y=10, width=100, height=30)

if \_\_name\_\_ == "\_\_main\_\_":

db = DB1()

root = tk.Tk()

root.geometry('830x485')

root.title('Клиент спортивного магазина')

root.resizable(False, False)

sale = Assortment()

cust = Shopping\_cart()

orde = Workers()

s = tkAssortment()

c = tkShopping\_cart()

o = tkWorkers()

s.AssortmentButton()

c.Shopping\_cartButton()

o.WorkersButton()

root.mainloop()

**Приложение 2. «db.py»**

import sqlite3

class DB1:

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

# Создаем подключение к БД

self.conn = sqlite3.connect('Sports\_store\_customer.db')

self.cur = self.conn.cursor()

self.selected\_item = 0

# Создаем таблицу Assortment(Ассортимент)

self.cur.execute(

'''CREATE TABLE IF NOT EXISTS Assortment (ID\_1 integer primary key AUTOINCREMENT NULL, 'Наименование' text, 'Цена' text, 'Количество' INT DEFAULT 1)''')

# Создаем таблицу Shopping\_cart(Корзина покупок)

self.cur.execute(

'''CREATE TABLE IF NOT EXISTS Shopping\_cart (ID\_2 integer primary key AUTOINCREMENT NULL, 'Наименование' text, 'Цена' text, 'Количество' INT )''')

# Создаем таблицу Workers(Работники)

self.cur.execute(

'''CREATE TABLE IF NOT EXISTS Workers (ID\_3 integer primary key AUTOINCREMENT NULL, 'Фамилия' text, 'Имя' text, 'Должность' text)''')

self.conn.commit()

def insertDataAssortment(self, Name, Price, Quantity):

''' Добавление данных для Assortment '''

self.cur.execute(

'''INSERT INTO Assortment('Наименование', 'Цена', 'Количество') VALUES (?, ?, ?)''', (Name, Price, Quantity))

self.conn.commit()

def insertDataWorkers(self, Surname, Name, Post):

''' Добавление данных для Workers '''

self.cur.execute(

'''INSERT INTO Workers('Фамилия', 'Имя', 'Должность') VALUES (?, ?, ?)''', (Surname, Name, Post))

self.conn.commit()

#Функции DB (Ассортимент)

def records1(self, Name, Price, Quantity):

''' Ввод новых данных '''

self.insertDataAssortment(Name, Price, Quantity)

def updateRecord1(self, Name, Price, Quantity, ID\_1):

''' Редактирование данных '''

self.cur.execute(

'''UPDATE Assortment SET 'Наименование'=?, 'Цена'=?, 'Количество'=? WHERE ID\_1=?''', (Name, Price, Quantity, ID\_1))

self.conn.commit()

def deleteRecords1(self, ID\_1):

''' Удаление результата '''

self.cur.execute(

'''DELETE FROM Assortment WHERE ID\_1=?''', (ID\_1,))

self.conn.commit()

def addcart(self, ID\_1):

''' Добавление товара в корзину '''

self.cur.execute(f'''INSERT INTO Shopping\_cart SELECT NULL, Наименование, Цена, Количество FROM Assortment WHERE ID\_1=?''', (ID\_1,))

#[self.variable.append(row) for row in self.db.cur.fetchall()] # записываем новый результат

self.conn.commit()

#Функции DB (Корзина покупок)

def records2(self, Name, Price, Quantity):

''' Ввод новых данных '''

self.insertDataShopping\_cart(Name, Price, Quantity)

def updateRecord2(self, Quantity, ID\_2):

''' Редактирование данных '''

self.cur.execute(

'''UPDATE Shopping\_cart SET 'Количество'=? WHERE ID\_2=?''', (Quantity, ID\_2))

self.conn.commit()

def deleteRecords2(self, ID\_2):

''' Удаление результата '''

self.cur.execute(

'''DELETE FROM Shopping\_cart WHERE ID\_2=?''', (ID\_2,))

self.conn.commit()

#Функции DB (Работники)

def records3(self, Surname, Name, Post):

''' Ввод новых данных '''

self.insertDataWorkers(Surname, Name, Post)

def updateRecord3(self, Surname, Name, Post, ID\_3):

''' Редактирование данных '''

self.cur.execute(

'''UPDATE Workers SET 'Фамилия'=?, 'Имя'=?, 'Должность'=? WHERE ID\_3=?''', (Surname, Name, Post, ID\_3))

self.conn.commit()

def deleteRecords3(self, ID\_3):

''' Удаление результата '''

self.cur.execute(

'''DELETE FROM Workers WHERE ID\_3=?''', (ID\_3,))

self.conn.commit()

1. 1) ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов [↑](#footnote-ref-1)
2. 2) ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов [↑](#footnote-ref-2)
3. 3) ГОСТ 19.104-78\* ЕСПД. Основные надписи [↑](#footnote-ref-3)
4. 4) ГОСТ 19.105-78\* ЕСПД. Общие требования к программным документам [↑](#footnote-ref-4)
5. 5) ГОСТ 19.106-78\* ЕСПД. Общие требования к программным документам, выполненным печатным способом [↑](#footnote-ref-5)
6. 6) ГОСТ 19.505-79\* ЕСПД. Руководство оператора. Требования к содержанию и оформлению [↑](#footnote-ref-6)
7. 7) ГОСТ 19.604-78\* ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом [↑](#footnote-ref-7)