Государственное бюджетное общеобразовательное учреждение "Республиканский физико-математический лицей-интернат".

**Индивидуальный проект**

Бухгалтерия в малом предприятии

Работу выполнил:

Марчуков Антон Игоревич,

Ученик 10 А класса

Руководитель:

Акоева Е.Н.

Владикавказ, 2023

**Оглавление**

[Введение 3](#_Toc132011874)

[Почему именно такое приложение 5](#_Toc132011875)

[Почему именно «Python» 6](#_Toc132011876)

[Цель 7](#_Toc132011877)

[Описание работы 8](#_Toc132011878)

[Работа программы 9](#_Toc132011879)

[Код программы 10](#_Toc132011880)

[Заключение 12](#_Toc132011881)

[Список использованной литературы 13](#_Toc132011882)

[Приложение 14](#_Toc132011883)

# Введение

То, что еще совсем недавно казалось новым и неизведанным, сегодня уже кажется обычным. Мы покоряем космос уже не в околоземном пространстве, а отправляем беспилотники на Марс, ведется разведка Сатурна и Юпитера. Когда-то об этом можно было прочесть только в фантастических книгах. Настало время реализации самых смелых фантазий современности. Человек изучает океан с помощью сверхсложный приборов и в онлайн-режиме это могут наблюдать миллионы пользователей Интернета. Глобальная паутина объединила все и вся. Люди из разных уголков нашей планеты свободно общаются в режиме реального времени друг с другом, обмениваются фото, видеозаписями, мнениями, обсуждают вопросы. С уверенностью можно сказать что, 21 век — век информационных технологий. Сегодня информационные технологии задействованы везде: в промышленности, в транспорте, науке, образовании, социальных структурах, государственном управлении, экономики и культуре. Мы совершаем [покупки в сети](https://berichnow.ru/stati/reklamnyie-meropriyatiya-ochen-vazhnyiy-ryichag-lyubogo-biznesa), работаем здесь, обучаемся дистанционно в учебных заведениях посредством Интернет-ресурсов, обращаемся в государственные органы и службы в онлайн-режиме. Скоро мы вообще перестанем представлять себе жизнь без информационных технологий.

На современном этапе развития компьютерных технологий трудно представить высококвалифицированного специалиста, который не владеет информационными технологиями. Чтобы свободно ориентироваться в информационных потоках современному человеку необходимо уметь получать, обрабатывать информацию с помощью компьютеров, других новейших средств, а также необходимо уметь программировать на языках высокого уровня.

Программирование… Многим миллионам людей на нашей планете эти слова ни о чём не говорят. Несомненно, у всех это слово на слуху, но что оно обозначает, многие наверняка даже не задумывались.

Программисты-профессионалы сейчас утверждают о том, что научиться программировать может каждый. Программирование сегодня становится второй грамотностью. Кто умеет программировать, становятся обладателями «суперсилы», способной претворять в жизнь самые невероятные вещи, помогать себе и окружающим людям быстрее решать их проблемы и задачи, делать жизнь всего человечества лучше. Языки программирования – является

актуальной темой в IT индустрии. Они помогают реализовать разные виды задач и воплотить свои мысли в реальность. От простых мини-игр, до огромных проектов в IT индустрии. Поэтому я решил попробовать научиться программировать самостоятельно.

# Почему именно такое приложение

Моя мама работает в магазине компьютерной технике, поэтому я с детства видел, как она работает в какой-то непонятной программе, меня это сильно интересовало. Повзрослев, я узнал, что это программа называется “1С Предприятие (Торговля + Склад)”. В ней ведется учет прихода и продаж товара, количество его на складе, установка цен номенклатуры. Рассчитывается ежемесячная прибыль, от чего зависит размер зарплаты каждого сотрудника предприятия. Начав изучать программирование, первое, что мне захотелось сделать – это подобие на эту программу. Конечно, я понимал, что у меня не получится сделать точную копию, так как над программой “1С Предприятие (Торговля + Склад)” трудилось немало профессиональных разработчиков. Но я попытался сделать хоть что – то подобное.

# Почему именно «Python»

Когда я решил изучать программирование, передо мной встал вопрос: «Какой же именно выбрать язык программирования?» Тогда я решил, что самым лучшим решением будет посмотреть в интернете. Первое, что мне выдал браузер – это статью с сайта TIOBE. В ней по состоянию на сентябрь 2022 года Python занимает первое место в рейтинге по популярности языков программирования. Так же я узнал, что «Языком года» по версии TIOBE Python объявлялся в 2007, 2010, 2018, 2020 и 2021 годах. Изучая дальше этот вопрос, я больше узнал об этом языке. Он используется в [анализе данных](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), [машинном обучении](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), и [веб-разработке](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0), а также в других сферах, включая [разработку игр](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80). Применяется язык многими крупными компаниями, такими как [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)) или [Facebook](https://ru.wikipedia.org/wiki/Facebook). Python является высокоуровневым языком с [динамической](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [строгой](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%B3%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) типизацией. Это означает, что [переменная](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) связывается с [типом](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) в момент [присваивания](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D1%81%D0%B2%D0%B0%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) [значения](https://ru.wikipedia.org/wiki/%D0%97%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), а не в момент объявления переменной. За счёт читабельности, простого синтаксиса и отсутствия необходимости в компиляции язык хорошо подходит для обучения программированию. Необычной особенностью языка является выделение [блоков кода](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%B0) пробельными отступами, за счёт чего на практике редко возникает необходимость обращаться к документации. Отладка же и экспериментирование в значительной степени облегчаются тем фактом, что язык является интерпретируемым. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) или [C++](https://ru.wikipedia.org/wiki/C%2B%2B). Но эти минусы мне показались не столь важными в моей работе. Я потратил много времени на сбор данных и мнение об этом языке и пришел к выводу, что он мне идеально подходит.

# Цель

* Написание собственного приложения
* Более подробное изучение «мира IT»

**Задачи**

* Изучение базовых аспектов языка программирования «Python»
* Изучение объектно-ориентированного программирования (ООП) в «Python»
* Работа со сторонними библиотека «Python», а именно «PyQt5»
* Изучение работы с базами данных
* Изучение работы с «SQL Server ManagementStudio**»**
* Изучение работы с «Qt Designer»

**Актуальность**

Оно создано для упрощения контроля финансовой части предприятия. С помощью него работники легче будут отслеживать заработанную плату, налоги, а также покупки конкретных людей и организации. Теперь каждый сотрудник предприятия будет более четко представлять его пенсионное накопление и вычеты наконец года

# Описание работы

Когда у меня появился примерный план моих действий, я начал работать. На изучение основы Python я потратил довольно много времени. Я записался на курс, в котором мне помогли с освоением базы. После прохождения данного курса, я неплохо знал различные типы данных в Python. Например, integer, float, string, list, set, dictionary, tuple. Так же я научился работать со встроенными модулями, логическими операциями, циклами, функциями и методами списков и строк.

Начав более подробное изучение ООП, я познакомился с тремя основными принципами. Первый полиморфизм - свойство кода работать с разными типами данных. Второй инкапсуляция - объединение данных и функций в единый компонент. Третий наследование - это процесс, когда один класс наследует атрибуты и методы другого.

Для реализации моей идеи мне понадобится создание своего собственного графического интерфейс. Для этого я решил воспользоваться библиотекой Python PyQt5. А так же для более легкого создания форм интерфейса, я решил изучить приложение «Qt Designer». (Рисунок 1)

В ходе разработки я понял, что для нормальной реализации, мне нужна база данных. Поэтому я начал изучение работы с ними на Python. Для упрощения я разобрался в «SQL Server Management Studio» – приложение для работы с базами данных. Итак, я создал свою собственную базу данных, состоящую из нескольких таблиц. (Рисунок 2) Она облегчила мою работу.

# Работа программы

Проект представляет собой приложение, помогающее проще контролировать финансовую часть предприятия. В данном приложении я реализовал расчет прибыли, зарплаты, налогов в предприятии. Запуская приложение, пользователь видит диалоговое окно, в котором он должен ввести имя покупателя. (Рисунок 3) После он переходит на основное окно. (Рисунок 4) Дальше пользователь должен создать список всех товаров, которые он хочет приобрести. Для этого существует кнопка «Добавить товар в список». Далее пользователю предлагается ввести количество рабочих дней каждого работника в месяце. Если пользователь захочет узнать валовую прибыль для какого-то конкретного товара, то он может выбрать этот товар в выпадающем списке в правом верхнем углу. После нажатия на кнопку “Посчитать” каждое поле заполнится значениями. При нажатии на кнопку сохранить, вся информация будет сохранена в csv файл. В файле сохраняется информация в виде: имя покупателя, зарплата каждого из сотрудников, предстоящие выплаты налога на каждого сотрудника, а также остаток. Так же на этом окне расположена кнопка “показать”. При нажатии на нее будет вызвано новое окно, с изображением выбранного товара. (Рисунок 5). Но эта кнопка работает, если выбран какой-то определенный товар. Ведь все товары одновременно нельзя показать. Кнопка «Добавить товар в базу данных» отвечает за добавление нового товара в базу данных. (Рисунок 6) После добавление пользователь сможет выбрать этот товар в текущий список. Цены товара, а также информации о каждой покупке пользователя хранится в базе данных.

# Код программы

Мой проект содержит 6 файлов типа Python file, 2 файлов типов UI и 1 файл типа Data Base. В файле под именем main.py ([Код 1](#_Код_1)) находится основная часть кода. В нем создан мой собственный класс MyClass. При запуске он создает графический интерфейс. Сам он находится в файле под именем qt\_1.py. Этот файл создан с помощью команды pyuic5, которая из файла qt\_1.ui создала файла типа py. Так же в \_\_init\_\_ происходит инициализация всех виджетов и других классов. (Рисунок 4) В проекте реализованы такие функции:

1. Count, предназначенная для обработки всей введенной информации и подсчетов.
2. Save отвечает за сохранение данных в csv файл и в базу данных.
3. add\_product отвечает за вызов нового окна для добавления нового товара в базу данных. Форма 2-го окна представляет собой другой класс Window\_Add([Код 2](#_Код_2)) в файле add\_product.py. Данный файл был создан с помощью команды pyuic5 и файла add.py
4. add\_product\_list необходима для создания списка продажи
5. change\_val\_cost нужна для подсчета валовая стоимости товара
6. change\_pol адресована для смены текущего пользователя
7. rb предназначена для обработки сигналов радиокнопки
8. picther\_show нужна для отображения фото товара. Для этого я использовал другой класс под именем MainWindow\_Picther, находящийся в файле picther.py([Код 3](#_Код_3))
9. counting\_cost необходимая для подсчета разницы между закупочной и продажной стоимостью
10. count\_percent отвечает за подсчет процентов

В проекте я продумал возможные ошибки, такие как: ввод букв вместо цифр, некорректно введенные числа или активация подсчета без введенных данных. При возникновении ошибки пользователь увидит новое окно, в котором будет говориться об возникшей ошибке. Это функция реализована с помощью класса Window\_Exaption находившегося в файле Exeption.py. ([Код 4](#_Код_4)) В проекте были использованы так же библиотеки:

* os для работы с операционной системой
* sys для получения необходимой информации об операционной системе
* sqlite3 для работы с SQL запросами
* csv для записи в с csv файл
* subprocess для перезапуска программы

# Заключение

Я изучил интерпретируемый язык программирования Python. С помощью него смог написать свое собственное приложение. Оно создано для упрощения контроля финансовой части предприятия. С помощью него работники легче будут отслеживать заработанную плату, налоги, а также покупки конкретных людей и организации. Теперь каждый сотрудник предприятия будет более четко представлять его пенсионное накопление и вычеты наконец года. А так же я изучил работу с базами данных, графическими интерфейсами, невстроенными модулями Python, такими как PyQt5.

# Список использованных источников

1. Курс для изучения «Python». Stepik [Электронный ресурс] — Режим доступа: https://stepik.org/course/67/promo, свободный
2. Документация по «Python» [Электронный ресурс] — Режим доступа: https://docs.python.org/3/, свободный
3. Документация по PyQt5 [Электронный ресурс] — Режим доступа: https://doc.qt.io/qtforpython/, свободный
4. Документация по приложению «Qt Designer» [Электронный ресурс] — Режим доступа: https://doc.qt.io/qt-6/qtdesigner-manual.html, свободный
5. Документация по приложению «SQL Server Management Studio» [Электронный ресурс] — Режим доступа: https://metanit.com/sql/sqlserver/, свободный

# Приложение

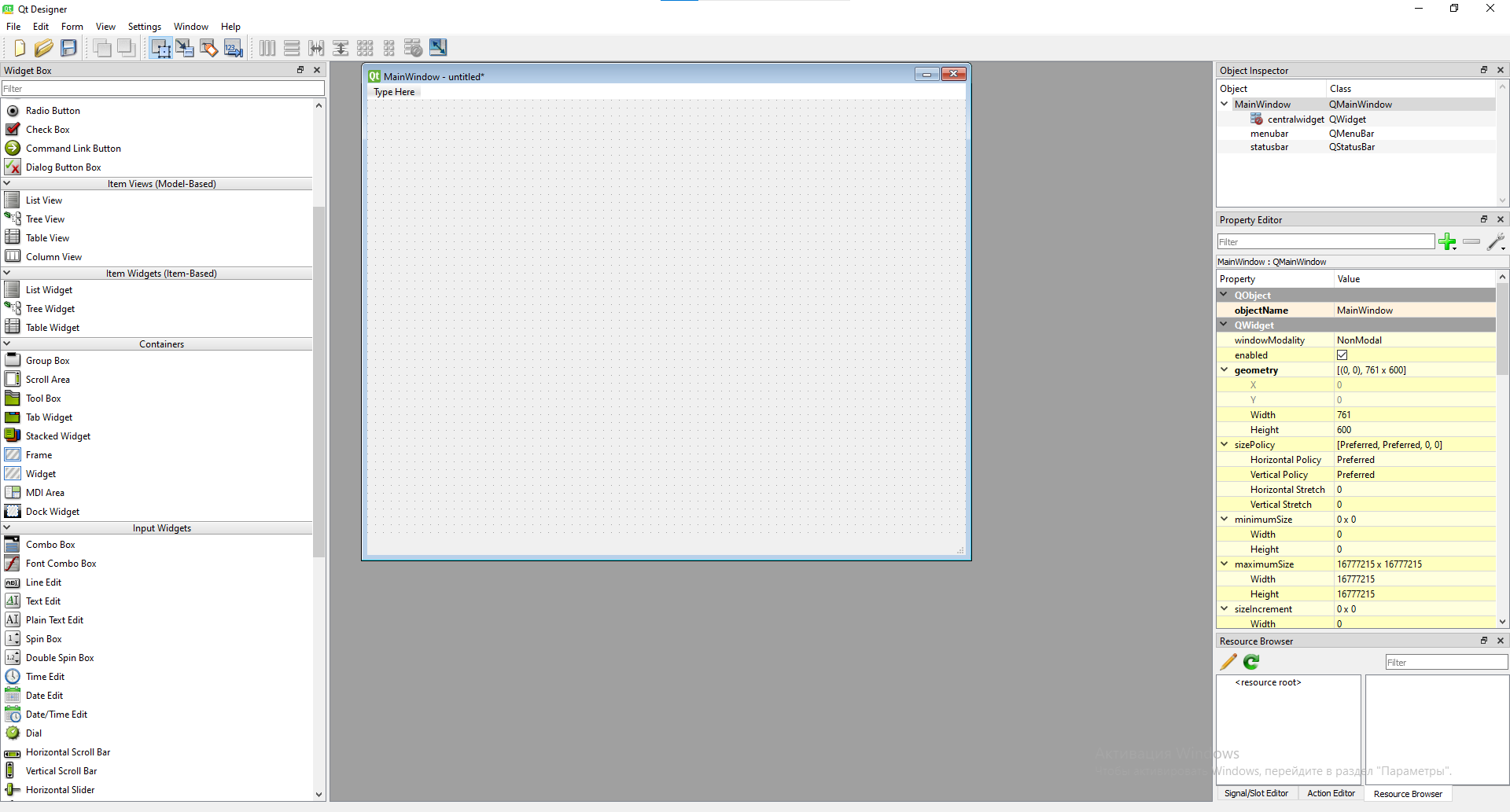


Рисунок 1

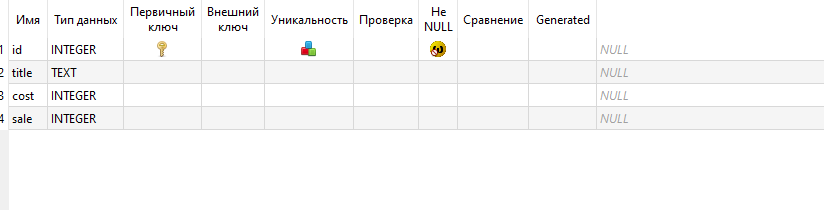
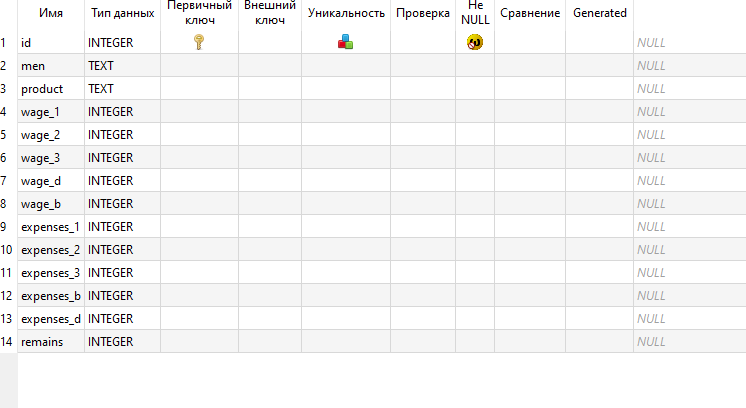


Рисунок 2

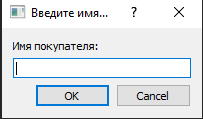


Рисунок 3

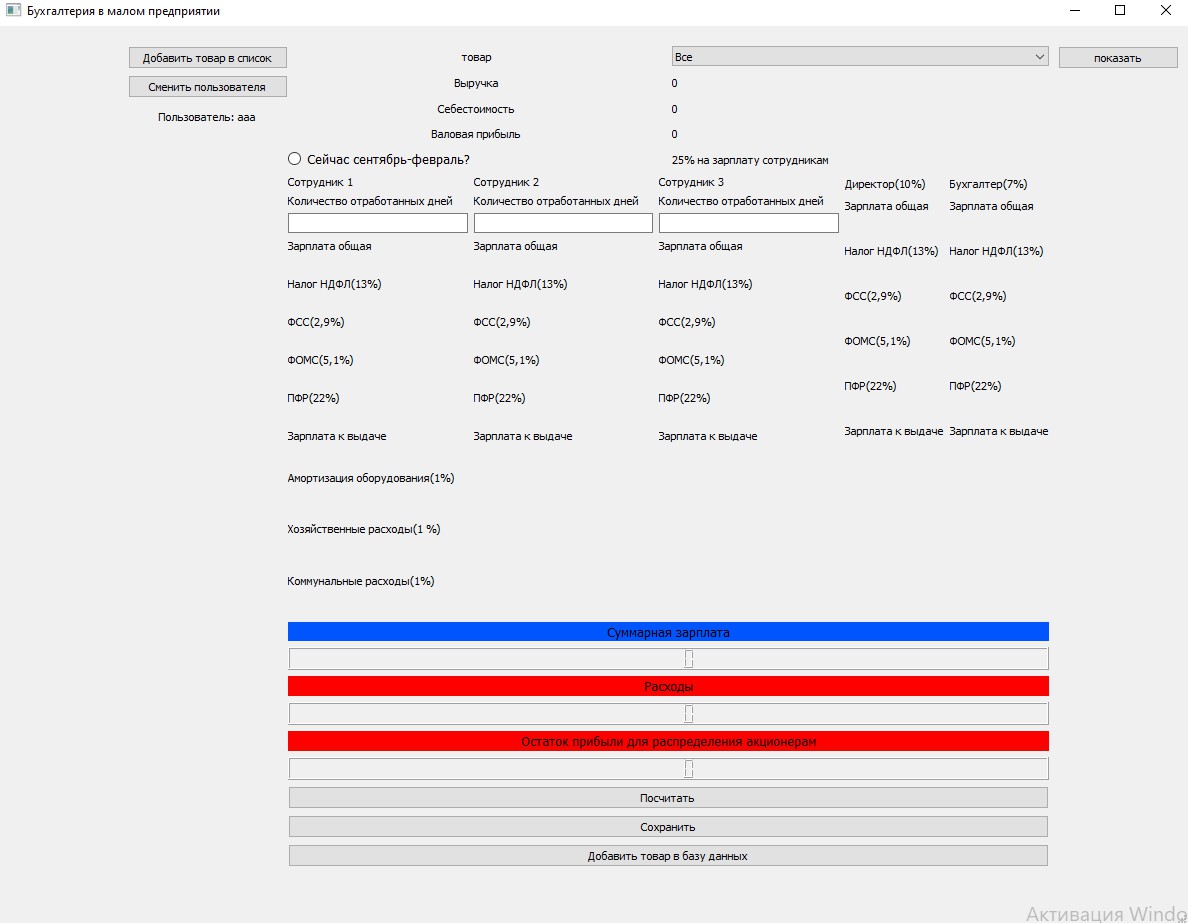


Рисунок 4



Рисунок 5

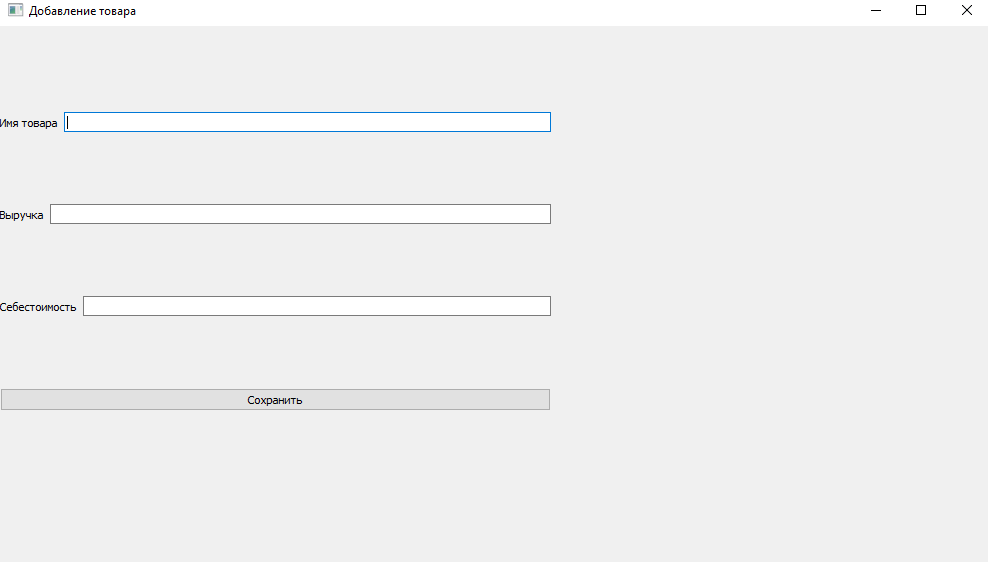


Рисунок 6

# Код 1

# import os import sys from PyQt5.QtWidgets import QApplication, QMainWindow, QInputDialog from qt\_1 import Ui\_mainWindow import sqlite3 from picther import MainWindow\_Picther from Exeption import Window\_Exaption from add\_product import Window\_Add from math import floor import csv import subprocess def counting\_cost(cost, sale): return sale - cost def count\_percent(wage, p): a = wage \* (p / 100) return a class MyClass(QMainWindow, Ui\_mainWindow): def \_\_init\_\_(self): super().\_\_init\_\_() self.setupUi(self) # используется для того чтобы пользователь ввел свое имя и товар self.name\_pol, ok\_pressed\_pol = QInputDialog.getText(self, "Введите имя покупателя", "Имя покупателя:") if not ok\_pressed\_pol: sys.exit() self.comboBox\_name.addItem('Все') # кнопка отвечающая за показ товара пользователю self.label\_sale.setText('0') self.label\_cost.setText('0') self.label\_val\_cost\_2.setText('0') self.exaption = Window\_Exaption() self.pushButton\_show.clicked.connect(self.picther\_show) # радио кнопка отвечающая за общий процент зарплаты сотрудникам self.label\_procent.setText('25% на зарплату сотрудникам') self.radioButton.clicked.connect(self.rb) self.pushButton\_count.clicked.connect(self.count) # кнопка сохранение в csv файл self.comboBox\_name.activated.connect(self.change\_val\_cost) self.pushButton\_save.clicked.connect(self.save) self.add\_p = Window\_Add(list\_product) self.pushButton\_add.clicked.connect(self.add\_product) self.pushButton\_list\_product.clicked.connect(self.add\_product\_list) self.pushButton\_list\_product.clicked.connect(self.change\_val\_cost) self.pushButton\_change\_pol.clicked.connect(self.change\_pol) self.label\_current\_user.setText('Пользователь: ' + self.name\_pol) def change\_pol(self): subprocess.call([sys.executable, os.path.relpath(\_\_file\_\_)] + sys.argv[1:]) def change\_val\_cost(self): # Подключаем БД con = sqlite3.connect('qt\_bd.db') cur = con.cursor() # выбираем из таблицы стоимость товара if self.comboBox\_name.currentText() == 'Все': sale = 0 cost = 0 for index\_sale in range(self.comboBox\_name.count() - 1): product = self.comboBox\_name.itemText(index\_sale + 1) sale\_current = cur.execute(f''' SELECT sale FROM product WHERE title = '{product}' ''') sale += list(sale\_current)[0][0] cost\_current = cur.execute(f''' SELECT cost FROM product WHERE title = '{product}' ''') cost += list(cost\_current)[0][0] else: product = self.comboBox\_name.itemText(self.comboBox\_name.currentIndex()) sale\_current = cur.execute(f''' SELECT sale FROM product WHERE title = '{product}' ''') sale = list(sale\_current)[0][0] cost\_current = cur.execute(f''' SELECT cost FROM product WHERE title = '{product}' ''') cost = list(cost\_current)[0][0] # превращаем sqlite в строку self.label\_sale.setText(str(sale)) self.label\_cost.setText(str(cost)) self.label\_val\_cost\_2.setText(str(counting\_cost(cost, sale))) con.commit() con.close() def add\_product\_list(self): self.name\_product, ok\_pressed\_product = QInputDialog.getItem(self, "Введите имя товара", "Имя товара:", tuple(list\_product), 0, False) if ok\_pressed\_product: self.comboBox\_name.addItem(self.name\_product) def add\_product(self): global list\_product self.add\_p.show() list\_product = self.add\_p.list\_product # функция сохранения def save(self): if self.label\_wages\_full\_1.text() == '': self.label\_print.setText('Сначала посчитайте зарплату и расходы') else: self.wage\_1 = round(float(self.label\_wages\_full\_1.text()), 2) self.wage\_2 = round(float(self.label\_wages\_full\_2.text()), 2) self.wage\_3 = round(float(self.label\_wages\_full\_3.text()), 2) self.wage\_b = round(float(self.label\_wages\_full\_b.text()), 2) self.wage\_d = round(float(self.label\_wages\_full\_d.text()), 2) self.expenses\_1 = round(count\_percent(self.wage\_1, 43), 2) self.expenses\_2 = round(count\_percent(self.wage\_2, 43), 2) self.expenses\_3 = round(count\_percent(self.wage\_3, 43), 2) self.expenses\_b = round(count\_percent(self.wage\_b, 43), 2) self.expenses\_d = round(count\_percent(self.wage\_d, 43), 2) self.remains = round(float(self.lcdNumber\_3.value()), 2) con = sqlite3.connect('qt\_bd.db') cur = con.cursor() if self.comboBox\_name.currentText() == 'Все': list\_name = [] for index\_sale in range(self.comboBox\_name.count() - 1): product = self.comboBox\_name.itemText(index\_sale + 1) name = cur.execute(f''' SELECT title FROM product WHERE title = '{product}' ''') list\_name.append(list(name)[0][0]) name = ', '.join(list\_name) else: name = self.comboBox\_name.currentText() # добавление пользователя cur.execute(f'''INSERT INTO shopper(men, product, wage\_1, wage\_2, wage\_3, wage\_d, wage\_b, expenses\_1, expenses\_2, expenses\_3, expenses\_b, expenses\_d, remains) VALUES("{self.name\_pol}", "{name}", "{self.wage\_1}", "{self.wage\_2}", "{self.wage\_3}" , "{self.wage\_d}", "{self.wage\_b}", "{self.expenses\_1}", "{self.expenses\_2}" , "{self.expenses\_3}", "{self.expenses\_b}", "{self.expenses\_d}", "{self.remains}") ''') le = len(list(cur.execute('''SELECT men FROM shopper'''))) # сохранение данных в csv файл with open('data/results.csv', 'w', newline='') as csvfile: writer = csv.writer( csvfile, delimiter=';', quotechar='"', quoting=csv.QUOTE\_MINIMAL) writer.writerow(['Покупатель', 'Товар', 'Зарплата 1', 'Зарплата 2', 'Зарплата 3', 'Зарплата Бугалтера', 'Зарплата Директора', 'Налоги 1', 'Налоги 2', 'Налоги 3', 'Налоги Бугалтера', 'Налоги Директора', 'Остаток']) for i in range(le): list\_bd = list(cur.execute(f""" SELECT men, product, wage\_1, wage\_2, wage\_3, wage\_d, wage\_b, expenses\_1, expenses\_2, expenses\_3, expenses\_b, expenses\_d, remains FROM shopper WHERE id == {i + 1} """))[0] writer.writerow(list\_bd) con.commit() con.close() self.label\_print.setText('Таблица успешно сохранена') # функция подсчитывающая всю зарплату и расходы def count(self): val = int(self.label\_val\_cost\_2.text()) sum\_wages = val \* (self.rb() / 100) # проверка на корректные вводные данные if not self.lineEdit\_count\_day\_1.text().isdigit(): self.exaption.show() return 0 if not self.lineEdit\_count\_day\_2.text().isdigit(): self.exaption.show() return 0 if not self.lineEdit\_count\_day\_3.text().isdigit(): self.exaption.show() return 0 count\_1 = int(self.lineEdit\_count\_day\_1.text()) if count\_1 > 31: self.exaption.show() return 0 count\_2 = int(self.lineEdit\_count\_day\_2.text()) if count\_2 > 31: self.exaption.show() return 0 count\_3 = int(self.lineEdit\_count\_day\_3.text()) if count\_3 > 31: self.exaption.show() return 0 # подсчет зарплат работников и администрации sum\_count = sum([count\_1, count\_2, count\_3]) count\_wages\_day = round(sum\_wages / sum\_count, 1) wage\_1 = floor(count\_wages\_day \* count\_1) wage\_2 = floor(count\_wages\_day \* count\_2) wage\_3 = floor(count\_wages\_day \* count\_3) wage\_d = int(val \* 0.15) wage\_b = int(val \* 0.1) # показ полное зарплаты без учета уплаты налогов self.label\_wages\_full\_1.setText(str(wage\_1)) self.label\_wages\_full\_2.setText(str(wage\_2)) self.label\_wages\_full\_3.setText(str(wage\_3)) self.label\_wages\_full\_d.setText(str(wage\_d)) self.label\_wages\_full\_b.setText(str(wage\_b)) # показ количество рублей пойдущих на НДФЛ self.label\_ndfl\_1.setText(str(round(count\_percent(wage\_1, 13), 2))) self.label\_ndfl\_2.setText(str(round(count\_percent(wage\_2, 13), 2))) self.label\_ndfl\_3.setText(str(round(count\_percent(wage\_3, 13), 2))) self.label\_ndfl\_d.setText(str(round(count\_percent(wage\_d, 13), 2))) self.label\_ndfl\_b.setText(str(round(count\_percent(wage\_b, 13), 2))) # показ количество рублей пойдущих на ФСС self.label\_fss\_1.setText(str(round(count\_percent(wage\_1, 2.9), 2))) self.label\_fss\_2.setText(str(round(count\_percent(wage\_2, 2.9), 2))) self.label\_fss\_3.setText(str(round(count\_percent(wage\_3, 2.9), 2))) self.label\_fss\_d.setText(str(round(count\_percent(wage\_d, 2.9), 2))) self.label\_fss\_b.setText(str(round(count\_percent(wage\_b, 2.9), 2))) # показ количество рублей пойдущих на ФОМС self.label\_foms\_1.setText(str(round(count\_percent(wage\_1, 5.1), 2))) self.label\_foms\_2.setText(str(round(count\_percent(wage\_2, 5.1), 2))) self.label\_foms\_3.setText(str(round(count\_percent(wage\_3, 5.1), 2))) self.label\_foms\_d.setText(str(round(count\_percent(wage\_d, 5.1), 2))) self.label\_foms\_b.setText(str(round(count\_percent(wage\_b, 5.1), 2))) # показ количество рублей пойдущих на ПФР self.label\_pfr\_1.setText(str(round(count\_percent(wage\_1, 22), 2))) self.label\_pfr\_2.setText(str(round(count\_percent(wage\_2, 22), 2))) self.label\_pfr\_3.setText(str(round(count\_percent(wage\_3, 22), 2))) self.label\_pfr\_b.setText(str(round(count\_percent(wage\_d, 22), 2))) self.label\_pfr\_d.setText(str(round(count\_percent(wage\_b, 22), 2))) # Зарплата которую сотрудники получат на руки self.label\_wages\_out\_1.setText(str(round(wage\_1 - count\_percent(wage\_1, 43), 2))) self.label\_wages\_out\_2.setText(str(round(wage\_2 - count\_percent(wage\_2, 43), 2))) self.label\_wages\_out\_3.setText(str(round(wage\_3 - count\_percent(wage\_3, 43), 2))) self.label\_wages\_out\_d.setText(str(round(wage\_d - count\_percent(wage\_d, 43), 2))) self.label\_wages\_out\_b\_2.setText(str(round(wage\_b - count\_percent(wage\_b, 43), 2))) # расходы на комунлку, хозрасходы, амортизацию оборудования self.label\_amort.setText(str(round(count\_percent(val, 1), 2))) self.label\_com.setText(str(round(count\_percent(val, 1), 2))) self.label\_hoz.setText(str(round(count\_percent(val, 1), 2))) wage\_b = int(self.label\_wages\_full\_b.text()) wage\_d = int(self.label\_wages\_full\_d.text()) sum\_wages += wage\_b + wage\_d self.lcdNumber\_1.display(sum\_wages) # создаем список расходов sum\_expenses = [] # добавляем в список расходов все имеющиеся расходы sum\_expenses.append(self.label\_amort.text()) sum\_expenses.append(self.label\_hoz.text()) sum\_expenses.append(self.label\_com.text()) sum\_expenses.append(self.label\_pfr\_1.text()) sum\_expenses.append(self.label\_pfr\_2.text()) sum\_expenses.append(self.label\_pfr\_3.text()) sum\_expenses.append(self.label\_pfr\_d.text()) sum\_expenses.append(self.label\_pfr\_b.text()) sum\_expenses.append(self.label\_foms\_1.text()) sum\_expenses.append(self.label\_foms\_2.text()) sum\_expenses.append(self.label\_foms\_3.text()) sum\_expenses.append(self.label\_foms\_d.text()) sum\_expenses.append(self.label\_foms\_b.text()) sum\_expenses.append(self.label\_fss\_1.text()) sum\_expenses.append(self.label\_fss\_2.text()) sum\_expenses.append(self.label\_fss\_3.text()) sum\_expenses.append(self.label\_fss\_d.text()) sum\_expenses.append(self.label\_fss\_b.text()) sum\_expenses.append(self.label\_ndfl\_1.text()) sum\_expenses.append(self.label\_ndfl\_2.text()) sum\_expenses.append(self.label\_ndfl\_3.text()) sum\_expenses.append(self.label\_ndfl\_b.text()) sum\_expenses.append(self.label\_ndfl\_d.text()) # каждый элемент списка преобразуем в число sum\_expenses = list(map(float, sum\_expenses)) sum\_expenses = sum(sum\_expenses) # выводим на экран суммарные расходы self.lcdNumber\_2.display(sum\_expenses) # создаем список остатка прибыли sum\_remains = [] sum\_remains.append(sum\_wages) sum\_remains.append(sum\_expenses) self.lcdNumber\_3.display(val - sum(sum\_remains)) # функция проверяющая состояние радио кнопки def rb(self): if self.radioButton.isChecked(): self.label\_procent.setText('20% на зарплату сотрудникам') return 20 else: self.label\_procent.setText('25% на зарплату сотрудникам') return 25 # функция показывающая товар def picther\_show(self): name = self.comboBox\_name.currentText() if name == 'Все': self.exaption.show() else: self.show\_product = MainWindow\_Picther(name) self.show\_product.show() con = sqlite3.connect('qt\_bd.db') cur = con.cursor() cur.execute('''CREATE TABLE IF NOT EXISTS product (id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE NOT NULL, title TEXT, cost INTEGER, sale INTEGER);''') list\_product = [] [list\_product.append(i[0]) for i in list(cur.execute(''' SELECT title FROM product '''))] if 'Ноутбук' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Ноутбук", 30000, 40000)''') if 'Монитор' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Монитор", 15000, 20000)''') if 'Оперативная память' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Оперативная память", 1500, 2000)''') if 'Видео карта' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Видео карта", 30000, 40000)''') if 'Блок питания' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Блок питания", 5000, 6000)''') if 'Жесткий диск' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Жесткий диск", 2500, 3000)''') if 'Сетевая карта' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Сетевая карта", 500, 700)''') if 'Звуковая карта' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Звуковая карта", 2500, 3000)''') if 'Системный блок' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Системный блок", 25000, 30000)''') if 'Мышь' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Мышь", 500, 800)''') if 'Процессор' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Процессор", 10000, 15000)''') if 'Клавиатура' not in list\_product: cur.execute('''INSERT INTO product(title, cost, sale) VALUES("Клавиатура", 1500, 2000)''') cur.execute('''CREATE TABLE IF NOT EXISTS shopper (id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE NOT NULL, men TEXT, product TEXT wage\_1 INTEGER wage\_2 INTEGER wage\_3 INTEGER wage\_d INTEGER wage\_b INTEGER expenses\_1 INTEGER expenses\_2 INTEGER expenses\_3 INTEGER expenses\_b INTEGER expenses\_d INTEGER remains INTEGER);''') con.commit() con.close() if \_\_name\_\_ == '\_\_main\_\_': app = QApplication(sys.argv) ex = MyClass() ex.show() sys.exit(app.exec\_())

# Код 2

# from PyQt5.QtWidgets import QMainWindow, QLabel from add import Ui\_MainWindow import sqlite3 class Window\_Add(QMainWindow, Ui\_MainWindow): def \_\_init\_\_(self, list\_product): super().\_\_init\_\_() self.list\_product = list\_product self.setupUi(self) self.pushButton\_save.clicked.connect(self.add\_product) def add\_product(self): self.name = self.lineEdit\_name.text() # Проверка на корректные введенные данные try: self.sale = int(self.lineEdit\_sale.text()) self.cost = int(self.lineEdit\_cost.text()) if self.cost < self.sale and self.cost > 0 and self.sale > 0: con = sqlite3.connect('qt\_bd.db') cur = con.cursor() # добавляем в список все товары if self.name in self.list\_product: self.label\_print.setText('Этот товар уже есть в списке') else: cur.execute(f'''INSERT INTO product(title, cost, sale) VALUES("{self.name}", "{self.cost}", "{self.sale}")''') self.label\_print.setText('Товар успешно добавлен') self.list\_product.append(self.name) con.commit() con.close() else: self.label\_print.setText('Введите удовлетворящие цены(выручка должна быть больше)') except: self.label\_print.setText('Введите цену числами')

# Код 3

# from PyQt5.QtGui import QPixmap from PyQt5.QtWidgets import QMainWindow, QLabel, QPushButton class MainWindow\_Picther(QMainWindow): def \_\_init\_\_(self, name): super(MainWindow\_Picther, self).\_\_init\_\_() list\_photo = ['Блок питания', 'Видео карта', 'Жесткий диск', 'Звуковая карта', 'Клавиатура', 'Монитор', 'Мышь', 'Ноутбук', 'Оперативная память', 'Процессор', 'Сетевая карта', 'Системный блок'] self.title = "Товар" self.setWindowTitle(self.title) label = QLabel(self) self.setCentralWidget(label) self.name = name if self.name not in list\_photo: label.setText("Извините, но данного товара нет в списке возможно отображаемых") else: label = QLabel(self) pixmap = QPixmap(f'data\{self.name}.jpg') label.setPixmap(pixmap) self.setCentralWidget(label) self.resize(pixmap.width(), pixmap.height())

# Код 4

# from PyQt5.QtWidgets import QMainWindow, QLabel from PyQt5.QtCore import Qt class Window\_Exaption(QMainWindow): def \_\_init\_\_(self): super(Window\_Exaption, self).\_\_init\_\_() self.setWindowTitle('ОШИБКА') self.label = QLabel(self) self.label.setText('Ошибка!!!') self.label.setAlignment(Qt.AlignCenter) self.setCentralWidget(self.label) self.label.setStyleSheet("background-color: rgb(255, 0, 0)")