**Geekbrains**

**Дипломный проект**

**Аналитик больших данных**

**Цифровые Профессии**

**Тема проекта:**

**Разработка приложения для калькуляции и организация процесса приготовления готовых блюд на предприятиях общественного питания.**

**Исаева Татьяна Викторовна**

**пгт. Шерегеш**

**2023г**

**Тема проекта: Разработка приложения для калькуляции и организация процесса приготовления готовых блюд на предприятиях общественного питания.**

**Цель:**

Дипломный проект посвящен решению вопросов калькуляции готовых блюд. И как более обширная тема – вся организация процесса приготовления блюд, начиная с планирования закупок продуктов и заканчивая составлением меню.

[Задачи:](https://github.com/MorgunovaAO/Diploma_project/blob/main/MorgunovaA_group3189.md#%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B8)

1. Изучить темы и литературу, посвященные калькуляции готовых блюд;
2. Познакомиться с языком программирования Python;
3. Освоить азы работы с MySQL;
4. Изучить Qt Designer;
5. Познакомиться с принципами работы предприятий общественного питания;
6. Разработать структуру базы данных на MySQL ;
7. Разработать дизайн приложения в Qt Designer;
8. Прописать функционал для виджетов приложения в Visual Studio Code.

Структура дипломного проекта:

Введение

**Глава 1. Калькуляция готовых блюд**

1.1 Что такое калькуляция, её виды и способы расчёта

1.2 Калькуляционная карта

1.3 Этапы калькуляции

1.4 Технологическая карта в общепите

**Глава 2. Установки и настройки, необходимые для разработки приложения**

* 1. Установка Python ( не ниже 3 версии)
  2. Установка MySQL
  3. Установка Qt Designer
  4. Установка QtPy5

**Глава 3. Этапы работы по разработке приложения**

3.1 Создать необходимые для проекта таблицы с помощью MySQ

3.2 Создать необходимые для проекта интерфейсы с помощью программы декоратора интерфейсов Qt Designer

3.3 Написать программу для взаимодействия табличных данных с интерфейсом с помощью фреймворка Qt для языка программирования Python – QtPy5

Заключение. **Какие доработки ещё можно провести для улучшения приложения.**

Список используемой литературы.

**Введение.**

Ни одно предприятие, в том числе и заведение общепита, не сможет плодотворно работать и успешно развиваться в современном мире без автоматизированной системы учета. На сегодняшний день на российском рынке представлено много различных IT-продуктов для предприятий общественного питания. У каждого продукта есть свои особенности, свои плюсы и минусы, а также стратегии для дальнейшего развития. Все это не помеха для создания и появления новых IT-продуктов, позволяющих автоматизировать учет анализ и управление производственных процессов в сфере общественного питания.

Приложение задумано для использования непосредственными исполнителями ролей – приемщиком товара или кладовщиком, поваром и менеджером по закупкам. Обычно пользователем программ по управлению и планированию баз данных является бухгалтер либо оператор, которые далеки от производства, как физически, так и профессионально. Они не понимают и не знают все нюансы работы общественного питания по простой причине, у них другая сфера деятельности и образование. Зачастую коммуникации между отделами производства оставляют желать лучшего, поэтому сами работники спланируют и организуют свою работу лучше, чем при помощи посредников.

Организация работы по снабжению необходимыми товарами:

1. Повар составляет меню и передает список необходимых товаров менеджеру по закупкам.
2. Руководитель одобряет меню.
3. Менеджер изучает рынок предложений, находит поставщиков и заказывает товар, необходимый для производства.
4. Приемщик или кладовщик принимает поступивший от поставщиков товар, заносит его в базу данных и передает в производство.
5. Руководитель, совместно с бухгалтерией разрабатывают и вносят в базу данных торговые наценки.
6. Повар расходует товар на приготовление блюд и соответственно оформляет в базе данных расход на производство и приход готовых блюд.
7. В приложении происходит автоматический расчёт товарных остатков и продажных цен.
8. Повар корректирует меню в соответствии со спросом клиентов.
9. Менеджер на основании товарных остатков базы данных и возможных изменений меню составляет последующие заявки на товар.

**Глава 1. Калькуляция готовых блюд.**

1.1 Что такое калькуляция, её виды и способы расчёта

Калькуляция —  это определение затрат в стоимостной (денежной) форме на производство единицы или группы единиц изделий, или на отдельные виды производств.

Калькуляция даёт возможность определить плановую или фактическую себестоимость объекта или изделия и является основой для их оценки.

**Виды калькуляций.**

Калькуляция бывают следующих видов:

- Плановая калькуляция — себестоимость, составленная на планируемый период на основе прогрессивных норм затрат труда и средств производства, отражающих дальнейший технический прогресс и улучшение организации производства и труда:

- Нормативная калькуляция — текущая плановая калькуляция, составленная нормативным методом учёта затрат по текущим, действующим нормам, характеризующих достигнутый уровень затрат;

- Проектная калькуляция — перспективная плановая калькуляция, необходимая для определения эффективности капитальных вложений и новой техники.

Отчётная калькуляция — себестоимость, исчисленная на основе данных учёта и характеризующая фактический уровень затрат.

Себестоимость — это стоимостная оценка текущих затрат предприятия на производство и реализацию продукции.

**Виды себестоимости**

Себестоимость готовой продукции изменяется в зависимости от объёма затрат при её изготовлении. Таким образом, существуют следующие виды себестоимости:

- Прямая себестоимость — это затраты на сырьё и материалы, комплектующие, возвратные отходы, з/п рабочим, налоги и отчисления от з/п, а также расходы на оборудование;

- Цеховая себестоимость — включает прямую себестоимость и цеховые затраты, участвующие в обслуживание и управление цехом и других производственных структур, которые непосредственно участвовали в процессе изготовления определенного набора товаров и услуг;

- Производственная себестоимость — включает цеховую себестоимость и общехозяйственные расходы;

- Полная себестоимость — включает производственную себестоимость и затраты организации на реализацию продукции и внепроизводственные расходы.

**Калькуляция блюд в общественном питании**

Единой системы расчета продажной цены блюд в организациях общественного питания нет. В школьных столовых расчет может осуществляться одним способом, а в престижном ресторане совершенно другим. Но, наиболее оптимальным вариантом организации вычислительной системы в кафе и ресторанах является внедрение финансово-бухгалтерской системы, которая связывает все материальные потоки общепита. К таким системам можно отнести, например, систему 1С “Общественное питание для организации” и 1С “Ресторанный менеджмент”.

Также расчет блюд в документе excel будет удобным в любом заведении общественного питания и позволит выполнить качественный расчет калькуляции себестоимости блюд.

Кафе, рестораны и другие заведения общественного питания должны иметь четкую методику определения цены блюд. Это необходимо для того, чтобы предприятия могли своевременно регулировать цены на свою продукцию, получать прибыль от продажи и не отталкивать клиентов резким повышением цен.

1.2 Калькуляционная карта

На калькуляционной карте делается расчет стоимости одной порции, но также должен быть расчет на стоимость 10 или 100 порций. При расчете себестоимости и продажной цены одной порции, данные будут неверными, так как количество продуктов для нее будет взято больше, чем необходимо.

Также необходимо учитывать человеческий фактор. Фактическое использование ингредиентов будет отличаться от определенной нормы на одну порцию. К примеру, в норме предусмотрено 30 грамм соуса на одну порцию салата, а повар использовал 25 грамм в одну порцию и 35 в другую. Для каждой порции эти неточности существенны, но при расчете 10 порций они могут быть исправлены.

Калькуляционная карта — документ, необходимый для каждого блюда, которое реализуется в общепите. Калькуляция блюд дает возможность получить точную информацию о весе, стоимости сырья и себестоимости блюда.

Все данные для этого расчета берутся из рецепта. Калькуляция блюда выполняется в двух документах:

1.  Для поваров на кухне.

2.  Для бухгалтерии.

1.3 Этапы калькуляции

Чтобы рассчитать стоимость блюда проводят следующие действия:

-   Определяют блюда, на которые нужно выполнить расчет;

-   Основываясь на рецептах каждого блюда, определяют расход продуктов на одно блюдо;

-   Определяют закупочную цену ингредиентов;

-   Рассчитывают цену готового блюда.

Чтобы рассчитать цену готового блюда, стоимость сырья умножают на торговую наценку (проценты, определенные руководителем предприятия).

На размер наценки влияет опыт и класс шеф-повара, стоимость его работы. А еще — расходы на амортизацию оборудования, электроэнергию, аренду помещения и другие сопутствующие расходы заведения.

Важно не только наличие калькуляционной карты, но и правильность ее заполнения. Чтобы правильно составить калькуляционную карту необходимо помнить, что таблица затрат ОП-1 заполняется на основании технологического документа. В технологическом документе содержится полный рецепт блюда и его выход.

Все блюда и полуфабрикаты на предприятии имеют технологические и расчетные листы. Но, в технологической карте содержится информация только о рецепте блюда, а финансовая сторона не указывается. Такая карта используется поварами. А в расчетных листах представлена информация о себестоимости блюда и прибыли. Такие листы необходимы бухгалтерии и руководству общепита.

При этом технологическая карта и расчетная карта взаимосвязаны. Если допущена ошибка при заполнении технологической карты, то неверные данные будут и в расчетной карте.

1.4 Технологическая карта в общепите

Технологическая карта представляет собой документ, в котором отображается рецепт и основные питательные характеристики конкретного блюда. Работа общепита строго контролируется различными нормативными документами и стандартами. Но, основным нормативным документом является ГОСТ 31987-2012. В ГОСТ 31987-2012 указано, что производство блюд в заведениях общественного питания должно осуществляться в соответствии с технической документацией.

Технологическая карта (рецептура) – это документ общепита, по которому готовит повар или бармен, а бухгалтер-калькулятор общепита составляет калькуляцию и рассчитывает себестоимость продукции. Его наличие обязательно для любой организации, работающей в сфере общественного питания и самостоятельно занимающейся приготовлением продукции. Она составляется, если продукт приготовлен в точном соответствии со Сборником рецептур.

В остальных случаях, составляется Технико-технологическая карта (ТТК), которая отличается от такого документа, как Технологическая карта голубцы, наличием раздела «Показатели качества и безопасности блюда», в который входят:

1. Требования к подаче и реализации блюда,

2. Органолептические показатели,

3. Физико-химические показатели,

4. Микробиология.

Технологическая карта разрабатывается каждым предприятием индивидуально. Она утверждается руководителем предприятия или его замом. Сколько времени карта будет актуальной предприятие тоже определяет самостоятельно.

При этом если в рецепт блюда вносятся изменения, то технологическая карта тоже переоформляется.

Многие продукты при термической обработке теряют или прибавляют в весе и объеме. Например, рыба ужаривается и вес готового продукта уменьшается, а вермишель, наоборот, становится тяжелее после варки. Это важно учитывать при составлении технологической карты и расчете веса блюд.

Глава 2. Установки и настройки, необходимые для разработки приложения

Приложение разработано в редакторе исходного кода - Visual Studio Code

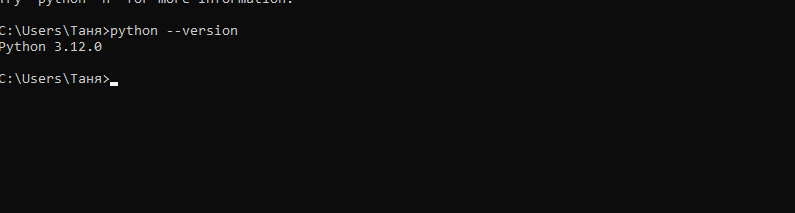
* 1. Установка Python ( не ниже 3 версии)

Python – это высокоуровневый объектно-ориентированный, очень популярный язык программирования.

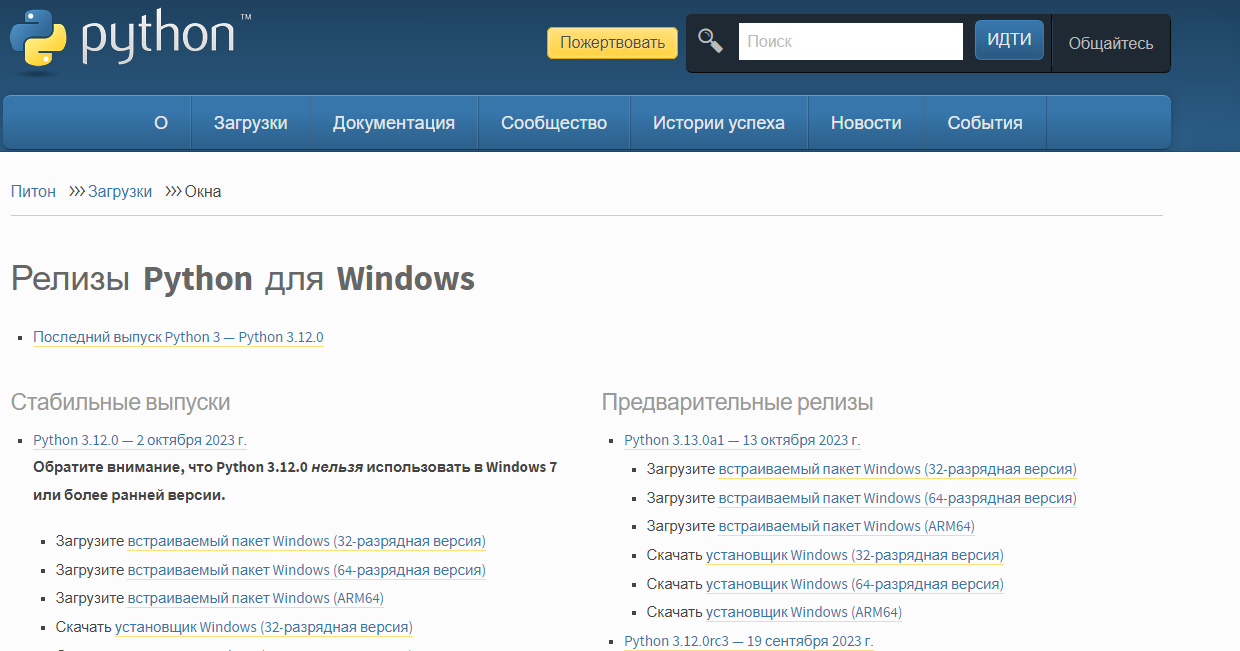
Сначала нужно проверить, возможно, на вашем компьютере python уже установлен. Откройте командную строку или терминал и введите:

python –version

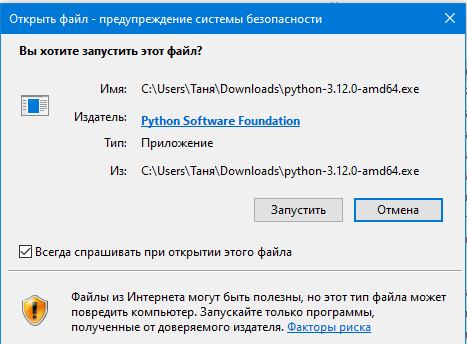
Если Python установлен командная строка выведет его версию:



Загрузить Python для Windows нужно с официального веб-сайта: <https://www.python.org/downloads/windows/>, выбрав подходящую версию.



Далее запускаем установку, в ходе которой нужно выбрать (установить флажок) на пункте «Add Python 3.8 to the PATH» - это добавление Python к системной переменной PATH.

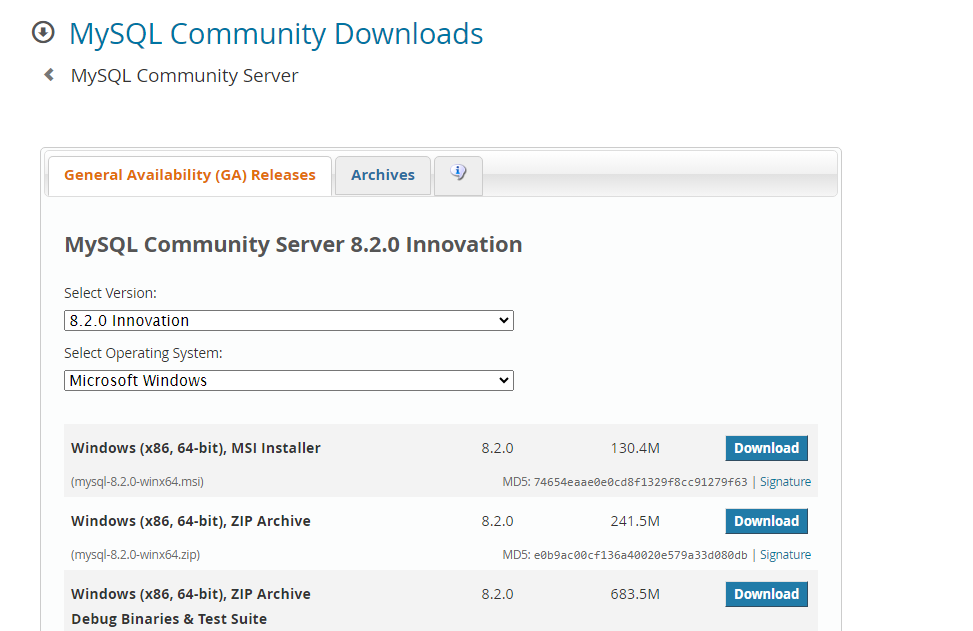


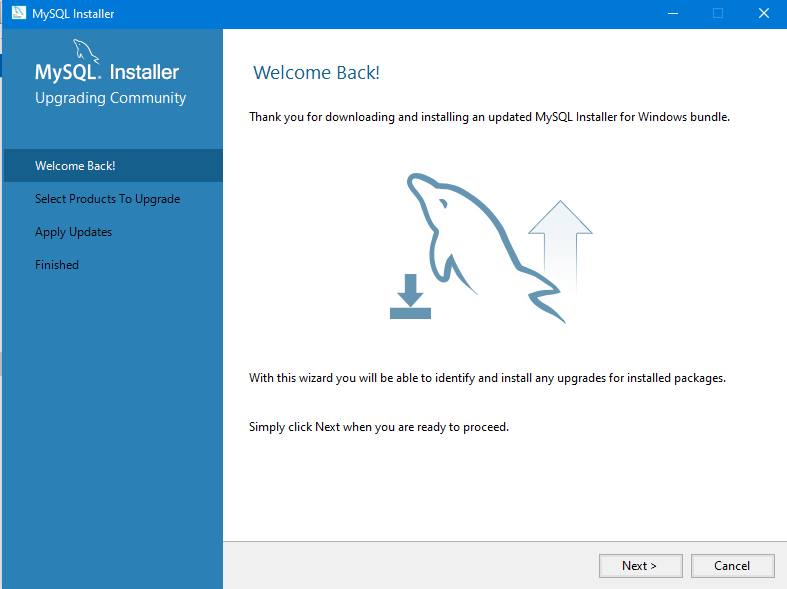
2.2 Установка MySQL

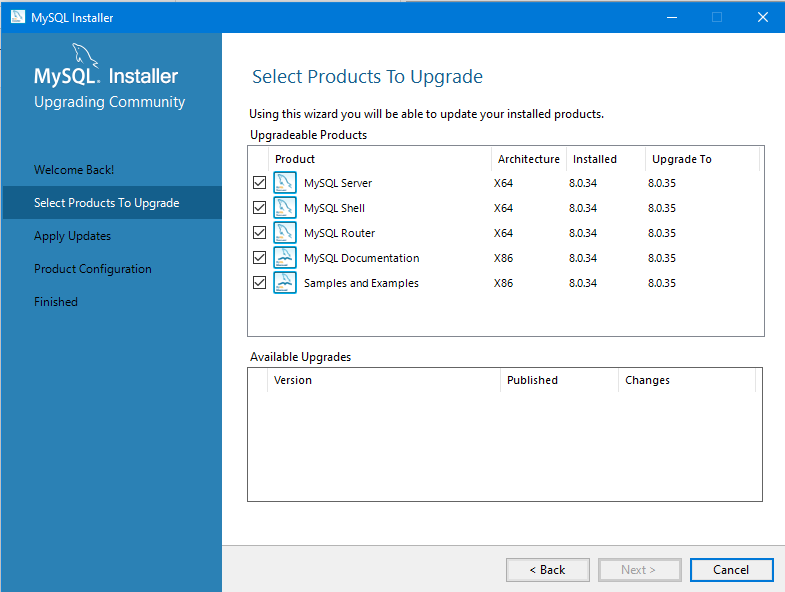
MySQL - это реляционная система управления базами данных. Она подходит для малых и средних приложений. Обычно MySQL используется как сервер, к которому обращаются клиенты.

Загружаем установщик MySQL для Windows с данного адреса <https://dev.mysql.com/downloads/installer/> и запускаем его.

MySQL имеет API и коннекторы для многих языков программирования , в том числе и Python.







При настройке установки нажмите Выборочная и выберите MySQL Server и connector/ODBC.

В разделе High Availability нужно выбрать Standalone MySQL Server / Classic MySQL Replication.

В разделе Type and Networking из раскрывающегося списка Config Type выберите Server Computer.

В разделе Authentication Method нужно выбрать вариант, который рекомендуется Use Strong Password Encryption for Authentication.

В разделе Accounts and Roles нужно дважды ввести пароль в поле MySQL Root Password. Рекомендуется также создать учетную запись базы данных.

Далее оставьте все без изменений и дождитесь окончания установки MySQL Server.

2.3 Установка Qt Designer

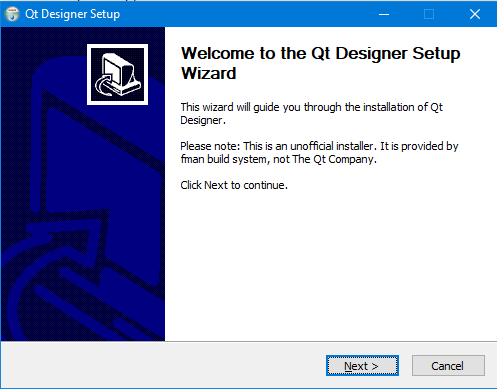
Qt Designer – является инструментом для проектирования и создания графических пользовательских интерфейсов (GUI) с использованием библиотеки PyQt5. Созданные интерфейсы сохраняются в формате .ui, который потом можно просто загрузить в ваше Python-приложение. Все параметры, установленные в Qt Designer для виджета при необходимости можно изменять динамически внутри самого кода, то есть графический интерфейс можно собрать (нарисовать) в Qt Designer, а помощью кода откорректировать его и настроить управление.

Qt Designer для Windows

- Скачайте установочный файл Qt Designer с адреса:

https://build-system.fman.io/static/public/files/Qt%20Designer%20Setup.exe,

- Запустите установочный файл,



- Установите приложение, следуя его подсказкам,

- Начните использование.

* 1. Установка QtPy5

PyQt5 является более продвинутой и функциональной библиотекой для создания графических интерфейсов на Python. Она предоставляет широкий спектр виджетов и возможностей для создания сложных интерфейсов. Для начала работы с PyQt5 установите соответствующий пакет:

pip install PyQt5

Для разработки приложений с помощью QtPy5 необходимо её тщательно изучить.

В состав библиотеки **PyQt5** входит очень много модулей,

Вот список самых важных из них:

* **QtCore** - содержит классы, не связанные с реализацией графического интерфейса. От него зависят все остальные модули;
* **QtGui** - содержит классы, реализующие низкоуровневую работу с оконными элементами, обработку сигналов, вывод двухмерной графики и текста и др.;
* **QtWidgets** - содержит классы, реализующие компоненты графического интерфейса: окна, диалоговые окна, надписи, кнопки, текстовые поля и др.;
* **QtWebKit** - включает низкоуровневые классы для отображения **Web**-страниц;
* **QtWebKitWidgets** - реализует высокоуровневые компоненты графического интерфейса, предназначенные для вывода **Web**-страниц и использующие модуль **QtWebKit**;
* **QtMultimedia** - включает низкоуровневые классы для работы с мультимедиа;
* **QtMultimediaWidgets** - реализует высокоуровневые компоненты графического интерфейса с мультимедиа и использующие модуль **QtMultimedia**;
* **QtPrintSupport** - содержит классы, обеспечивающие поддержку печати и предварительного просмотра документов;
* **QtSql** - включает поддержку работы с базами данных, а также реализацию **SQLite**;
* **QtSvg** - позволяет работать с векторной графикой **(SVG)**;
* **QtOpenGL** - обеспечивает поддержку **OpenGL**;
* **QtNetwork** - содержит классы, предназначенные для работы с сетью;
* **QtXml** и **QtXmlPatterns** - предназначены для обработки **XML**;
* **QtHelp** - содержат инструменты для создания интерактивных справочных систем;
* **QtWinExtras** - включает поддержку специфических возможностей **Microsoft Windows**;

**Qt** - включает классы из всех модулей сразу

Основные компоненты PyQt5:

1. QWidget и QDialog - окно

window **=** QtWidgets.QWidget()

создает объект окна в виде объекта класса QWidget. Этот класс наследуют практически все классы, реализующие компоненты графического интерфейса. Поэтому такой компонент, не имеющий родителя, выводится в своем собственном окне.

1. QPushButton – кнопка

btnQuit **=** QtWidgets .QPushButton

создает объект-кнопки. Текст, который будет отображен на кнопке, задается в качестве параметра в конструкторе класса QPushButton.

1. QLabel - текстовая метка

label **=** QtWidgets.QLabel

создает объект надписи. Текст надписи задается в качестве параметра в конструкторе

класса QLabel.

1. QLineEdit - текстовое поле для ввода/вывода

a = self.lineEdit\_a.text()

создает объект  который разрешает вводить и редактировать простую строку простого текста.

1. Конвертация файла «ui» в скрипт Python

(PyQt5 UI code generator)

pyuic5 name.ui — o name.py

Запускаем из папки с файлом ui в cmd, после чего появляется “py” скрипт в той же папке.

Сигналы и слоты

btnQuit.clicked.connect(app.quit)

назначает обработчик сигнала clicked () кнопки, активизирующегося при возникновениии события нажатия кнопки.

Событие - это уведомление о том, что в программе что-либо произошло: пользователь щелкнул мышью, нажал клавишу на клавиатуре, переместил окно и др.

Сигнал - это особый объект, соответствующий событию определенного типа и генерируемый каждый раз, когда возникает событие, относящееся к этому типу.

Обработчик сигнала - это метод (или функция), связанный с сигналом и вызьваемый при его генерировании. Обработчик назначается сигналу вызовом у последнего метода connect (). Метод, используемый в качестве обработчика, называется слотом.

Сигнал clicked (), генерируемый при возникновении события нажатии кнопки, доступен

через одноименный атрибут класса кнопки. Обработчик представляет собой метод qui t () объекта программы, немедленно завершающий ее работу.

Отметим, что не все события связаны со слотами. Часть событий обрабатьmается особымиспециальными методами, называемых обработчиками событий.

window. show ( )

выводит на экран окно и все компоненты, которые мы ранее в него добавили.

sys.exit(app.exec() )

запускает цикл обработки сигналов в программе.

Код, расположенный после вызова метода ехес (), будет выполнен только после завершения работы программы, - поскольку результат выполнения метода ехес () мы передаем функции exit (), дальнейшее выполнение программы будет прекращено, а код возвратит- передан операционной системе.

**Глава 3. Этапы работы по разработке приложения**

Перед началом разработки приложения нужно детально продумать его дизайн, параметры и функционал, а также структуру базы данных.

3.1 Создание необходимых для проекта таблицы с помощью MySQ

Сначала я продумала структуру базы данных:

CREATE DATABASE IF NOT EXISTS Catalog\_001;

USE Catalog\_001;

CREATE TABLE IF NOT EXISTS supplier\_list # список поставщиков

(

id\_supplier INT PRIMARY KEY AUTO\_INCREMENT, #(id поставщика)

name\_supplier VARCHAR(40) NOT NULL UNIQUE, #(наименование поставщика)

inn\_supplier VARCHAR(25) NOT NULL, #(инн поставщика)

address\_supplier VARCHAR(50) NOT NULL, #(адрес поставщика)

telephone\_supplier VARCHAR(30) NOT NULL #(телефон поставщика)

);

CREATE TABLE IF NOT EXISTS list\_of\_units # список единиц измерения

(

id\_unit INT PRIMARY KEY AUTO\_INCREMENT, #(id единицы измерения)

name\_unit VARCHAR(6) NOT NULL #(наименование единицы измерения)

);

CREATE TABLE IF NOT EXISTS group\_product # группа товаров

(

id\_group\_product INT PRIMARY KEY AUTO\_INCREMENT, #(id группы товаров)

name\_group\_product VARCHAR(40) NOT NULL UNIQUE #( название группы товаров)

);

CREATE TABLE IF NOT EXISTS group\_receipt # группы поступлений

(

id\_group\_receipt INT PRIMARY KEY AUTO\_INCREMENT, #(id группы поступления)

name\_group\_receipt VARCHAR(25) NOT NULL, # (наименование группы поступления)

total\_sum\_receipt FLOAT NOT NULL # (общая сумма по группе поступления)

);

CREATE TABLE IF NOT EXISTS group\_consumption # группы расходов

(

id\_group\_consumption INT PRIMARY KEY AUTO\_INCREMENT, #(id группы расходов)

name\_group\_consumption VARCHAR(20) NOT NULL, #(наименование группы расходов)

total\_sum\_consumption FLOAT NOT NULL #(общая сумма по группе расхода)

);

CREATE TABLE IF NOT EXISTS Nomenclature # номенклатура

(

name\_product\_Nomenclature VARCHAR(100) NOT NULL UNIQUE, #(наименование товара)

id\_product INT PRIMARY KEY AUTO\_INCREMENT, #(id товара)

barcode\_Nomenclature FLOAT NOT NULL UNIQUE ,

name\_unit\_Nomenclature VARCHAR(25) NOT NULL, #(единица измерения)

balance\_Nomenclature FLOAT, #(остаток)

cost\_price FLOAT, #(остаток)

margin INT NOT NULL, #(наценка)

retai\_price INT, #(розничная цена)

id\_group\_product\_Nomenclature INT , #(группа товаров)

total\_product FLOAT, #(итого по товару)

FOREIGN KEY (id\_group\_product\_Nomenclature) REFERENCES group\_product(id\_group\_product)

);

CREATE TABLE IF NOT EXISTS Receipt\_of\_goods\_and\_ready\_meals # поступление товаров и готовых блюд

(

id\_group\_receipt INT, #(id группы поступления)

id\_group\_receipt\_Receipt\_of\_goods INT NOT NULL, #(наименование группы поступления)

id\_receipt INT PRIMARY KEY AUTO\_INCREMENT, #(id поступления)

date\_001 DATETIME NOT NULL, #(дата и время поступления)

document\_base VARCHAR(25) NOT NULL, #(документ основание)

id\_supplier\_Receipt\_of\_goods INT NOT NULL, #(id поставщика)

sum\_document INT NOT NULL, #(сумма по документу)

total\_receipt\_supplier INT NOT NULL, #(итого поступления по поставщику)

total\_group\_receipt INT NOT NULL, #(итого по группе поступления)

FOREIGN KEY (id\_group\_receipt\_Receipt\_of\_goods) REFERENCES group\_receipt(id\_group\_receipt),

FOREIGN KEY (id\_supplier\_Receipt\_of\_goods) REFERENCES supplier\_list(id\_supplier)

);

CREATE TABLE IF NOT EXISTS Consumption\_of\_goods # расход товаров и готовых блюд

(

id\_group\_consumption INT, #(id группы расходов)

id\_consumption INT PRIMARY KEY AUTO\_INCREMENT, #(id расхода)

id\_group\_consumption\_Consumption\_of\_goods INT NOT NULL, #(наименование группы расходов)

date\_001 DATETIME NOT NULL, #(дата и время поступления)

document\_base VARCHAR(25), #(документ основание)

id\_supplier\_Consumption\_of\_goods INT NOT NULL, #(имя поставщика)

sum\_document INT NOT NULL, #(сумма по документу)

total\_consumption\_supplier INT NOT NULL, #(итого поступления по поставщику)

total\_group\_consumption INT NOT NULL, #итого по группе поступления)

FOREIGN KEY (id\_group\_consumption\_Consumption\_of\_goods) REFERENCES group\_consumption(id\_group\_consumption),

FOREIGN KEY (id\_supplier\_Consumption\_of\_goods) REFERENCES supplier\_list(id\_supplier)

);

CREATE TABLE IF NOT EXISTS technological\_cards\_list # список технологических карт

(

Id\_technological\_cards INT NOT NULL, #(id технологической карты)

id\_product\_technological\_cards INT NOT NULL, #(наименование товара)

barcode\_technological\_cards INT NOT NULL, #(штрих-код),

document\_base VARCHAR(25), # документ основание

id\_group\_product\_technological\_cards INT NOT NULL, #(группа товаров)

cooking\_method VARCHAR(100) NOT NULL, #(способ приготовления)

quantity\_unit\_receipt FLOAT NOT NULL, #(вес поступления)

quantity\_dish INT NOT NULL, #(количество штук готового блюда)

total\_cost\_price FLOAT NOT NULL, #(итого себестоимость)

dish\_output\_group\_product INT NOT NULL, #(выход готового блюда)

margin INT NOT NULL, #(наценка)

retai\_price INT NOT NULL, #(розничная цена),

FOREIGN KEY (id\_product\_technological\_cards) REFERENCES Nomenclature(id\_product),

FOREIGN KEY (id\_group\_product\_technological\_cards) REFERENCES group\_product(id\_group\_product)

);

CREATE TABLE IF NOT EXISTS goods\_for\_consumption # список товаров к расходу

(

Id\_list\_of\_consumables INT PRIMARY KEY AUTO\_INCREMENT, #(id списка к расходу)

id\_product\_goods\_for\_consumption INT NOT NULL, #(наименование товара)

id\_unit\_goods\_for\_consumption INT NOT NULL, #(единица измерения)

balance\_goods\_for\_consumption FLOAT NOT NULL, #(остаток)

id\_group\_product\_goods\_for\_consumption INT NOT NULL, #(группа товаров)

cost\_price FLOAT NOT NULL, #(себестоимость )

quantity\_unit\_consumption FLOAT NOT NULL, #(количество единиц измерения к расходу)

sum\_product FLOAT NOT NULL, #(сумма по товару)

FOREIGN KEY (id\_product\_goods\_for\_consumption) REFERENCES Nomenclature(Id\_product),

FOREIGN KEY (id\_group\_product\_goods\_for\_consumption) REFERENCES group\_product(id\_group\_product),

FOREIGN KEY (id\_unit\_goods\_for\_consumption) REFERENCES list\_of\_units(Id\_unit)

);

CREATE TABLE IF NOT EXISTS list\_of\_dishes #перечень блюд

(

id\_dish INT PRIMARY KEY AUTO\_INCREMENT, #(id блюда)

id\_product\_list\_of\_dishes INT NOT NULL, # (наименование товара)

barcode\_list\_of\_dishes INT NOT NULL, #(штрих код)

name\_unit VARCHAR(6) NOT NULL, #(единица измерения)

balance\_list\_of\_dishes INT NOT NULL, #(остаток)

id\_group\_product\_list\_of\_dishes INT NOT NULL, #(группа товаров)

FOREIGN KEY (id\_product\_list\_of\_dishes) REFERENCES Nomenclature(id\_product),

FOREIGN KEY (id\_group\_product\_list\_of\_dishes) REFERENCES group\_product(id\_group\_product)

);

CREATE TABLE IF NOT EXISTS arrival\_goods # список товаров прихода

(

id\_arrival\_goods\_list INT PRIMARY KEY AUTO\_INCREMENT, #(id списка к расходу)

id\_product\_goods\_for\_consumption INT NOT NULL, #(наименование товара)

id\_unit\_goods\_for\_consumption INT NOT NULL, #(наименование единицы измерения)

balance\_arrival\_goods FLOAT NOT NULL, #(остаток)

id\_group\_product\_goods\_for\_consumption INT NOT NULL, #(группа товаров)

cost\_price FLOAT NOT NULL, #(себестоимость )

arrival\_quantity FLOAT NOT NULL, #(количество единиц измерения к расходу)

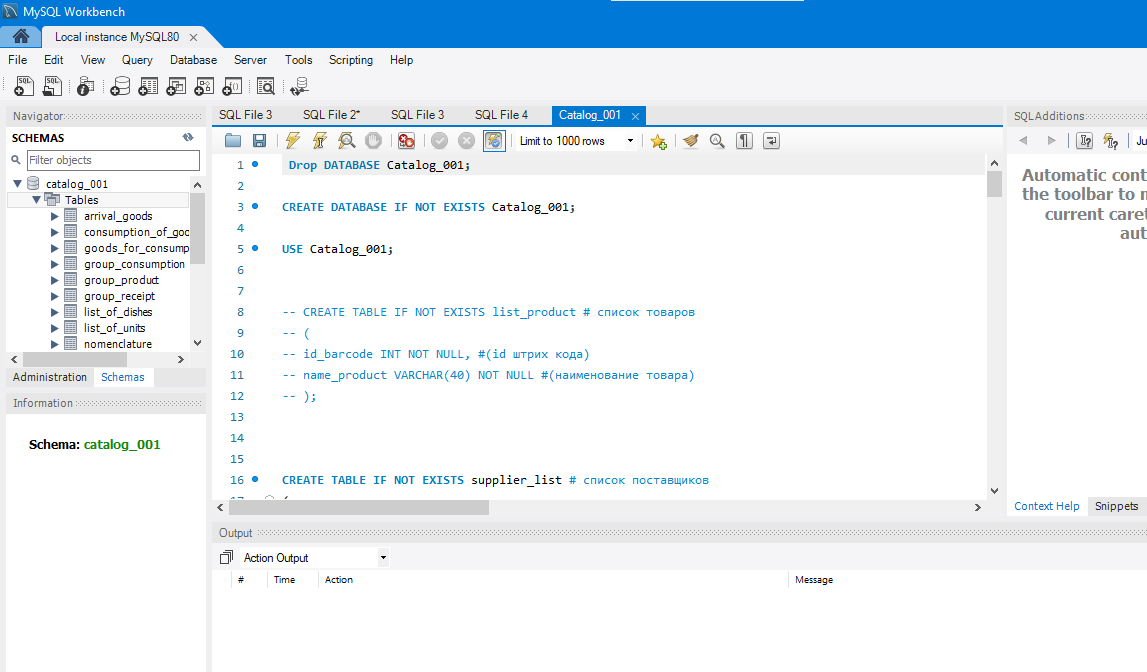
sum\_product FLOAT NOT NULL, #(сумма по товару)

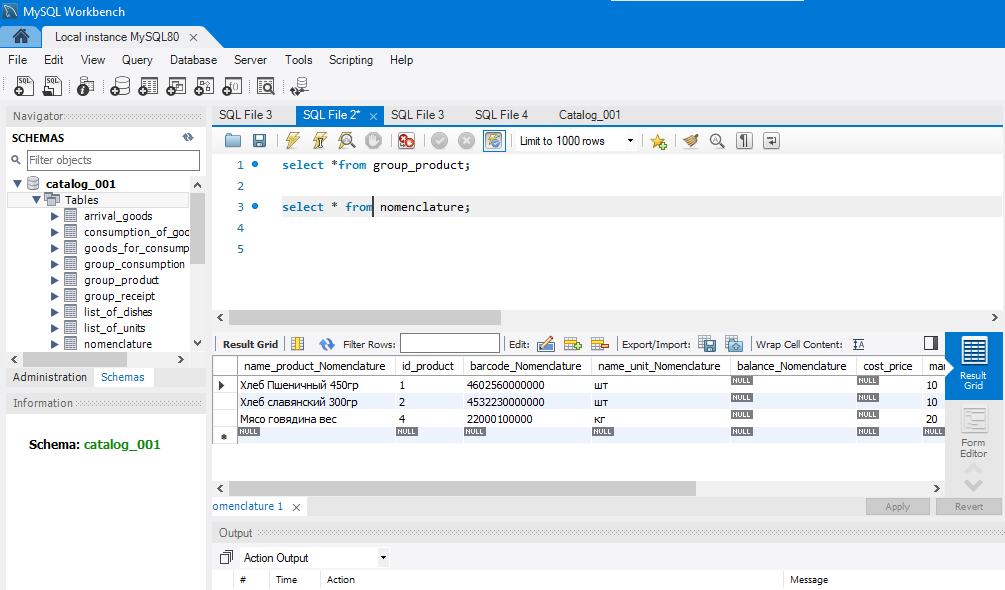
FOREIGN KEY (id\_product\_goods\_for\_consumption) REFERENCES Nomenclature(Id\_product),

FOREIGN KEY (id\_group\_product\_goods\_for\_consumption) REFERENCES group\_product(id\_group\_product),

FOREIGN KEY (id\_unit\_goods\_for\_consumption) REFERENCES list\_of\_units(Id\_unit)

);

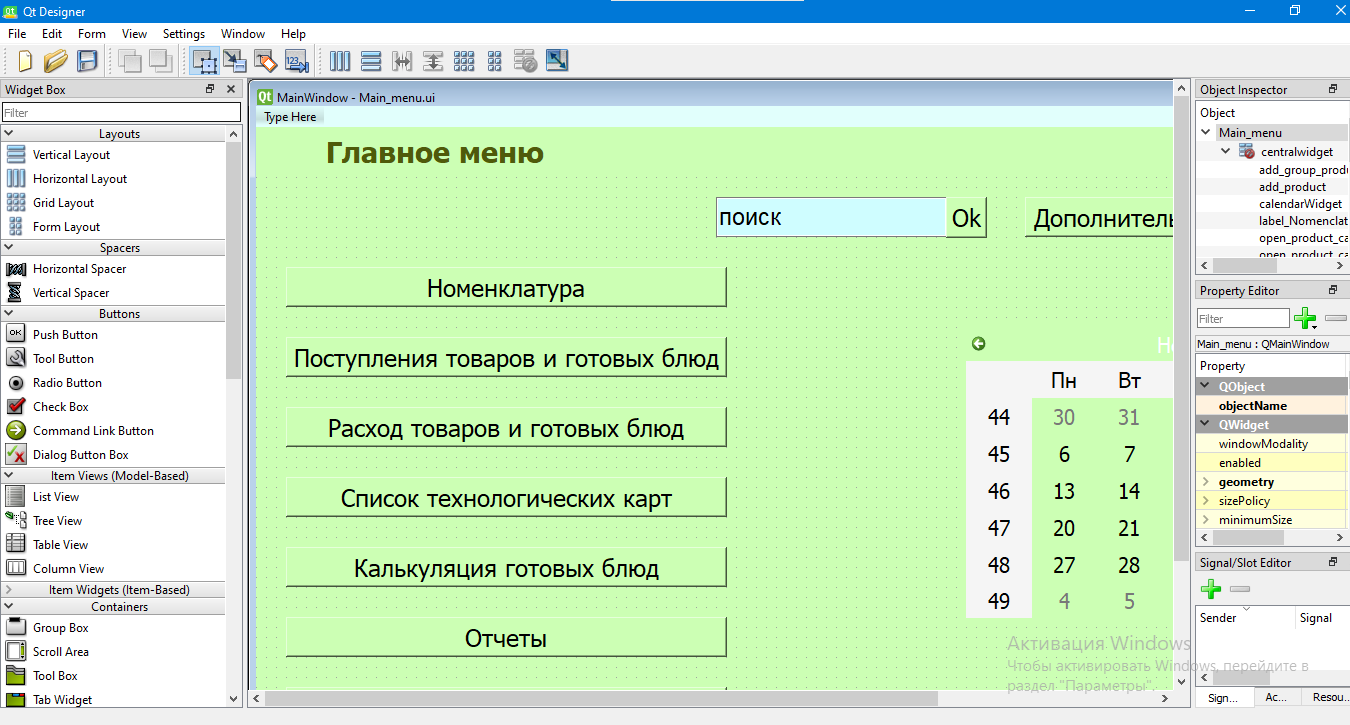


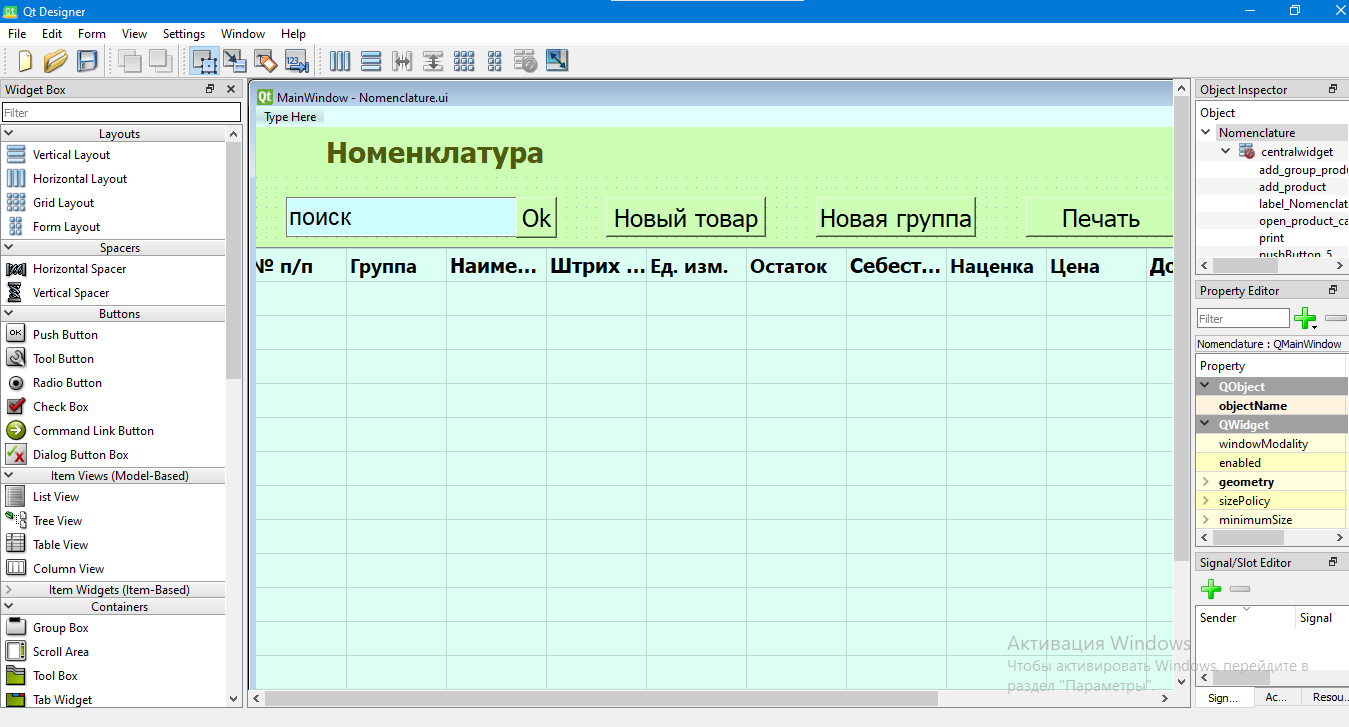


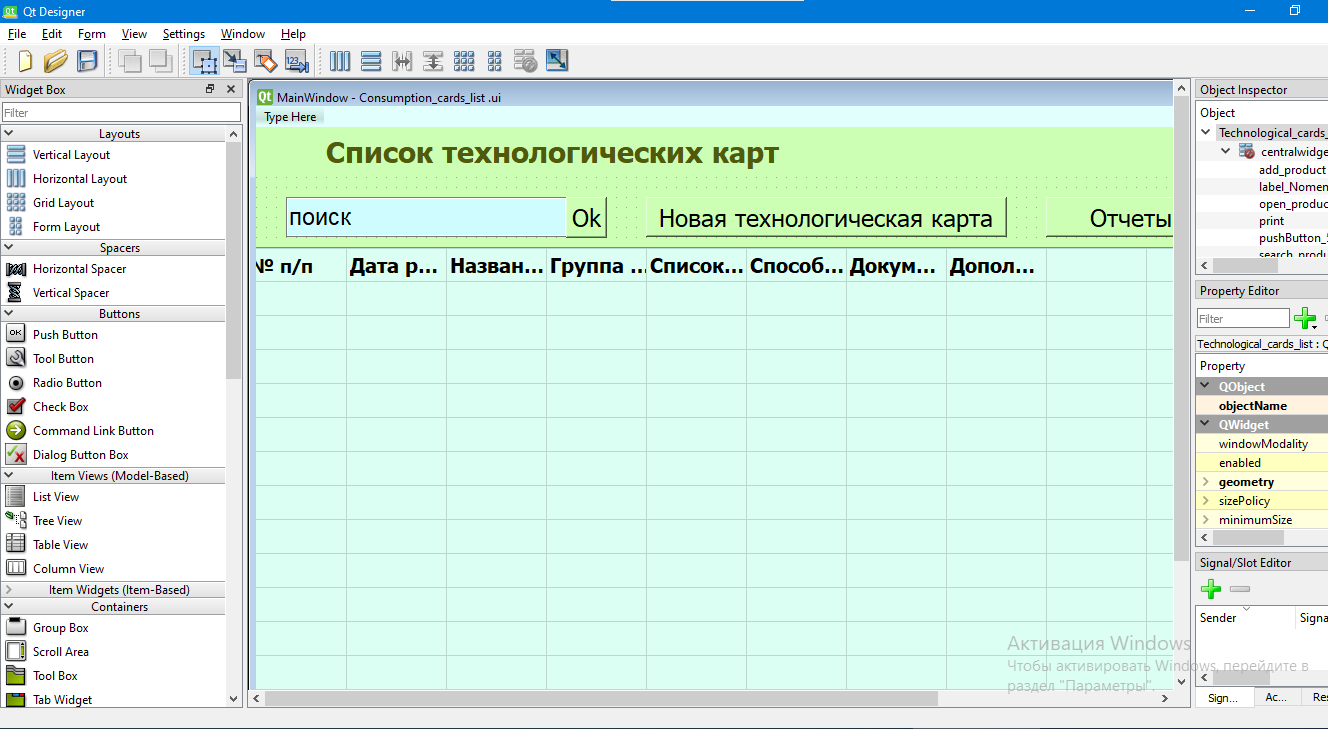
3.2 Создание необходимых для проекта интерфейсы с помощью программы декоратора интерфейсов Qt Designer

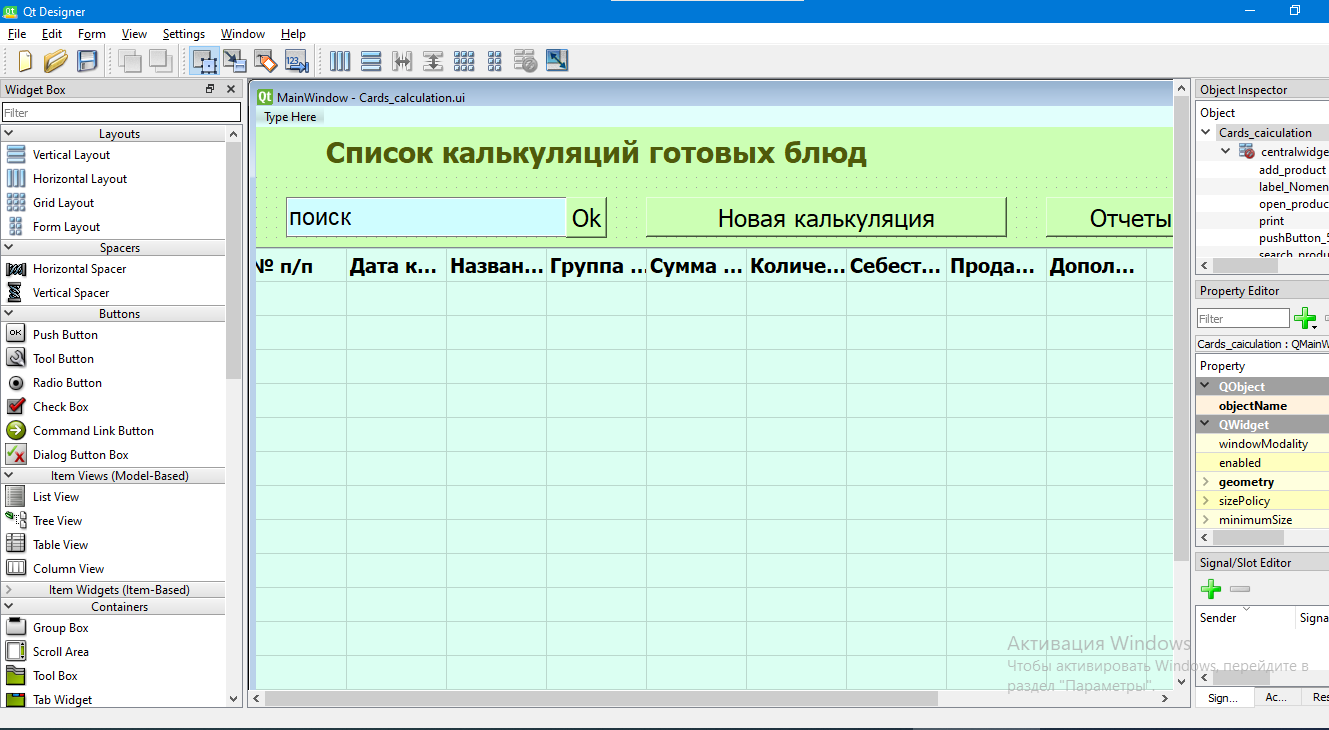
Все, что видит пользователь в приложениях PyQt, называется виджетами. Окна, диалоговые окна, кнопки, сообщения, индикаторы выполнения и так далее — это все виджеты. Они в Qt обычно вложены друг в друга.

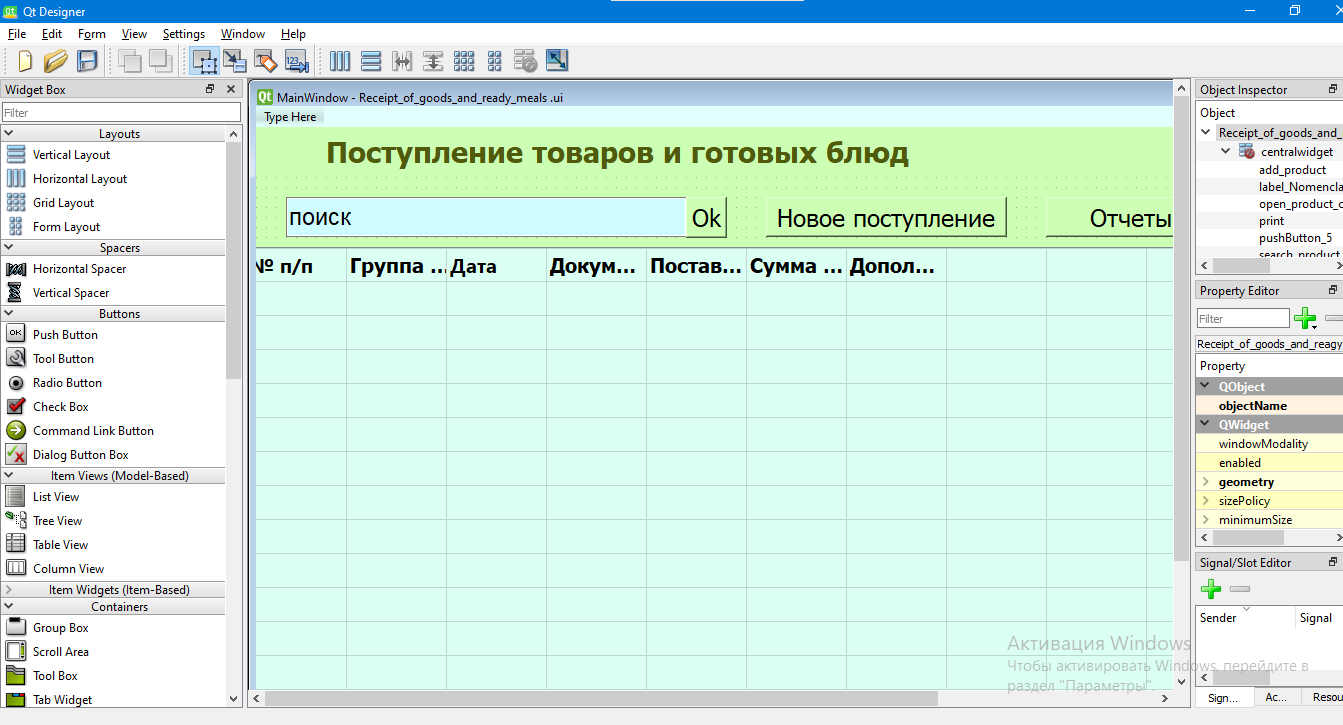
Виджеты, которые я разработала в Qt Designer:

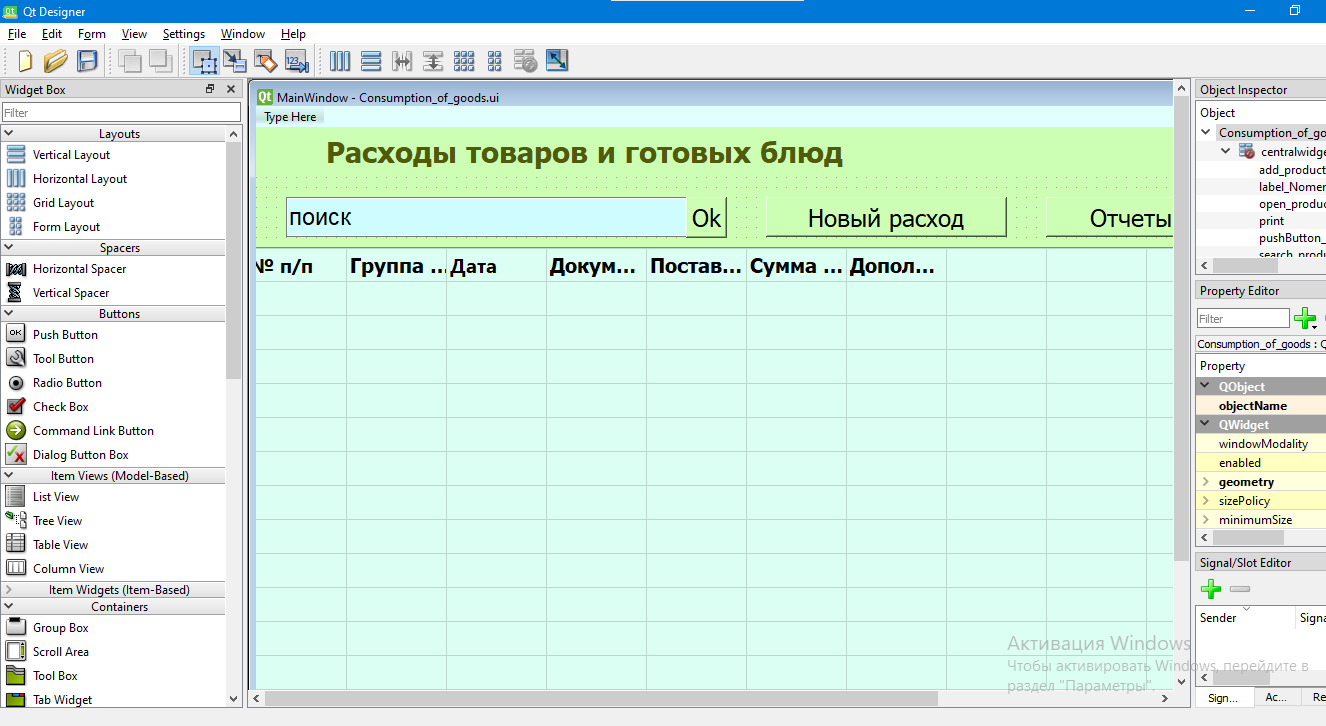




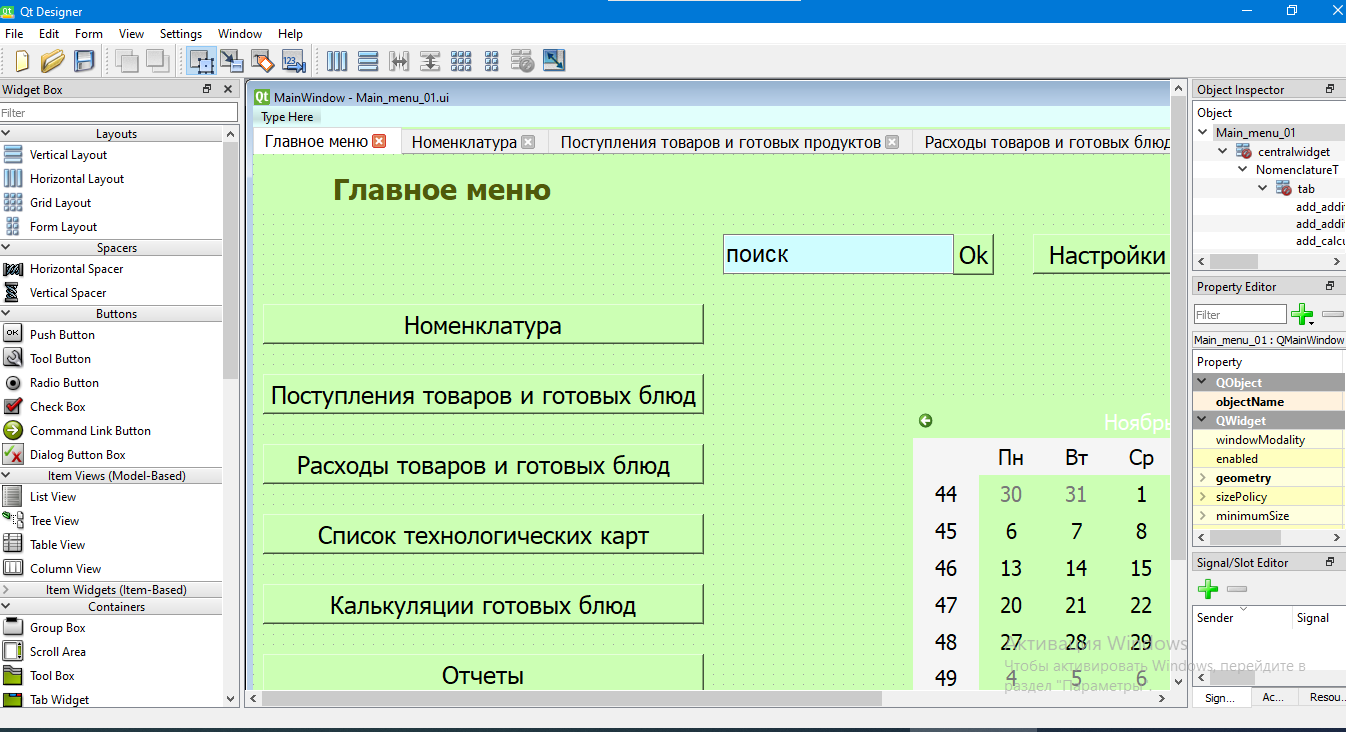




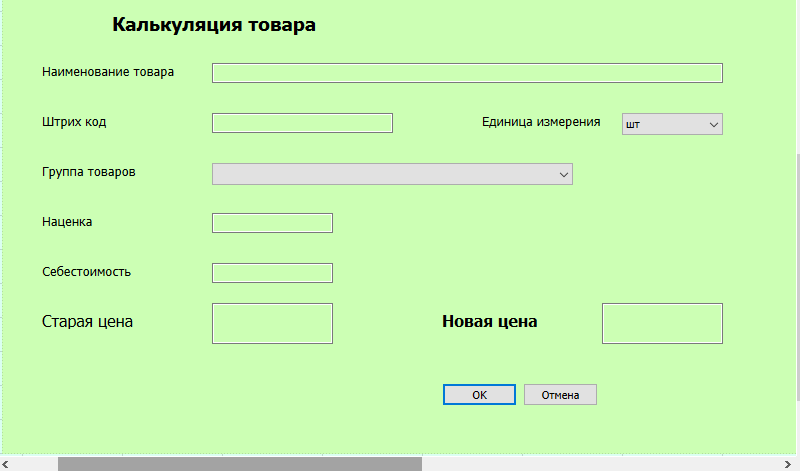


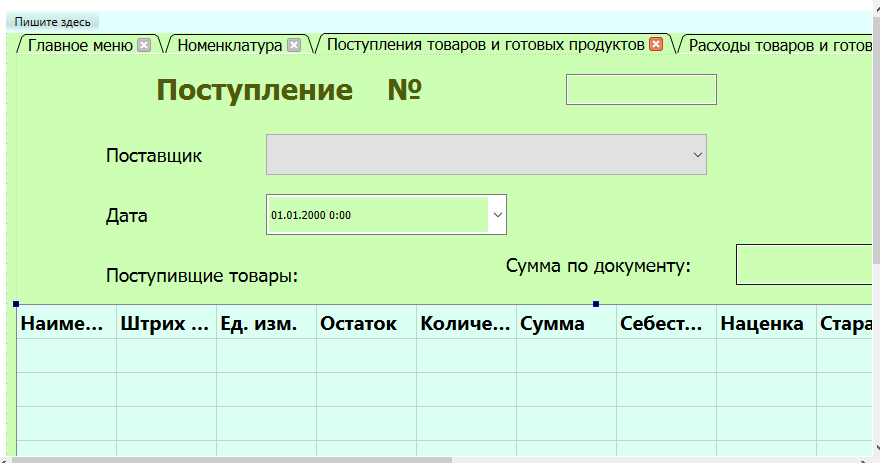


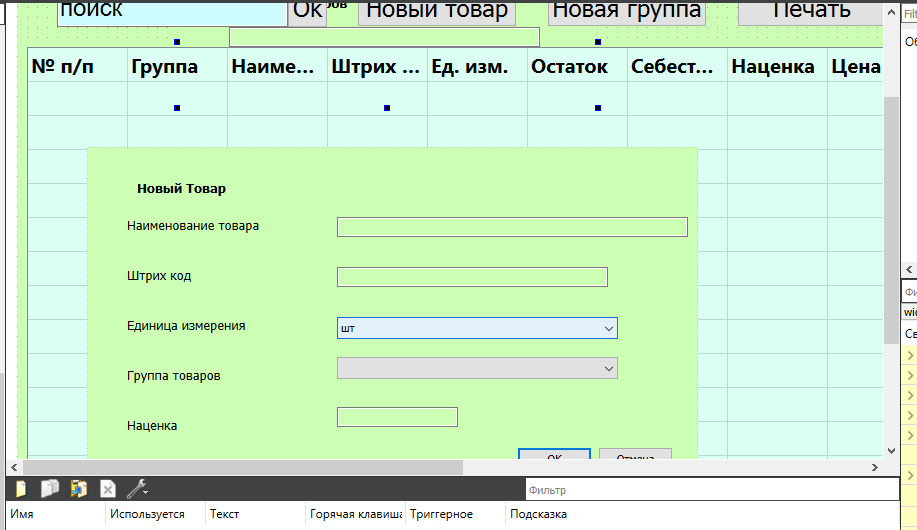
Решила, что их удобнее объединить в один со вкладками:

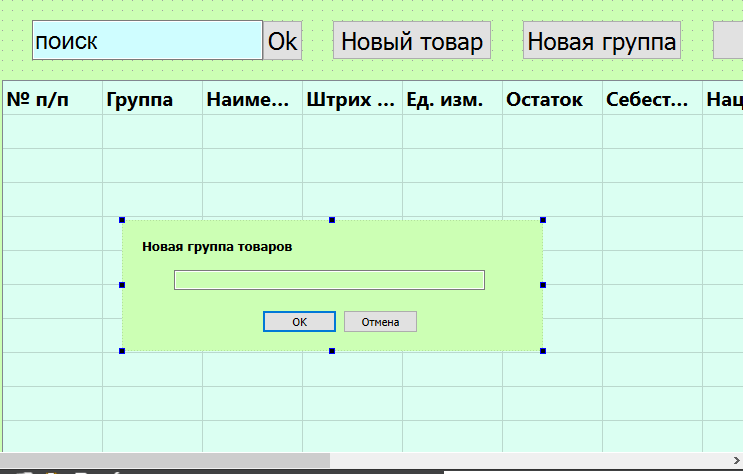


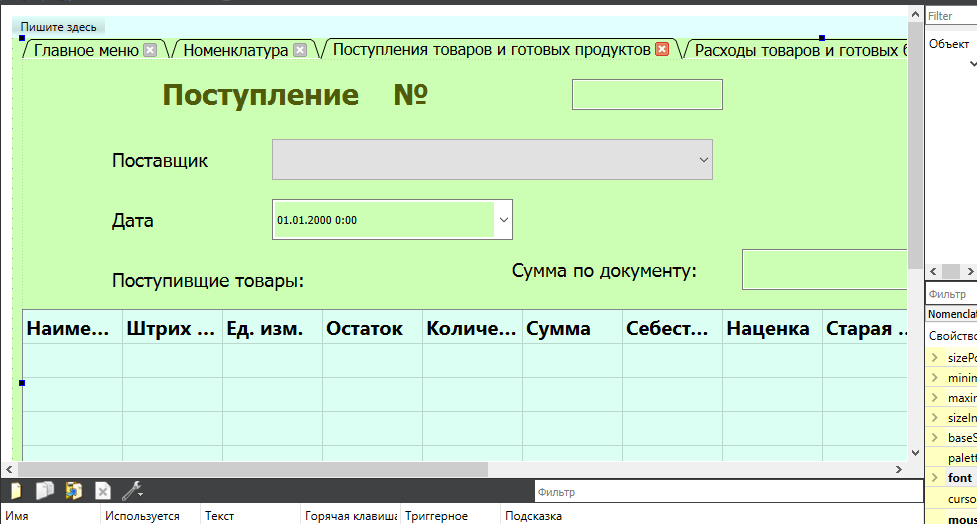
Добавила всплывающие окна:

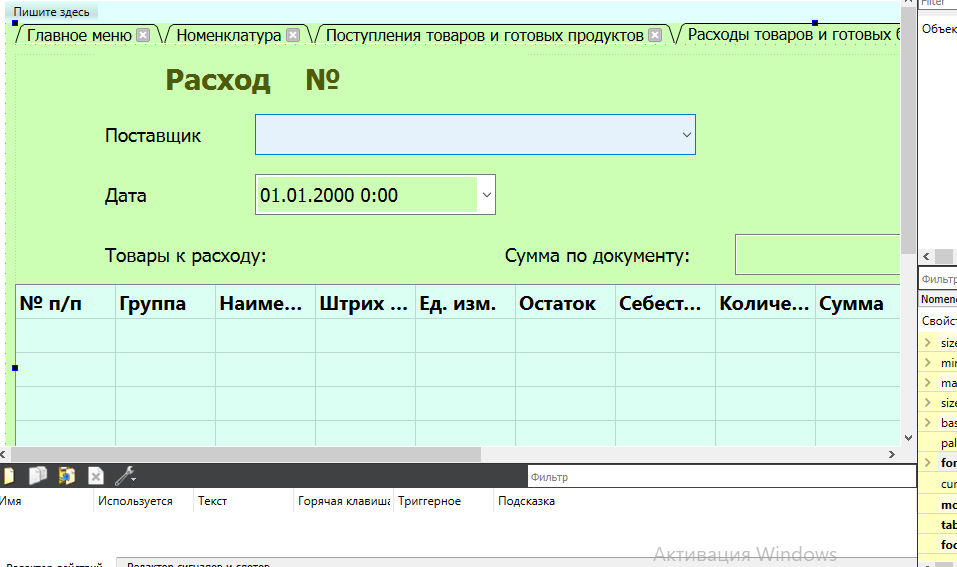


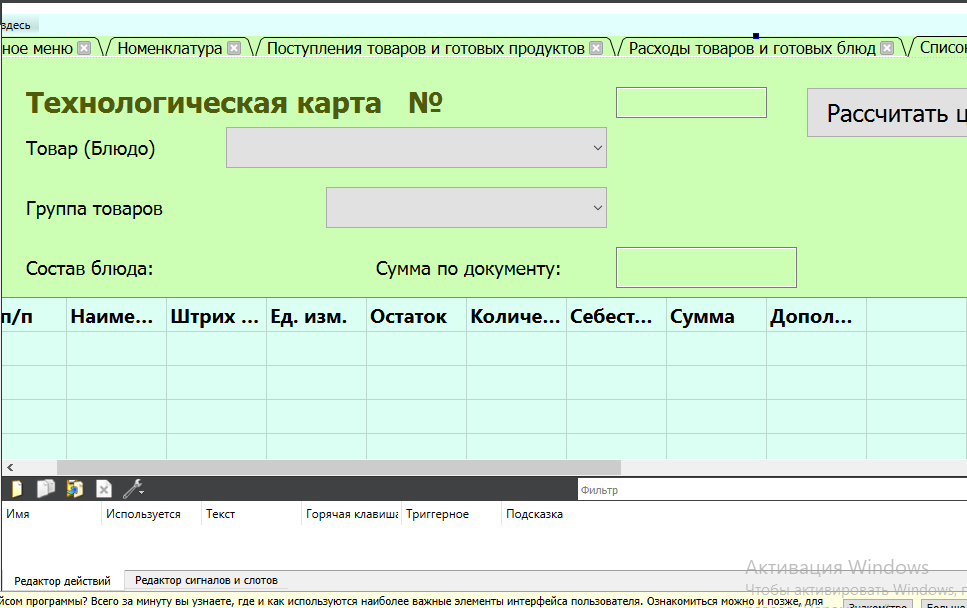


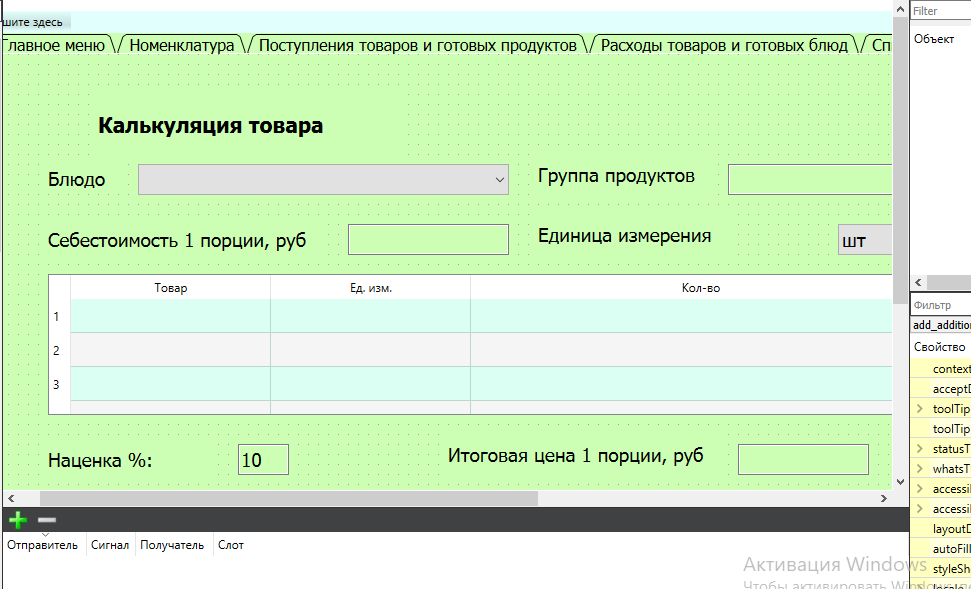


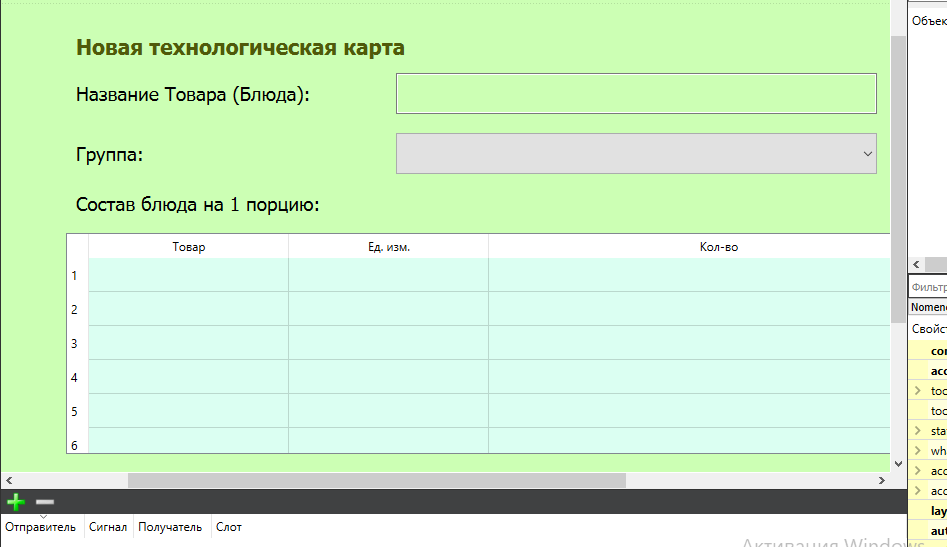


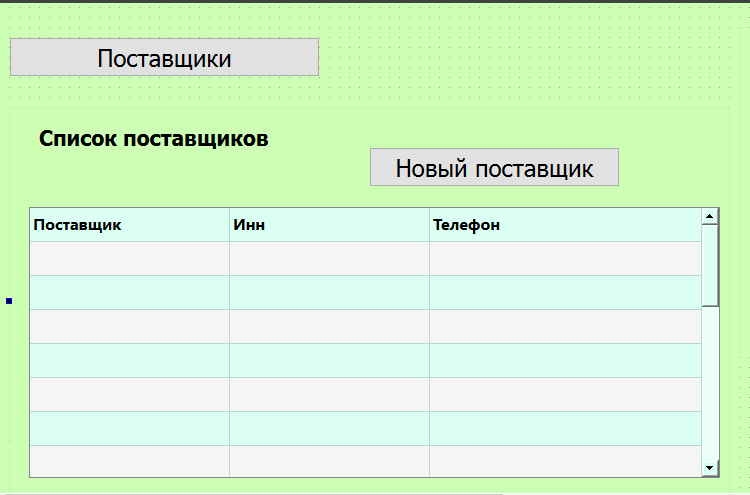


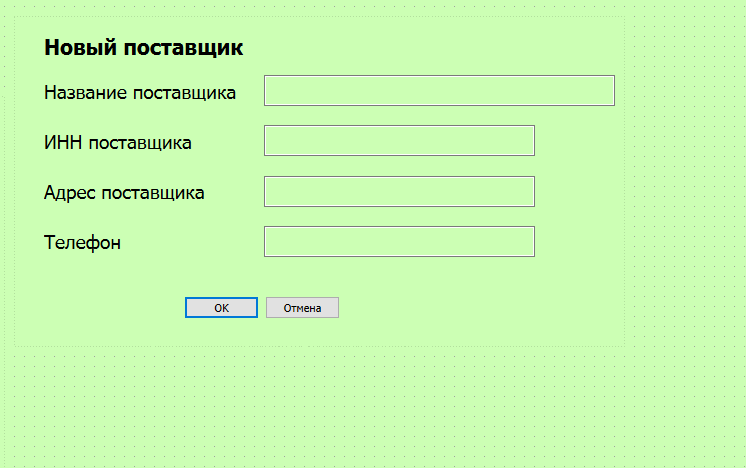












3.3 Написание кода для взаимодействия табличных данных с интерфейсом с помощью фреймворка Qt для языка программирования Python – QtPy5

Для того, чтобы создать виртуальную среду в текущем каталоге выполните следующую команду:

python3 -m venv venv

Для того, чтобы активировать виртуальную среду в операционной системе Windows, нужно выполнить следующую команду:

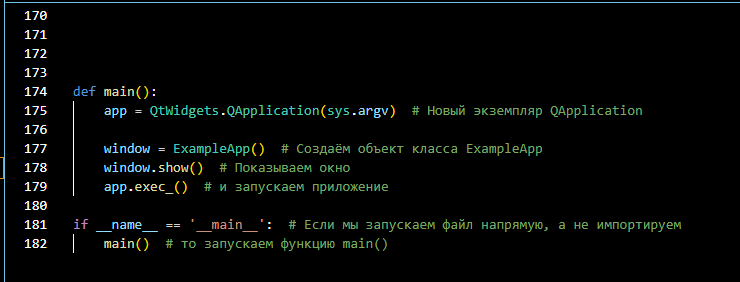
call venv/scripts/activate.bat

Сначала нужно загрузить библиотеку PyQt:

from PyQt5.QtWidgets import QApplication, QLabel

Затем создать QApplication с использование данной команды:

app = QApplication([])



Обязательное условие библиотеки Qt – в приложении с графическим интерфейсом пользователя должен быть ровно один экземпляр класса QApplication. Если данная строка не выполнена, то многие вещи в Qt не могут работать. Поэтому такая команда необходима в каждом Qt.

Метод .show() выводит на экран приложение.

Далее нужно передать управления библиотеке Qt и указать, чтобы приложение запустилось и работало, пока пользователь не захочет его закрыть:

app.exec\_()

Конвертация файла .ui в файл .py при помощи pyuic5:

pyuic5 Main\_menu\_01.ui -o Main\_menu\_01.py

Нужно импортировать этот файл в main.py, чтобы показать его в окне:

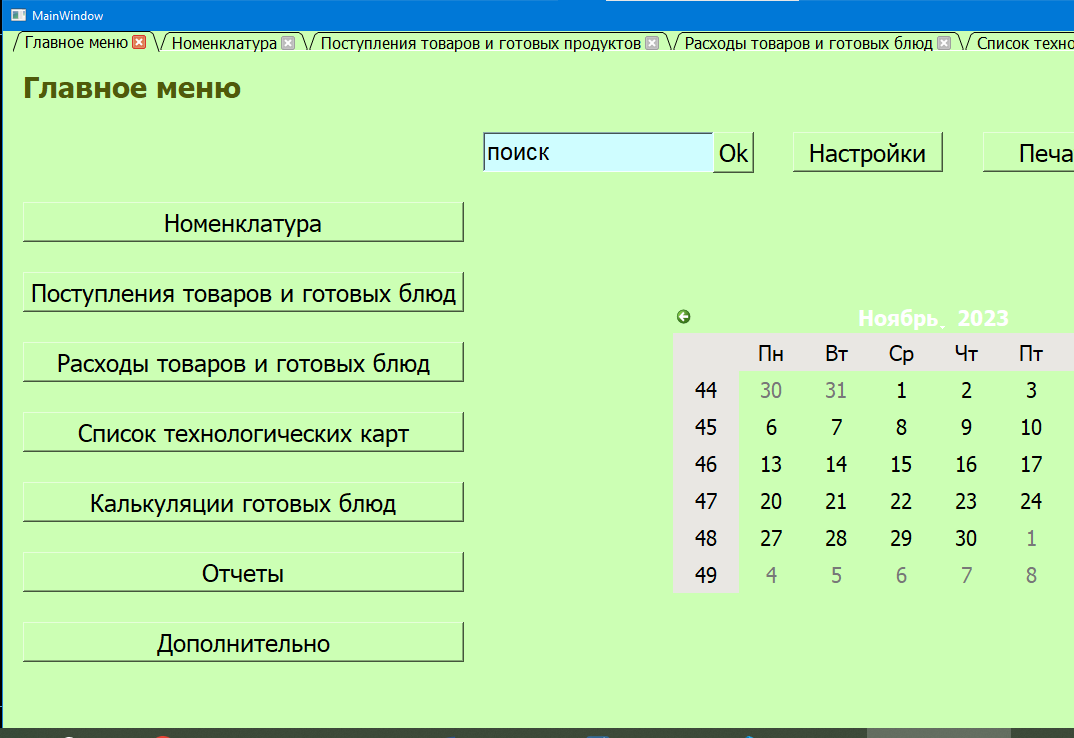
import Main\_menu\_01



Далее переходим к сигналам и слотам – прописываем все объекты.

Делать это нужно в файле main.py, чтобы при изменениях в дизайне - Main\_menu\_01.ui не прописывать все объекты снова.

Кнопки главного меню:



Кнопка «Номенклатура»:

        def sTE():

            self.NomenclatureT.setCurrentWidget(self.Page\_nomenclature)

        self.add\_nomenclature.clicked.connect(sTE)

Кнопка «Поступления товаров и готовых блюд»:

        def sTE1():

            self.NomenclatureT.setCurrentWidget(self.Page\_receipt)

        self.add\_receipt.clicked.connect(sTE1)

Кнопка «Расходы товаров и готовых блюд»:

        def sTE2():

            self.NomenclatureT.setCurrentWidget(self.Page\_consumption)

        self.add\_consumption.clicked.connect(sTE2)

Кнопка «Список технологических карт»:

        def sTE3():

            self.NomenclatureT.setCurrentWidget(self.Page\_technolog)

        self.add\_technological.clicked.connect(sTE3)

Кнопка «Калькуляции готовых блюд»:

        def sTE4():

            self.NomenclatureT.setCurrentWidget(self.Page\_calculation)

        self.add\_calculation.clicked.connect(sTE4)

Кнопка «Отчеты»:

        def sTE5():

            self.NomenclatureT.setCurrentWidget(self.Page\_report)

        self.add\_report.clicked.connect(sTE5)

PyQt содержит средства для работы с базами данных формата SQLite, MySQL, Oracle, PostgreSQL и других., не требующие установкок дополнительных Руthоn-библиотек. А также поддерживается работа с любыми базами данных через платформу ODBC.

Подключение к базе данных, через код  (программно)

        def create\_connection(host\_name, user\_name, user\_password, Catalog\_001):

            connection = None

            try:

               connection = mysql.connector.connect(

                   host=host\_name,

                   user=user\_name,

                   passwd=user\_password,

                   database=Catalog\_001

               )

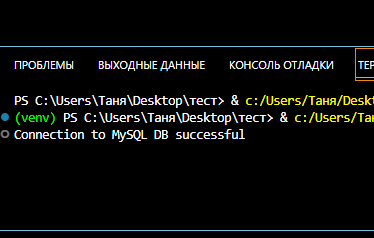
               print("Connection to MySQL DB successful")

            except Error as e:

               print(f"The error '{e}' occurred")

            return connection

        connection = create\_connection("localhost", "root", "пароль", "Catalog\_001")



Функция Заполнение таблицы номенклатуры данными из базы SQL

def set\_nom\_table():

self.tableNomenclature.clear()

self.tableNomenclature.setHorizontalHeaderLabels([

"Наименование товара",

"Номер",

"Штрих-код",

"Еденица измерения",

"Остаток",

"Себестоимость",

"Наценка",

"Цена",

"Номер группы продуктов",

"Дополнительно"

])

with connection.cursor() as cursor:

cursor.execute("""

SELECT \* FROM catalog\_001.nomenclature;

""")

numenclature\_arr = cursor.fetchall()

self.tableNomenclature.setRowCount(len(numenclature\_arr))

for row in range(len(numenclature\_arr)):

for col in range(len(numenclature\_arr[0])):

value = str(numenclature\_arr[row][col])

if value == 'None':

self.tableNomenclature.setItem(row, col, QtWidgets.QTableWidgetItem(""))

else:

self.tableNomenclature.setItem(row, col, QtWidgets.QTableWidgetItem(value))

set\_nom\_table()

# Функция добавления записи Новый товар

def add\_new\_product():

with connection.cursor() as cursor:

cursor.execute("""

INSERT INTO Nomenclature (name\_product\_Nomenclature, barcode\_Nomenclature, name\_unit\_Nomenclature, margin, retai\_price)

VALUES

('"""+self.lineEdit\_new\_prdct\_name.displayText()+"""',

"""+self.lineEdit\_new\_prdct\_cod.displayText()+""",

'"""+self.comboBox\_new\_prdct\_unit.currentText()+"""',

'"""+self.lineEdit\_new\_prdct\_margin.displayText()+"""',

'"""+self.lineEdit\_new\_prdct\_price.displayText()+"""')""")

connection.commit()

self.new\_prdct.hide()

self.buttonBox\_new\_prdct.accepted.connect(add\_new\_product)



Функция добавления записи Новое поступление

self.widget\_receipt.hide()

self.add\_new\_receipt.clicked.connect(self.widget\_receipt.show)

def add\_new\_receipt():

with connection.cursor() as cursor:

cursor.execute("""

INSERT INTO receipt\_of\_goods\_and\_ready\_meals(date\_001,

document\_base, id\_supplier\_Receipt\_of\_goods, sum\_document)

VALUES

('"""+self.dateTimeEdit.displayText()+"""',

"""+self.lineEdit\_lineEdit\_receipt.displayText()+""",

'"""+self.comboBox\_comboBox\_receipt.currentText()+"""',

'"""+self.lineEdit\_lineEdit\_consumption\_5.displayText()+"""')""")

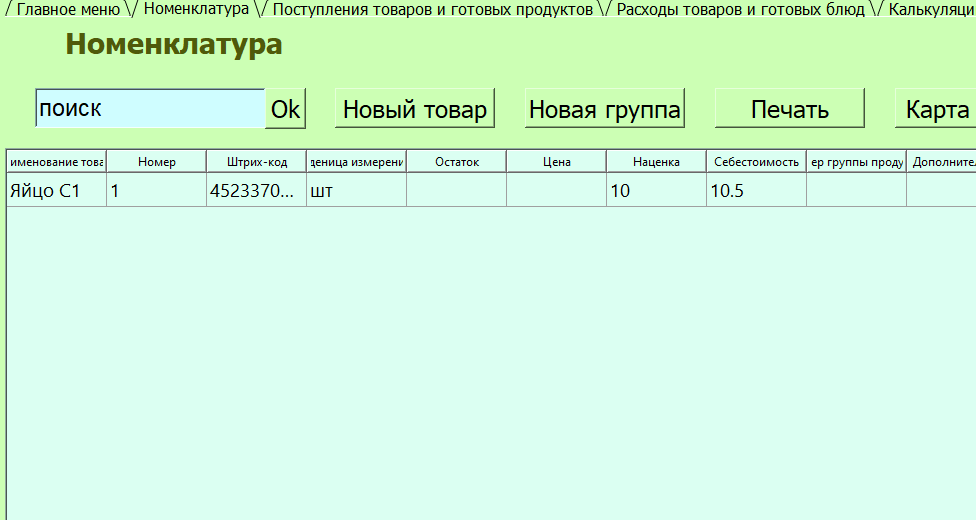
connection.commit()

self.buttonBox\_new\_prdct\_3.hide()

self.buttonBox\_new\_prdct\_3.accepted.connect(add\_new\_receipt)

Отмена добавления записи Новое поступление

self.buttonBox\_new\_prdct\_3.rejected.connect(self.widget\_receipt.show)



Функция выбора группы товаров при создании нового товара

def select\_group\_product\_to\_nomen():

self.comboBox\_new\_prdct\_group.clear()

with connection.cursor() as cursor:

cursor.execute("SELECT name\_group\_product FROM catalog\_001.group\_product;")

array\_group\_product = cursor.fetchall()

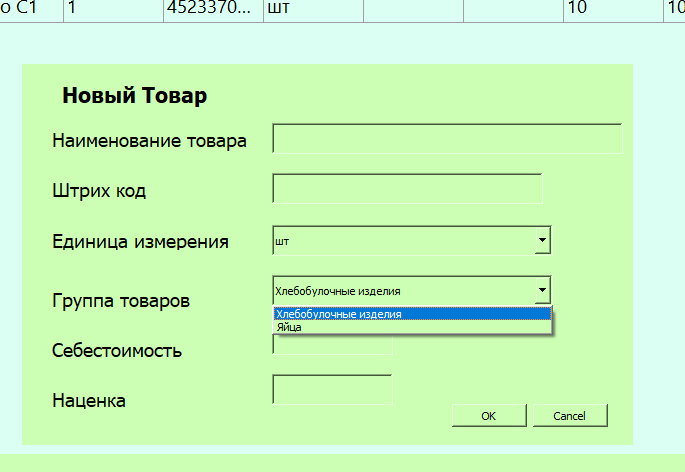
for row in array\_group\_product:

self.comboBox\_new\_prdct\_group.addItem(row[0])

self.new\_prdct.show()

self.new\_prdct.hide()

self.add\_new\_product.clicked.connect(select\_group\_product\_to\_nomen)



Отмена добавления Нового товара

self.buttonBox\_new\_prdct.rejected.connect(self.new\_prdct.hide)

Открыть окно Новая группа товаров

self.widget\_new\_group.hide()

self.add\_group\_product.clicked.connect(self.widget\_new\_group.show)

#Функция добавления записи Новая группа товаров

def add\_new\_group():

with connection.cursor() as cursor:

cursor.execute("""

INSERT INTO group\_product (name\_group\_product)

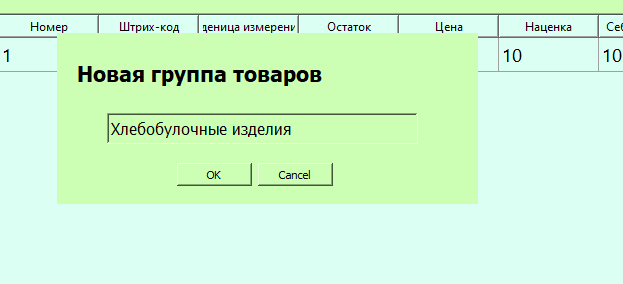
VALUES

('"""+self.lineEdit\_widget\_new\_group.displayText()+"""')""")

connection.commit()

self.widget\_new\_group.hide()

self.buttonBox\_widget\_new\_group.accepted.connect(add\_new\_group)



Отмена добавления Новой группы товаров

self.buttonBox\_widget\_new\_group.rejected.connect(self.widget\_new\_group.hide)

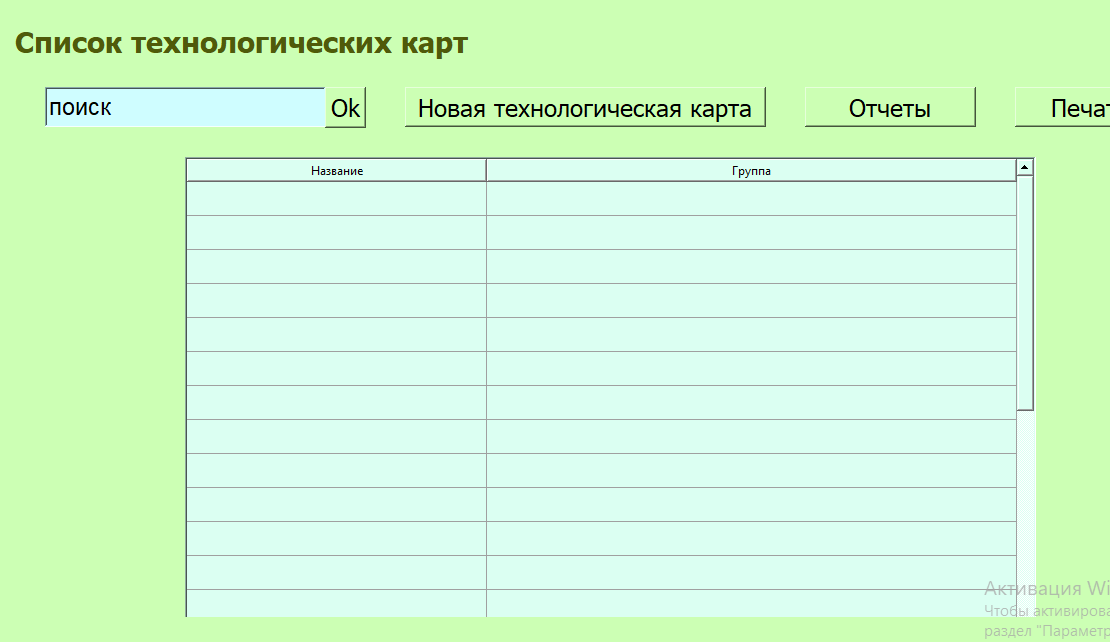
Открытие окна Технологические карты

self.widget\_receipt\_3.hide()

self.buttonBox\_new\_receip.rejected.connect(self.widget\_receipt\_3.hide)

self.table\_new\_receipt.horizontalHeader().setVisible(True)

self.table\_new\_receipt.setColumnCount(3)



Функция добавления записи Новая технологическая карта (рецепт)

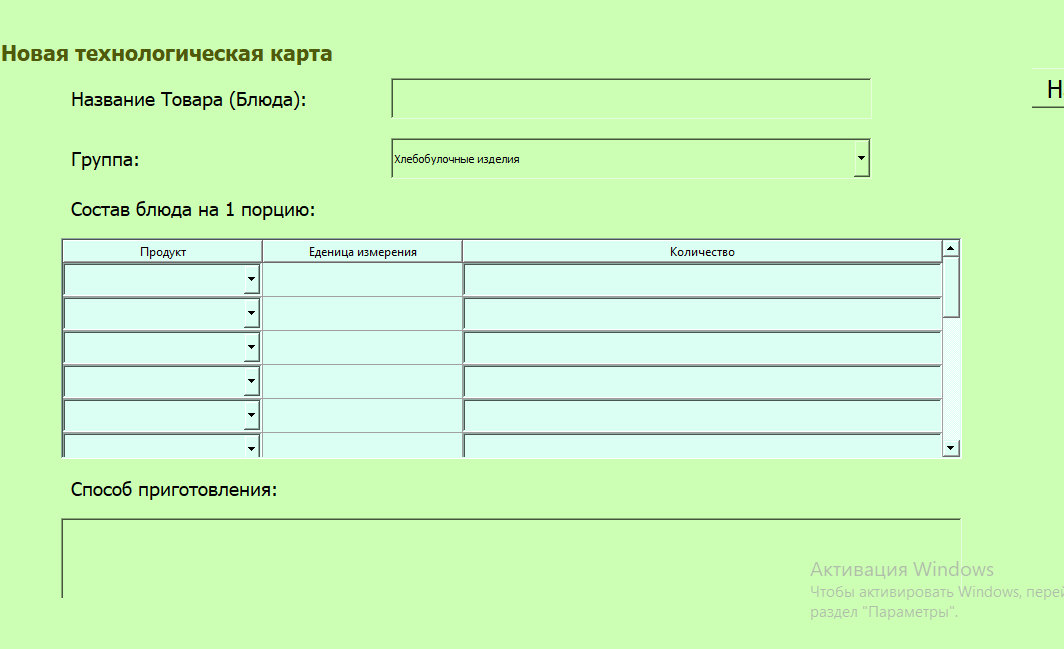
def open\_new\_receip():

# Очищаем таблицу

self.table\_new\_receipt.clear()

self.table\_new\_receipt.setHorizontalHeaderLabels(["Продукт", "Еденица измерения", "Количество"])

self.comboBox\_new\_receip\_group.clear()



# Выгружаем группы продуктов и добовляем в combotBox

with connection.cursor() as cursor:

cursor.execute("SELECT name\_group\_product FROM catalog\_001.group\_product;")

array\_group\_product = cursor.fetchall()

for row in array\_group\_product:

self.comboBox\_new\_receip\_group.addItem(row[0])

# массив с продуктами

array\_product = []

# картеж с продуктами и единицами измерения

tup\_product = []

# выгружаем продукты и единицы измерения

with connection.cursor() as cursor:

cursor.execute("SELECT name\_product\_Nomenclature, name\_unit\_Nomenclature FROM catalog\_001.nomenclature;")

tup\_product = cursor.fetchall()

# вставляем продуты в массив

for row in tup\_product:

array\_product.append(row[0])

# функция для автоматической записи единицы измерения в таблицу во второй столбец

def set\_unit ():

for row in range(self.table\_new\_receipt.rowCount()):

prod = self.table\_new\_receipt.cellWidget(row, 0).currentText()

unit = [itm for itm in tup\_product if itm[0] == prod]

if len(unit) != 0:

self.table\_new\_receipt.setCellWidget(row, 1, QtWidgets.QLabel(str([itm for itm in tup\_product if itm[0] == prod][0][1])))

# всталвяем в 1 столбец combotBox, в 3 - lineEdit

for row in range(self.table\_new\_receipt.rowCount()):

# создаём виджет combotBox

shm = QtWidgets.QComboBox()

shm.addItem("")

# вставляем список продуктов

shm.addItems(array\_product)

# устанавливаем слон на изменение combotBox

shm.currentIndexChanged.connect(set\_unit)

# вставляем в таблицу

self.table\_new\_receipt.setCellWidget(row, 0, shm)

# создаём и вставляем в таблицу lineEdit

self.table\_new\_receipt.setCellWidget(row, 2, QtWidgets.QLineEdit())

self.widget\_receipt\_3.show()

self.add\_new\_technolog.clicked.connect(open\_new\_receip)

Функция добавления технологической карты

def add\_new\_receip ():

with connection.cursor() as cursor:

# получаем имя блюда

name = self.lineEdit\_new\_receip\_name.displayText()

# получаем группу блюда

group = self.comboBox\_new\_receip\_group.currentText()

# начало запроса

qur = """

INSERT INTO list\_of\_dishes (name\_dish, group\_product, sost\_prod, sost\_unit, sost\_count)

VALUES

"""

# проверяем каждую строку таблицы на наличее выбранного продукта и добовляем к запросу

for row in range(self.table\_new\_receipt.rowCount()):

sost\_prod = self.table\_new\_receipt.cellWidget(row, 0).currentText()

if sost\_prod != "":

nrw = """

('"""+name+"', '"+group+"', '"+\

self.table\_new\_receipt.cellWidget(row, 0).currentText()+"', '"+\

self.table\_new\_receipt.cellWidget(row, 1).text()+"', "+\

self.table\_new\_receipt.cellWidget(row, 2).displayText()+"),"

qur += nrw

print(qur[:-1]+";")

cursor.execute(qur[:-1]+";")

connection.commit()

set\_receip\_in\_calc()

self.widget\_receipt\_3.hide()

self.buttonBox\_new\_receip.accepted.connect(add\_new\_receip)

# выгружаем и вставляем в таблицу список технологических карт

self.table\_receip.horizontalHeader().setVisible(True)

self.table\_receip.setColumnCount(2)

self.table\_receip.setHorizontalHeaderLabels(["Название", "Группа"])

with connection.cursor() as cursor:

cursor.execute("""

select DISTINCT name\_dish, group\_product from list\_of\_dishes;

""")

dish\_arr = cursor.fetchall()

for row in range(len(dish\_arr)):

for col in range(len(dish\_arr[0])):

self.table\_receip.setItem(row, col, QtWidgets.QTableWidgetItem(str(dish\_arr[row][col])))

# Функциия выгрузки технологических карт

def set\_receip\_in\_calc ():

self.comboBox\_calc\_dish.clear()

with connection.cursor() as cursor:

cursor.execute("select DISTINCT name\_dish from list\_of\_dishes;")

tup\_dish = cursor.fetchall()

# добавляем в combotBox первое пустое значение

self.comboBox\_calc\_dish.addItem("")

# добавляем в combotBox блюда

list\_dish = []

for row in range(len(tup\_dish)):

list\_dish.append(tup\_dish[row][0])

self.comboBox\_calc\_dish.addItems(list\_dish)

set\_receip\_in\_calc ()

# функция устанвоки группы продуктов

self.lineEdit\_calc\_group.setReadOnly(True)

def set\_dish\_group():

dish = self.comboBox\_calc\_dish.currentText()

self.lineEdit\_calc\_group.clear()

with connection.cursor() as cursor:

cursor.execute("select DISTINCT name\_dish, group\_product from list\_of\_dishes;")

tup\_dish = cursor.fetchall()

self.lineEdit\_calc\_group.setText([itm[1] for itm in tup\_dish if itm[0] == dish][0])

self.table\_calc.horizontalHeader().setVisible(True)

self.table\_calc.setColumnCount(4)

# Функция для расчёта себестоимости блюда

def calc\_price ():

self.lineEdit\_calc\_cost.clear()

self.lineEdit\_calk\_res\_price.clear()

self.table\_calc.clear()

self.table\_calc.setHorizontalHeaderLabels(["Товар", "Кол-во", "Цена за шт.", "Итог"])

self.table\_calc.setRowCount(0)

self.lineEdit\_calc\_group.clear()

dish = self.comboBox\_calc\_dish.currentText()

if dish != "":

query\_sost = """select DISTINCT sost\_prod, sost\_count from list\_of\_dishes

where name\_dish = '"""+dish+"';"

with connection.cursor() as cursor:

cursor.execute(query\_sost)

tup\_sost = cursor.fetchall()

query\_prod = """select name\_product\_Nomenclature, retai\_price from nomenclature

where name\_product\_Nomenclature = '"""

rows\_sost = len(tup\_sost)

for row in range(rows\_sost):

query\_prod += tup\_sost[row][0]+"'"

if row != rows\_sost-1:

query\_prod += " OR name\_product\_Nomenclature = '"

with connection.cursor() as cursor:

cursor.execute(query\_prod)

tup\_prod = cursor.fetchall()

fin\_cost = 0

self.table\_calc.setRowCount(rows\_sost)

for row in range(rows\_sost):

price = [itm[1] for itm in tup\_prod if itm[0] == tup\_sost[row][0]][0]\*tup\_sost[row][1]

fin\_cost += price

self.table\_calc.setItem(row, 0, QtWidgets.QTableWidgetItem(tup\_sost[row][0]))

self.table\_calc.setItem(row, 1, QtWidgets.QTableWidgetItem(str(tup\_sost[row][1])))

self.table\_calc.setItem(row, 2, QtWidgets.QTableWidgetItem(str([itm[1] for itm in tup\_prod if itm[0] == tup\_sost[row][0]][0])))

self.table\_calc.setItem(row, 3, QtWidgets.QTableWidgetItem(str(price)))

self.lineEdit\_calc\_cost.setText(str(fin\_cost))

calc\_res\_price()

set\_dish\_group()

self.comboBox\_calc\_dish.currentIndexChanged.connect(calc\_price)

# Функция расчета итоговой стоимости блюда

self.lineEdit\_calc\_cost.setReadOnly(True)

self.lineEdit\_calk\_res\_price.setReadOnly(True)

def calc\_res\_price():

price = self.lineEdit\_calc\_cost.displayText()

up = self.lineEdit\_calc\_up.displayText()

if re.fullmatch(r'^\d+$', up) and re.fullmatch(r'^\d+$', price):

self.lineEdit\_calk\_res\_price.setText(str(round(int(price) \* ((int(up) / 100) + 1))))

self.lineEdit\_calc\_up.textChanged.connect(calc\_res\_price)

# Функция внесения блюда в номенклатуру

def add\_dish\_in\_nom():

name = self.comboBox\_calc\_dish.currentText()

unit = self.comboBox\_calc\_unit.currentText()

count = self.lineEdit\_calc\_count.displayText()

cost = self.lineEdit\_calc\_cost.displayText()

margin = self.lineEdit\_calc\_up.displayText()

retail = self.lineEdit\_calk\_res\_price.displayText()

if name != "" and\

re.fullmatch(r'^шт$|^кг$', unit) and\

re.fullmatch(r'^\d+$', count) and\

re.fullmatch(r'^\d+$', cost) and\

re.fullmatch(r'^\d+$', margin) and\

re.fullmatch(r'^\d+$', retail):

with connection.cursor() as cursor:

qur = """

INSERT INTO nomenclature (

name\_product\_Nomenclature,

name\_unit\_Nomenclature,

balance\_Nomenclature,

cost\_price,

margin,

retai\_price,

barcode\_Nomenclature)

VALUES

('"""+name+"', '"+unit+"', "+count+", "+cost+", "+margin+", "+retail+", "+str(random.randint(1000000000000, 9999999999999))+")"

cursor.execute(qur)

connection.commit()

set\_nom\_table()

self.comboBox\_calc\_dish.setCurrentIndex(0)

self.pushButton\_calc\_add.clicked.connect(add\_dish\_in\_nom)

**Заключение. Какие доработки ещё можно провести для улучшения приложения.**

Приложение требует ещё очень много доработок. Оно создано как шаблон, по которому можно организовать и автоматизировать работу предприятия общественного питания, но это требует затрат большого количества времени, потому что у меня ещё нет достаточных навыков.

Каждое предприятие имеет свои особенности и требования, под которые можно данное приложение доработать.

На данный момент существует очень много требований со стороны государственных организаций по порядку работы любого предприятия, такие, например, как работа с ЕГАИС и Меркурий. Это приложение также нуждается в подобных доработках.

Главная цель создания данного приложения является показать, чему я научилась, на курсе Аналитик больших данных, начиная обучение с абсолютного нуля.

Используемая литература и сайты:

1. <https://doc.qt.io/qtforpython-5/contents.html>
2. <https://www.riverbankcomputing.com/static/Docs/PyQt5/api/qtwidgets/qtablewidget.html?highlight=qtablew#QTableWidget>
3. Python 3 и PyQt 6. Разработка приложений Н. А. Прохоренок, В. А. Дронов
4. <https://interlogika.ru/blog/tpost/ll29hp0ig1-kalkulyatsiya-blyud-kak-uchitivat-produk>
5. <https://kontur.ru/market/spravka/22625-kak_v_obshhepite_rasschitat_stoimost_blyuda>
6. <https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BB%D1%8C%D0%BA%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F>
7. <https://www.python.org/downloads/>
8. <https://ru.wikipedia.org/wiki/Qt_Designer>
9. <https://habr.com/ru/articles/651093/>
10. <https://pythonworld.ru/gui/pyqt5-firstprograms.html>
11. <https://pythonist.ru/rukovodstvo-po-pyqt5/>
12. <http://python-3.ru/page/pyqt5-create-widgets>