**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых» (ВлГУ)

**РАЗРАБОТКА МНОГОПОЛЬЗОВАТЕЛЬСКОЙ ОНЛАЙН-ИГРЫ**

Пояснительная записка

|  |  |  |
| --- | --- | --- |
| Руководитель |  | к.т.н. доцент кафедры ИЗИ Ю.М. Монахов |
| Исполнитель |  | студент гр. ИБ-122 Задворный А.П. |
|  |  |  |

Владимир 2023

**Оглавление**

[Введение 3](#_Toc154435881)

[Основные характеристики 4](#_Toc154435882)

[Архитектура приложения 5](#_Toc154435883)

[Функциональные требования 6](#_Toc154435884)

[Нефункциональные требования 7](#_Toc154435885)

[Классовая модель 8](#_Toc154435886)

[UML-Диаграммы 9](#_Toc154435887)

[Прототип Приложения 12](#_Toc154435888)

[Разработка Пользовательского интерфейса 14](#_Toc154435889)

[Проверка выполнения требований 23](#_Toc154435890)

[Ссылка на GitHub с проектом 25](#_Toc154435891)

[Использованная Литература 25](#_Toc154435892)

[Завершение 26](#_Toc154435893)

# Введение

Проект "Отображение Цен Валют с Нескольких Бирж" разрабатывается с целью предоставить пользователям удобный и информативный способ отслеживать цены валют на различных биржах. Этот пользовательский интерфейс будет обеспечивать доступ к актуальным данным с нескольких бирж, позволяя пользователям принимать обоснованные решения в сфере торговли криптовалютами и другими финансовыми активами.

# Основные характеристики

**Тип приложения**

**Десктопное приложение**

**Основной функционал**

* Интерфейс будет отображать актуальные цены для выбранных валютных пар с нескольких бирж.
* Пользователи смогут выбирать валютные пары и биржи, для которых они хотят видеть ценовую информацию.
* Цены будут обновляться в реальном времени.
* Запросы с бирж будут приходить синхронно
* Доступен конвертер
* Пользователи смогут выбрать криптовалюту и биржу через которую произойдет перевод
* Перевод в доллары выполнен по самому выгодному курсу
* Информация о переводе отправляется в базу данных и список который может отслеживать пользователь
* Пользователь может удалять записи о переводе

# Архитектура приложения

**Используемые технологии:**

1. Python для основной разработки.
2. API для взаимодействия с биржами.
3. Многопоточное программирование для синхронного отправления запроса.
4. База данных (например, SQLite или PostgreSQL) для хранения истории цен.

**Структура приложения:**

1. Главное окно приложения.
2. Окно конвертора
3. Окно со списком переводов
4. Модуль для работы с базой данных.
5. Модуль для обработки команд и запросов пользователя.

# Функциональные требования

* **Подключение к API бирж:** Приложение должно иметь функциональность для подключения к API различных бирж криптовалют, чтобы получать актуальные данные о ценах.
* **Выбор валютных пар:** Пользователи должны иметь возможность выбирать валютные пары для отображения цен и анализа.
* **Обновление данных в реальном времени:** Цены и данные о торгах должны обновляться в реальном времени, чтобы обеспечивать актуальность информации.
* **Конвертация:** программа будет переводить рубли в выбранную криптовалюту

# Нефункциональные требования

1. **Безопасность данных**

Должны приниматься меры по обеспечению безопасности хранения и передачи пользовательских данных.

1. **Производительность**

Время ответа от биржи должно быть допустимым для комфортного использования. При большом количестве пользователей система должна обеспечивать стабильную производительность.

1. **Доступность**

Система должна быть доступна для пользователей в течение большей части времени.

**Системные требования для клиентского приложения:**

* Операционная система: Windows 10,11, Ubuntu
* Процессор: есть
* Видеокарта: графический процессор совместимый с OpenGL 2 с 128 Мб видеопамяти
* Оперативная память 256 Мб
* Место на жестком диске: 100 Мб

# Классовая модель

1. **CurrencyExchange (Биржа):**

* selected\_currency
* price
* converter

1. **Converter (Конвертация в криптовалюту):**

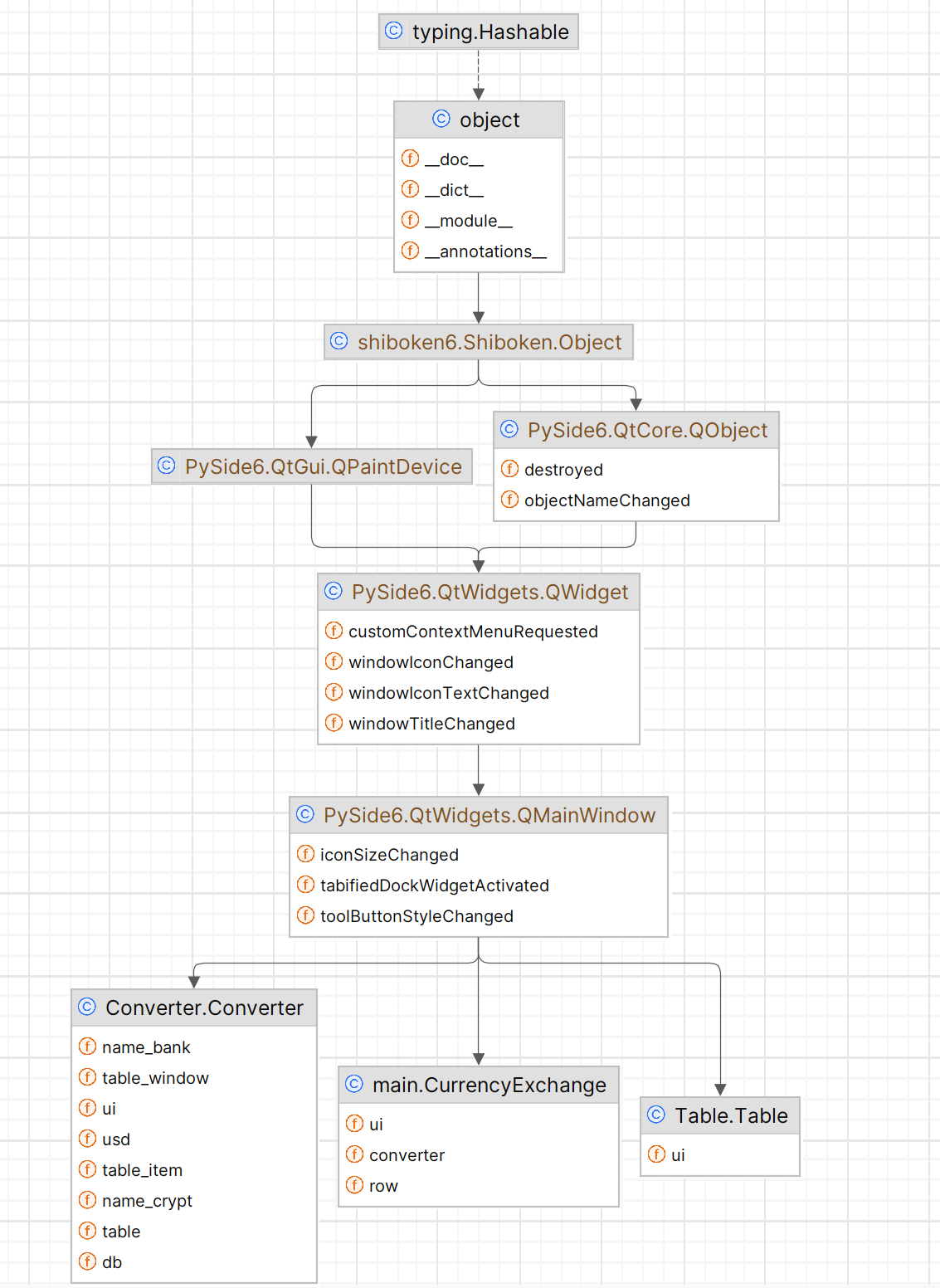
* table
* usd
* name\_bank
* name\_crypt
* db (DataBase)
* datetime

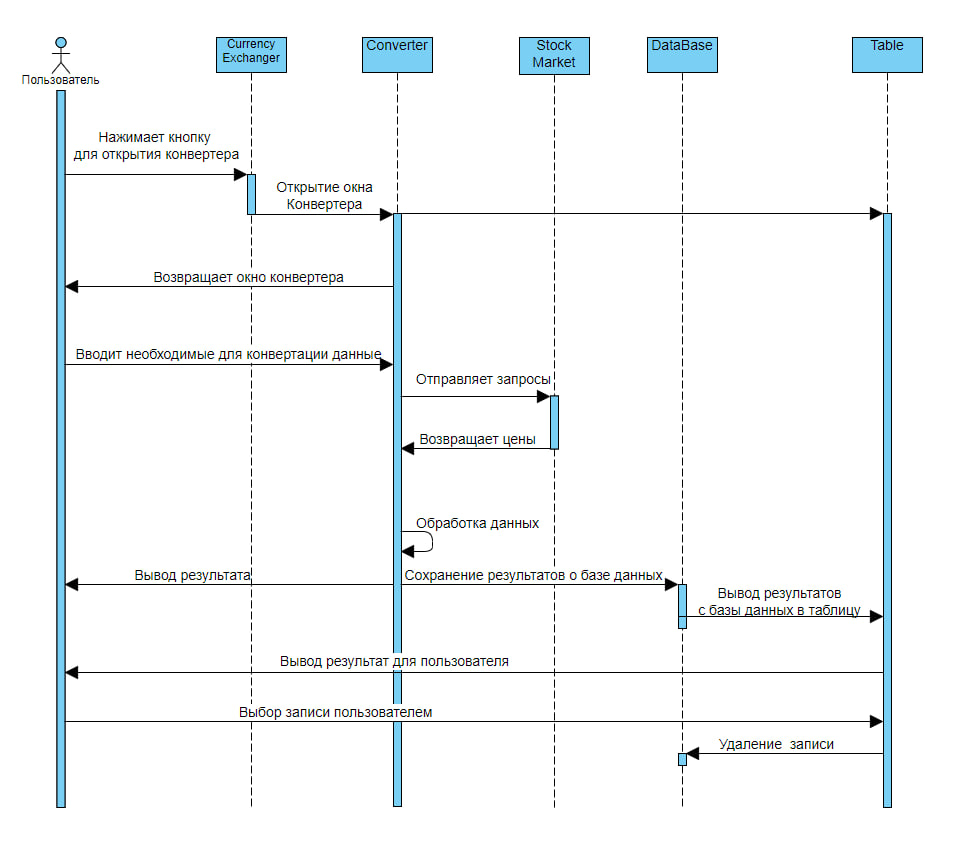
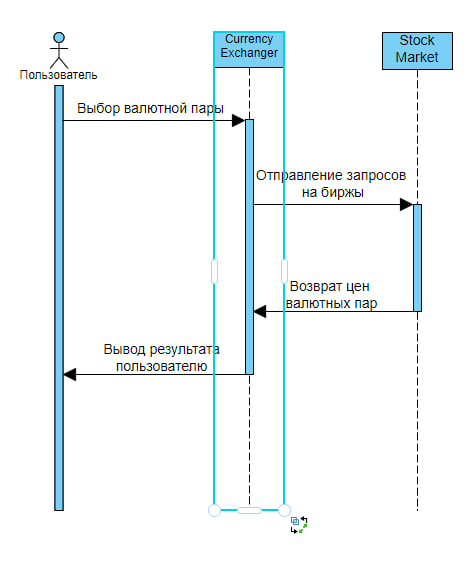
1. **Table (таблица с данными о переводах):**

* listWidgetItem
* selectedItem
* db

# UML-Диаграммы

**Диаграмма классовой модели**



**Диаграмма Последовательности**

# Прототип Приложения

**Код прототипа приложения:**

import requests

from PyQt6 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object): def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow") MainWindow.resize(300, 300)

self.centralwidget = QtWidgets.QWidget(parent=MainWindow) self.centralwidget.setObjectName("centralwidget") self.price = QtWidgets.QLabel(parent=self.centralwidget) self.price.setGeometry(QtCore.QRect(25, 120, 250, 60)) font = QtGui.QFont() font.setFamily("MS Shell Dlg 2") font.setPointSize(12) font.setBold(False) font.setItalic(False) font.setWeight(50) self.price.setFont(font)

self.price.setStyleSheet("background-color: rgb(161, 161, 161);\n"

"font: 12pt \"MS Shell Dlg 2\";\n" "color: rgb(255, 255, 255);") self.price.setText("") self.price.setObjectName("price")

self.send\_request = QtWidgets.QPushButton(parent=self.centralwidget) self.send\_request.setGeometry(QtCore.QRect(75, 210, 150, 50)) self.send\_request.setObjectName("send\_request")

self.currency\_2 = QtWidgets.QLabel(parent=self.centralwidget) self.currency\_2.setGeometry(QtCore.QRect(25, 40, 250, 60)) font = QtGui.QFont() font.setFamily("MS Shell Dlg 2") font.setPointSize(12) font.setBold(False) font.setItalic(False) font.setWeight(50) self.currency\_2.setFont(font)

self.currency\_2.setStyleSheet("background-color: rgb(161, 161,

161);\n"

"font: 12pt \"MS Shell Dlg 2\";\n"

"border-color: rgb(0, 0, 0);\n" "color: rgb(255, 255, 255);") self.currency\_2.setText("")

self.currency\_2.setObjectName("currency\_2") MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow) self.add\_functions()

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "app")) self.send\_request.setToolTip(\_translate("MainWindow", "<html><head/><body><p align=\"center\"><span style=\" fontsize:14pt;\"><br/></span></p></body></html>"))

self.send\_request.setText(\_translate("MainWindow", "Отправить запрос")) def add\_functions(self):

self.send\_request.clicked.connect(lambda: self.parser())

def parser(self):

data =

requests.get('https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT').js on()

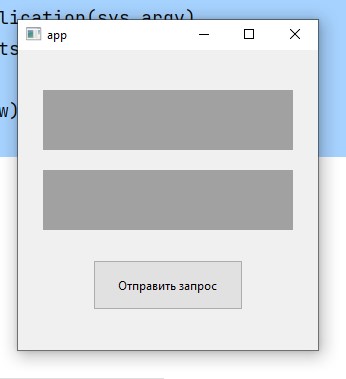
self.price.setText(f"{float(data['price']):.3f}") self.currency\_2.setText(f"{data['symbol']}")

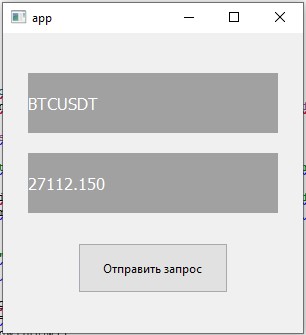
if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv) MainWindow = QtWidgets.QMainWindow() ui = Ui\_MainWindow() ui.setupUi(MainWindow) MainWindow.show()

sys.exit(app.exec())





# Разработка Пользовательского интерфейса

Для разработки пользовательского интерфейса используется библиотека PySide6.

Главное окно включает в себя:

* Таблица в которой отображаются данные с бирж
* Комбо-бокс для выбора валютной пары
* Кнопка для добавления валютной пары в таблицу
* Кнопка для ручного единоразового обновления
* Радио-кнопка для автоматического постоянного обновления
* Кнопка для открытия окна с конвертером

Интерфейс разрабатывался статичным. Изменение размеров окна невозможно. В разработке была соблюдена семантика созданий приложений.  
Был соблюден стандарт PEP8 при написании кода.  
Небольшой отрывок кода с примером реализации  
  
  
from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,  
 QMetaObject, QObject, QPoint, QRect,  
 QSize, QTime, QUrl, Qt)  
from PySide6.QtWidgets import (QApplication, QComboBox, QFrame, QGridLayout,  
 QHBoxLayout, QHeaderView, QLabel, QMainWindow,  
 QPushButton, QRadioButton, QSizePolicy, QSpacerItem,  
 QTableWidget, QTableWidgetItem, QVBoxLayout, QWidget, QCheckBox)  
  
  
class Exchanger\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 if not MainWindow.objectName():  
 MainWindow.setObjectName(u"MainWindow")  
 MainWindow.resize(818, 468)  
 MainWindow.setMinimumSize(QSize(818, 468))  
 MainWindow.setMaximumSize(QSize(818, 468))  
 MainWindow.setStyleSheet(u"background: #233340;\n"  
 "color: #ddedfa;\n"  
 "font-family: Ubuntu;")  
 MainWindow.setUnifiedTitleAndToolBarOnMac(False)  
 self.centralwidget = QWidget(MainWindow)  
 self.centralwidget.setObjectName(u"centralwidget")  
 self.gridLayout = QGridLayout(self.centralwidget)  
 self.gridLayout.setObjectName(u"gridLayout")  
 self.MainFrame = QFrame(self.centralwidget)  
 self.MainFrame.setObjectName(u"MainFrame")  
 self.MainFrame.setMinimumSize(QSize(800, 450))  
 self.MainFrame.setFrameShape(QFrame.StyledPanel)  
 self.MainFrame.setFrameShadow(QFrame.Raised)  
 self.gridLayout\_5 = QGridLayout(self.MainFrame)  
 self.gridLayout\_5.setObjectName(u"gridLayout\_5")  
 self.horizontalLayout\_2 = QHBoxLayout()  
 self.horizontalLayout\_2.setObjectName(u"horizontalLayout\_2")  
 self.ArticleFrame = QFrame(self.MainFrame)  
 self.ArticleFrame.setObjectName(u"ArticleFrame")  
 self.ArticleFrame.setMinimumSize(QSize(550, 0))  
 self.ArticleFrame.setStyleSheet(u"border: 2px solid #ddedfa;\n"  
 "border-radius: 10px;\n"  
 " QTreeView {\n"  
 " background-color: #0078d4; /\* \u0426\u0432\u0435\u0442 \u0444\u043e\u043d\u0430 \u0437\u0430\u0433\u043e\u043b\u043e\u0432\u043a\u0430 \*/\n"  
 " color: #ffffff; /\* \u0426\u0432\u0435\u0442 \u0442\u0435\u043a\u0441\u0442\u0430 \u0437\u0430\u0433\u043e\u043b\u043e\u0432\u043a\u0430 \*/\n"  
 " border: 1px solid #ccc; /\* \u0420\u0430\u043c\u043a\u0430 \u0437\u0430\u0433\u043e\u043b\u043e\u0432\u043a\u0430 \*/\n"  
 " font-size: 14px; /\* \u0420\u0430\u0437\u043c\u0435\u0440 \u0448\u0440\u0438\u0444\u0442\u0430 \u0437\u0430\u0433\u043e\u043b\u043e\u0432\u043a\u0430 \*/\n"  
 "}")  
 self.ArticleFrame.setFrameShape(QFrame.StyledPanel)  
 self.ArticleFrame.setFrameShadow(QFrame.Raised)  
 self.gridLayout\_3 = QGridLayout(self.ArticleFrame)  
 self.gridLayout\_3.setObjectName(u"gridLayout\_3")  
 self.tableWidget = QTableWidget(self.ArticleFrame)

**Окно Конвертера включает себя:**

* Поле для ввода рублей
* Комбо-бокс для выбора криптовалюты
* Комбо-бокс для выбора биржы
* Кнопка для конвертации
* Поле для вывода результата

class Converter\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 if not MainWindow.objectName():  
 MainWindow.setObjectName(u"MainWindow")  
 MainWindow.resize(483, 282)  
 MainWindow.setMinimumSize(QSize(483, 282))  
 MainWindow.setMaximumSize(QSize(483, 282))  
 MainWindow.setStyleSheet(u"background: #233340;\n"  
 "color: #ddedfa;\n"  
 "font-family: Ubuntu;")  
 self.centralwidget = QWidget(MainWindow)  
 self.centralwidget.setObjectName(u"centralwidget")  
 self.gridLayout = QGridLayout(self.centralwidget)  
 self.gridLayout.setObjectName(u"gridLayout")  
 self.frame = QFrame(self.centralwidget)  
 self.frame.setObjectName(u"frame")  
 self.frame.setAcceptDrops(True)  
 self.frame.setStyleSheet(u"border: 2px solid #ddedfa;\n"  
 "color: #ffffff;\n"  
 "font-size: 14px solid #ccc;")  
 self.frame.setInputMethodHints(Qt.ImhNoAutoUppercase)

Окно со Списком транзакций включает:

* QListWidget для отображения списка транзакций
* Кнопка удалить запись

class Table\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 if not MainWindow.objectName():  
 MainWindow.setObjectName(u"MainWindow")  
 MainWindow.resize(800, 400)  
 MainWindow.setMinimumSize(QSize(800, 400))  
 MainWindow.setMaximumSize(QSize(800, 400))  
 MainWindow.setStyleSheet(u"background-color: #233340;\n"  
 "color: #ddedfa;\n"  
 "font-family: Ubuntu;")  
 self.centralwidget = QWidget(MainWindow)  
 self.centralwidget.setObjectName(u"centralwidget")  
 self.gridLayout = QGridLayout(self.centralwidget)  
 self.gridLayout.setObjectName(u"gridLayout")  
 self.frame = QFrame(self.centralwidget)  
 self.frame.setObjectName(u"frame")  
 self.frame.setFrameShape(QFrame.StyledPanel)  
 self.frame.setFrameShadow(QFrame.Raised)  
 self.gridLayout\_2 = QGridLayout(self.frame)  
 self.gridLayout\_2.setObjectName(u"gridLayout\_2")  
 self.horizontalLayout = QHBoxLayout()  
 self.horizontalLayout.setObjectName(u"horizontalLayout")  
 self.delete\_button = QPushButton(self.frame)  
 self.delete\_button.setObjectName(u"delete\_item")  
 self.delete\_button.setMinimumSize(QSize(0, 30))  
 self.delete\_button.setMaximumSize(QSize(400, 100))

# Функционал

При добавление валютной пары отправляются запросы на биржы

**Функции для отправки запроса:**

def request\_binance(self, selected\_currency, row):  
 data\_price = requests.get(currencies\_list[selected\_currency][0]).json()  
 price = f"{float(data\_price['price']):.3f}"  
 self.ui.tableWidget.setItem(row, 0, QTableWidgetItem(price))  
  
def request\_commex(self, selected\_currency, row):  
 data\_price = requests.get(currencies\_list[selected\_currency][1])  
 print(data\_price.headers)  
 price = f"{float(data\_price.json()['price']):.3f}"  
 self.ui.tableWidget.setItem(row, 1, QTableWidgetItem(price))  
  
def request\_mexc(self, selected\_currency, row):  
 data\_price = requests.get(currencies\_list[selected\_currency][2]).json()  
 price = f"{float(data\_price['price']):.3f}"  
 self.ui.tableWidget.setItem(row, 2, QTableWidgetItem(price))  
  
def request\_bybit(self, selected\_currency, row):  
 data\_price = requests.get(currencies\_list[selected\_currency][3]).json()  
 price = f"{float(data\_price['result']['price']):.3f}"  
 self.ui.tableWidget.setItem(row, 3, QTableWidgetItem(price))  
  
def request\_bingx(self, selected\_currency, row):  
 data\_price = requests.get(currencies\_list[selected\_currency][4]).json()  
 price = f"{float(data\_price['data']['price']):.3f}"  
 self.ui.tableWidget.setItem(row, 4, QTableWidgetItem(price))

**Для хранения ссылок для взаимодействия с API создан словарь**currencies\_list = {  
 'BTCUSDT': ['https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT',  
 'https://api.commex.com/api/v1/ticker/price?symbol=BTCUSDT',  
 'https://api.mexc.com/api/v3/ticker/price?symbol=BTCUSDT',  
 'https://api-testnet.bybit.com/spot/v3/public/quote/ticker/price?symbol=BTCUSDT',  
 'https://open-api.bingx.com/openApi/swap/v2/quote/price?symbol=BTC-USDT'  
  
 ],  
 'ETHUSDT': ['https://api.binance.com/api/v3/ticker/price?symbol=ETHUSDT',  
 'https://api.commex.com/api/v1/ticker/price?symbol=ETHUSDT',  
 'https://api.mexc.com/api/v3/ticker/price?symbol=ETHUSDT',  
 'https://api-testnet.bybit.com/spot/v3/public/quote/ticker/price?symbol=ETHUSDT',  
 'https://open-api.bingx.com/openApi/swap/v2/quote/price?symbol=ETH-USDT'  
 ],  
 'LTCUSDT': [  
 'https://api.binance.com/api/v3/ticker/price?symbol=LTCUSDT',  
 'https://api.commex.com/api/v1/ticker/price?symbol=LTCUSDT',  
 'https://api.mexc.com/api/v3/ticker/price?symbol=LTCUSDT',  
 'https://api-testnet.bybit.com/spot/v3/public/quote/ticker/price?symbol=LTCUSDT',  
 'https://open-api.bingx.com/openApi/swap/v2/quote/price?symbol=LTC-USDT'  
 ]  
}

Данные функции добавляются в ThreadPool который отправляет одновременно запрос на все биржы. Полученный результат обрабатывается и добавляется в таблицу.

def add\_currency\_pair(self):  
 if self.row <= 10:  
 selected\_currency = self.select\_currency()  
 if selected\_currency == "":  
 QMessageBox.critical(self, "Error", "Please, Choose a Currency")  
 else:  
 self.ui.tableWidget.verticalHeaderItem(self.row).setText(selected\_currency)  
 *# multithreading----------------------------------* start\_time = time.time()  
 with concurrent.futures.ThreadPoolExecutor() as executor:  
 results = [  
 executor.submit(self.request\_binance, selected\_currency, self.row),  
 executor.submit(self.request\_mexc, selected\_currency, self.row),  
 executor.submit(self.request\_bybit, selected\_currency, self.row),  
 executor.submit(self.request\_commex, selected\_currency, self.row),  
 executor.submit(self.request\_bingx, selected\_currency, self.row)  
 ]  
  
 concurrent.futures.wait(results)  
 end\_time = time.time()  
  
 print(f'Time: {end\_time - start\_time}')  
 self.row += 1

Функция для обновления цен работает следующим образом.

Создается цикл который проходится по всем строкам таблицы и обновляет значения, заново создавая ThreadPool. Максимальное время обновления составляет 3 секунды (при стабильном интернет-соединение)

def update(self):  
 if self.row < 1:  
 QMessageBox.critical(self, 'Error', 'Please, add currencies!')  
 else:  
 for i in range(self.row):  
 key = self.ui.tableWidget.verticalHeaderItem(i).text()  
 with concurrent.futures.ThreadPoolExecutor() as executor:  
 results = [  
 executor.submit(self.request\_binance, key, i),  
 executor.submit(self.request\_mexc, key, i),  
 executor.submit(self.request\_bybit, key, i),  
 executor.submit(self.request\_commex, key, i)  
 ]  
 concurrent.futures.wait(results)

Автоматическое обновление реализовано по подобию ручного.  
Эта функция добавлена в цикл который работает с периодичностью в 3 секунды

def auto\_update(self):  
 if self.row < 1:  
 QMessageBox.critical(self, 'Error', 'Please, add currencies!')  
 return ''  
 while self.ui.auto\_update\_button.isChecked():  
 self.update()  
 time.sleep(2)

**Функционал Конвертера**

После ввода суммы в рублях отправляется запрос для получения цены доллара. Цена парсится с сайта с ценами в разных банках и выбирается лучшая цена.

def parse\_usd\_price(self):  
 url = 'https://www.sravni.ru/valjuty/cb-rf/usd/'  
 response = requests.get(url)  
 if response.status\_code == 200:  
 soup = BeautifulSoup(response.text, 'html.parser')  
 tags = soup.find\_all('div', class\_='styles\_value-text\_\_JklK4 \_5gmjom')  
 value = tags[1].text  
 new\_val = value.replace(',', '.')  
  
 self.name\_bank = tags[1].parent.find\_next('a').text  
  
 usd\_price = float(new\_val[0:5])  
 return usd\_price

Далее с выбранными данными парсится биржа

def request\_exchange(self):  
 select\_exchange = self.select\_exchange().lower()  
 select\_crypt = self.select\_currency().upper()  
 for value in currencies\_list[select\_crypt]:  
 if select\_exchange in value:  
 data\_price = requests.get(  
 currencies\_list[select\_crypt][currencies\_list[select\_crypt].index(value)]).json()  
 if select\_exchange == "bybit":  
 price\_crypt = f"{float(data\_price['result']['price']):.3f}"  
 return float(price\_crypt)  
 if select\_exchange == 'bingx':  
 price\_crypt = f"{float(data\_price['data']['price']):.3f}"  
 return float(price\_crypt)  
 else:  
 price\_crypt = f"{float(data\_price['price']):.3f}"  
 return float(price\_crypt)

Обработка и вывод информации происходит следующим образом

def convert\_to\_crypt(self):  
 rub = self.get\_rub()  
 usd\_price = self.parse\_usd\_price()  
 name\_crypt = self.request\_exchange()  
 usd = rub / usd\_price  
 result = usd / name\_crypt  
 date\_time = self.get\_datetime()  
 self.record\_to\_db(date\_time, rub, usd\_price, result)  
 self.table\_item = self.get\_result(date\_time, rub, usd\_price, result)  
 self.set\_lable\_text(self.name\_bank)  
 self.table.add\_item(self.table\_item)  
 return result

Также полученный результат отправляется в базу данных с последующим отображением в списке конвертаций

def record\_to\_db(self, datetime, rub, usd, crypt):  
 with sqlite3.connect('database.sqlite') as db:  
 cursor = db.cursor()  
 cursor.execute(f"SELECT datetime FROM conversion WHERE datetime = '{datetime}'")  
 if cursor.fetchone() is None:  
 cursor.execute(f"INSERT INTO conversion VALUES(?,?,?,?,?)",  
 (datetime, rub, usd, crypt, self.select\_currency()))  
 db.commit()  
 print('Запись сделана!')  
 else:  
 print("Error!")

**Функционал Списка конвертаций**

Добавления объекта происходит после записи данных о конвертации в базу данных

def add\_item(self, text):  
 item = QListWidgetItem(text)  
 self.ui.listWidget.addItem(item)

def record\_to\_db(self, datetime, rub, usd, crypt):  
 with sqlite3.connect('database.sqlite') as db:  
 cursor = db.cursor()  
 cursor.execute(f"SELECT datetime FROM conversion WHERE datetime = '{datetime}'")  
 if cursor.fetchone() is None:  
 cursor.execute(f"INSERT INTO conversion VALUES(?,?,?,?,?)",  
 (datetime, rub, usd, crypt, self.select\_currency()))  
 db.commit()  
 print('Запись сделана!')  
 else:  
 print("Error!")

Удаление объекта происходит следующим образом, мы удаляем запись из базы данных а затем из QListWidget.

def delete\_item(self):  
 selected\_items = self.ui.listWidget.selectedItems()  
 if len(selected\_items) > 0:  
 selected\_item = selected\_items[0]  
 row = self.ui.listWidget.row(selected\_item)  
 self.ui.listWidget.takeItem(row)  
 self.ui.listWidget.repaint()  
  
def delete\_from\_db(self):  
 item = self.ui.listWidget.currentItem().text()  
 date = item[0:16]  
 if item:  
 import sqlite3  
 with sqlite3.connect('database.sqlite') as db:  
 cursor = db.cursor()  
 cursor.execute('DELETE FROM conversion WHERE datetime = ?',(date,))  
 print('Запись удалена!')

Отображение всех записей с баз данных идет за счет повторного добавления всех элементов в QListWidget.

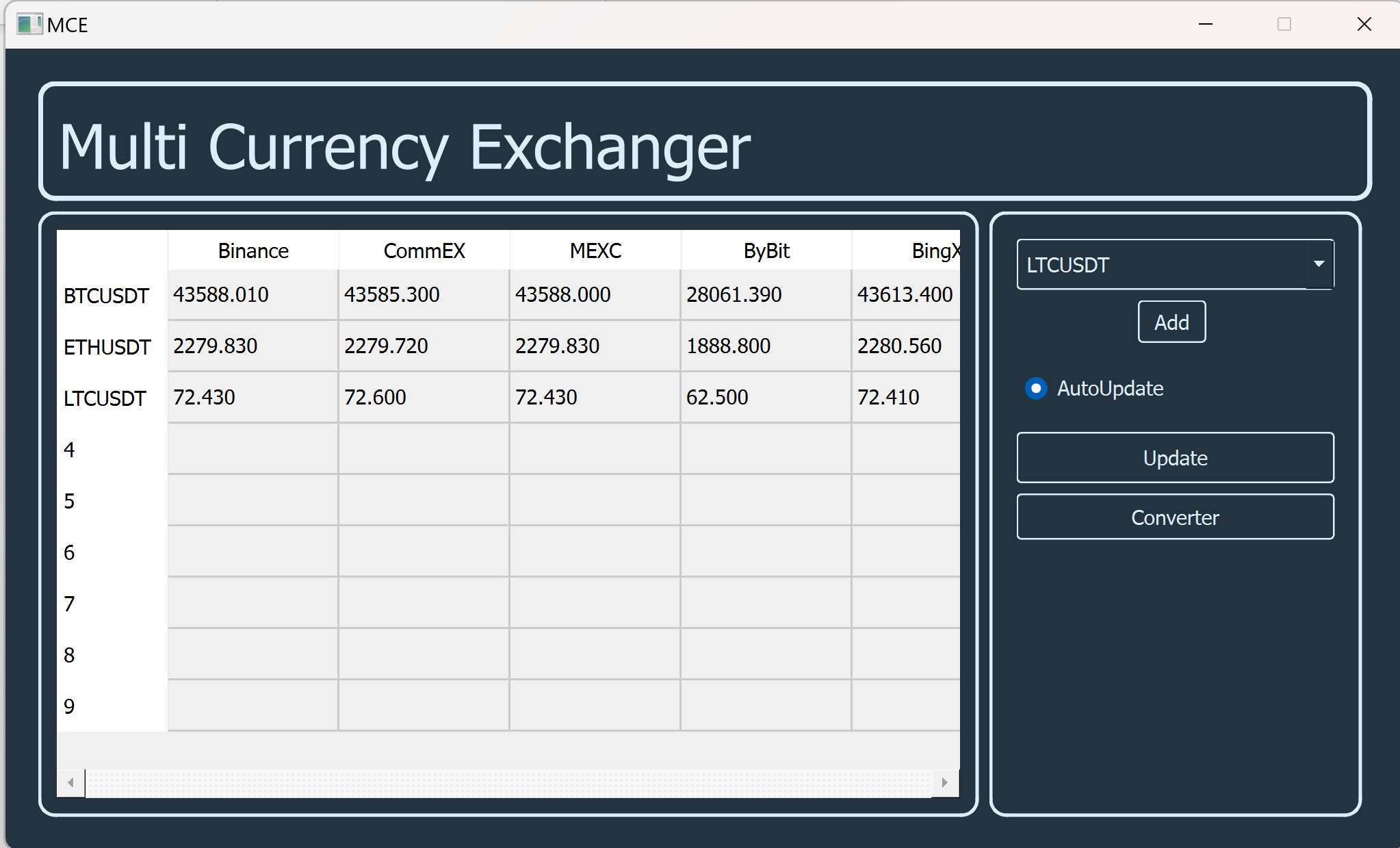
def update\_list(self):  
 import sqlite3  
 with sqlite3.connect('database.sqlite') as db:  
 cursor = db.cursor()  
 cursor.execute('SELECT \* FROM conversion')  
 items = cursor.fetchall()  
 for item in items:  
 string = f"{item[0]} {item[1]} rub -> {item[2]} usd -> {float(item[3]):.5f}{item[4]}"  
 self.ui.listWidget.addItem(str(string))

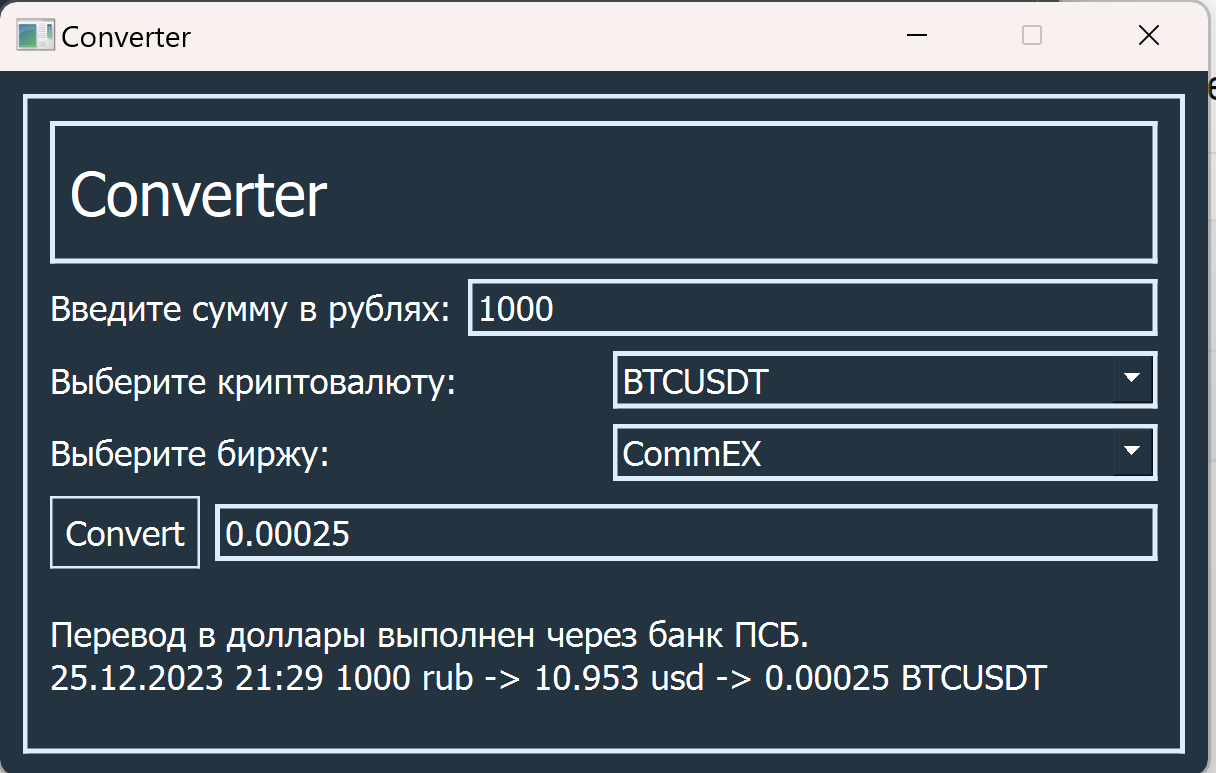
Реализация базы данных выполнена с помощью SQLite3. База данных локальна и хранится непосредственно на устройстве пользователя.

import sqlite3  
  
  
class DataBase:  
 def start\_database(self):  
 with sqlite3.connect('database.sqlite') as db:  
 cursor = db.cursor()  
 cursor.execute("""CREATE TABLE IF NOT EXISTS conversion(  
 datetime DATETIME,  
 rub INT,  
 usd INT,  
 crypt INT,  
 name\_crypt TEXT  
 )""")

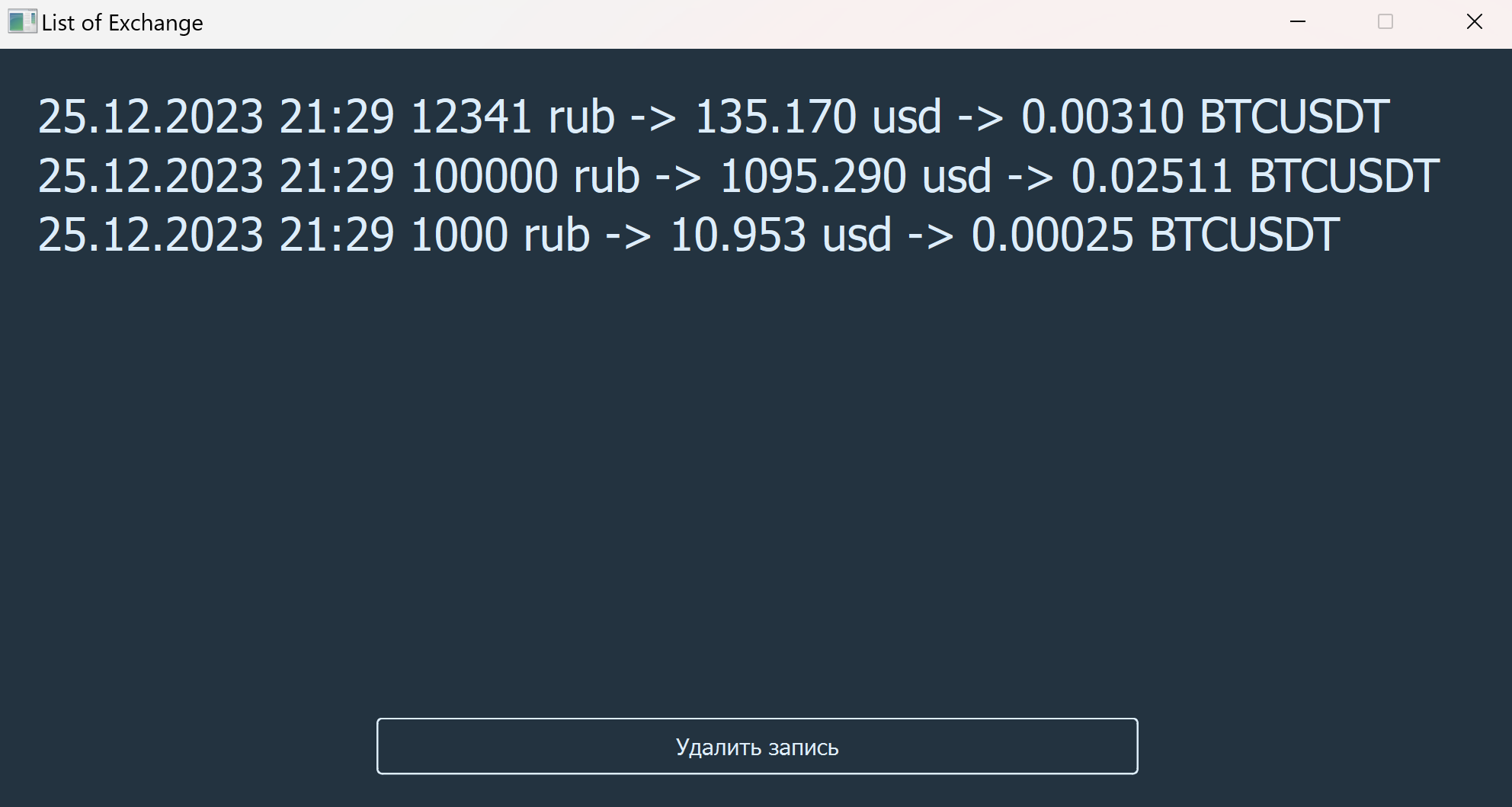
# Проверка выполнения требований

Отображение цен с нескольких бирж и автоматическое обновление

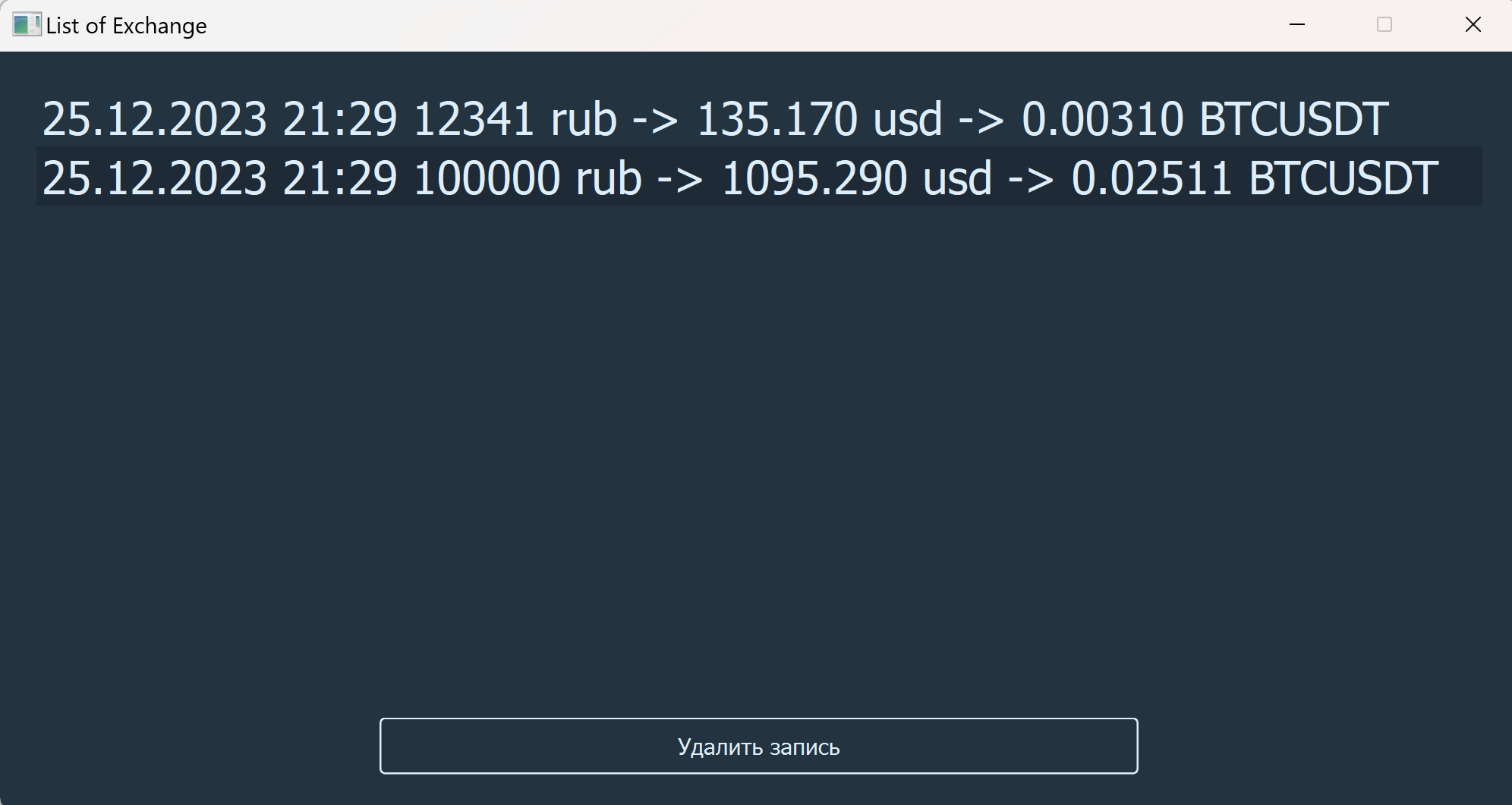


Конвертация рублей в выбранную криптовалюту  


Добавление в список конвертаций



Удаление записи



# Ссылка на GitHub с проектом

<https://github.com/a1ienforever/CurrencyExchange2>

# Использованная Литература

1. *Qt for Python: https://doc.qt.io/qtforpython-6/*
2. [*Requests: https://github.com/psf/requests4*](https://snoskainfo.ru/)
3. [*Beautiful Soup 4: https://beautiful-soup-4.readthedocs.io/en/latest/*](https://bing.com/search?q=%d0%9e%d1%84%d0%be%d1%80%d0%bc%d0%bb%d0%b5%d0%bd%d0%b8%d0%b5+%d1%81%d1%81%d1%8b%d0%bb%d0%be%d0%ba+%d0%bf%d0%be+%d0%93%d0%9e%d0%a1%d0%a2%d1%83)[*1*](https://bing.com/search?q=%D0%9E%D1%84%D0%BE%D1%80%D0%BC%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5+%D1%81%D1%81%D1%8B%D0%BB%D0%BE%D0%BA+%D0%BF%D0%BE+%D0%93%D0%9E%D0%A1%D0%A2%D1%83)

# Завершение

Данный документ описывает основные аспекты разработки мультивалютного конвертера с использованием многопоточного программирования. Реализация данного проекта будет включать в себя разработку структуры приложения, функционала для взаимодействия с пользователями. При разработке приложения будет уделено внимание краткости и эффективности кода.