Министерство сельского хозяйства Ставропольского края

ФГБОУ ВО СТАВРОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ

**«Разработка программы для работы пункта обмена валют»**

Выполнил:

Обучающийся 2 курса 22ИСИП9-2 группы

Специальность: 09.02.07 Информационные

системы и программирование

Дмитриев Дмитрий Владимирович

Ставрополь, 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc152401344)

[ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_Toc152401345)

[1.1. Описание задачи 6](#_Toc152401347)

[1.2. Обзор существующих решений 7](#_Toc152401351)

[1.3. Используемые технологии и инструменты](#_Toc152401348) 10

## ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

[2.1. Функциональные требования 1](#_Toc152401349)4

[2.2. Логика работы программы 14](#_Toc152401350)

[2.3. Создание базы данных 15](#_Toc152401350)

[2.4. Интерфейс программного модуля 16](#_Toc152401352)

[2.5. Описание структуры и основных классов 18](#_Toc152401352)

[2.6. Тестирование 23](#_Toc152401352)

[ЗАКЛЮЧЕНИЕ.](#_Toc152401355) 26

[ЛИТЕРАТУРА.](#_Toc152401355) 28

[ПРИЛОЖЕНИЯ.](#_Toc152401355) 29

# ВВЕДЕНИЕ

Данная курсовая работа разрабатывается в рамках учебной программы по специальности «Информационные системы и программирование» и связана с изучением МДК 01.01 «Разработка программных модулей». В рамках курсового проекта студентам предлагается разработать программный модуль для пункта обмена валют.

Целью данного проекта является обобщение знаний и умений, полученных студентами в области структурного и объектно-ориентированного программирования и их применение в реальной задаче разработки программного модуля. Основной задачей проекта является разработка модуля программного обеспечения, который позволит пользователю конвертировать валюты.

Разработка программных модулей стала неотъемлемой частью современного мира информационных технологий. Одним из таких модулей является «Обменник валют» - удобный и практический инструмент, предназначенный для управления и хранения информации о курсах различных валют и проведения операций по обмену валюты.

Цель данной курсовой работы заключается в разработке полноценного программного модуля «Обменник валют», способного эффективно управлять операциями по обмену валюты. Для достижения этой цели мы будем использовать современные технологии и методы разработки программного обеспечения, чтобы предоставить пользователям удобный и надежный инструмент для проведения операций по обмену валюты.

В ходе работы будут рассмотрены основные требования к функциональности модуля, а также его основные компоненты и архитектура. Будут проведены исследования, связанные с выбором оптимальных алгоритмов и структур данных для организации хранения записей. Кроме того, мы разработаем удобный и интуитивно понятный интерфейс пользователя, чтобы облегчить процесс работы с модулем.

В заключение работы будут проведены тестирование и анализ результатов, чтобы убедиться в корректной работе модуля и его соответствии.

Обменник валют представляет собой функциональный модуль, ориентированный на удобное и быстрое выполнение операций обмена различных валют. Реализация этой задачи требует учета ряда основных функциональных требований, которые можно распределить по следующим темам: авторизация пользователя, логирование операций, расширяемость количества доступных валют и адаптивность.

Для обеспечения безопасности и контроля доступа пользователей, в обменнике валют необходим механизм авторизации. Пользователи должны иметь возможность создать учетную запись и войти в систему, используя уникальные логин и пароль. Обменник валют должен также позволять пользователям восстановить пароль в случае его утраты.

Для учета и контроля каждой операции обмена валюты в системе необходимо вести логирование “операций. Это позволит сохранять историю совершенных операций и обладать надежной информацией об операциях, осуществленных пользователями в обменнике. Логирование может быть реализовано в виде базы данных, в которой будут храниться детали о каждой операции, такие как дата, время, валюта, сумма обмена и идентификатор пользователя.

Обменник валют должен быть гибким и расширяемым, чтобы иметь возможность работать с различными валютами. Для этого система должна поддерживать добавление новых валют и обновление курсов обмена. Модуль должен предоставлять возможность администратору обновлять информацию о валютах, а также добавлять новые валюты в систему. Это позволит обменнику быть актуальным и работать с широким спектром валют.

Адаптивность является важным аспектом в современных приложениях. Обменник валют должен быть способен адаптироваться к различным устройствам и размерам экранов. Это позволит пользователям удобно пользоваться сервисом обмена валют как на настольных компьютерах, так и на мобильных устройствах. Адаптивный дизайн позволит сохранить функциональность и удобство использования вне зависимости от размера экрана.

Таким образом, модуль обменника валют должен быть разработан с учетом основных функциональных требований, таких как добавление, просмотр, редактирование и удаление операций. Также необходимо обеспечить безопасность пользователей и ведение истории операций, а также предусмотреть адаптивность системы к различным устройствам и возможность расширения количества доступных валют.

# ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1. Описание задачи

Задача состоит в создании программного модуля, который будет представлять собой обменник валют с возможностью покупки и продажи. Основная цель модуля облегчить и упростить процесс обмена валюты, позволяя пользователям быстро и удобно конвертировать свои средства из одной валюты в другую.

В рамках задачи требуется разработать удобный интерфейс, который позволит пользователям легко взаимодействовать с операциями обмена и осуществлять их управление. Пользователи должны иметь возможность создавать новые операции, указывая соответствующие данные, такие как сумма обмена, исходная валюта, целевая валюта и другую информацию. Также должны быть предусмотрены возможности выбора желаемых валют из списка доступных, изменения типов операций, просмотра истории операций и отмены некорректных операций.

Обменник валют поможет пользователям быстро и удобно осуществлять обмен средств между различными валютами, сэкономив время и упростив процесс конвертации.

Необходимо предусмотреть функциональность по обмену и конвертации валюты, чтобы пользователи могли обменивать различные валюты и получать актуальную информацию о курсах обмена.

Кроме того, модуль должен обладать удобным и интуитивным интерфейсом, который позволит пользователям быстро освоиться с программой и использовать ее без надобности в дополнительном обучении.

В общем, задача заключается в создании эффективного и удобного инструмента для обмена и конвертации валюты, который будет полезен как для индивидуального использования, так и для командной работы.

Для реализации данной задачи мы используем современные технологии программирования, такие как язык Python, фреймворк PyQt5, базу данных SQLite и CSS для создания пользовательского интерфейса. PyQt5 обеспечит нам удобное взаимодействие с базой данных, а CSS позволит создать привлекательный и интуитивно понятный интерфейс для пользователей.

## 1.2. Обзор существующих решений

WebMoney является одним из наиболее распространенных обменных сервисов. Он предлагает обмен валюты между различными платежными системами и банками. WebMoney поддерживает большое количество валют, включая доллары США, евро, российские рубли и другие популярные валюты. Сервис предоставляет удобный интерфейс для совершения операций обмена, а также предлагает различные способы пополнения и вывода средств. Однако, следует отметить, что WebMoney может взимать комиссию за проведение операций обмена и наличие некоторых ограничений по сумме и времени обработки заявок.

Яндекс.Деньги также является популярным обменным сервисом, который предоставляет возможность обмена валюты и перевода денежных средств. Он поддерживает широкий спектр валют, включая доллары США, евро, британский фунт стерлингов и другие. Яндекс.Деньги предлагает удобный интерфейс с простыми и интуитивно понятными средствами навигации. Сервис также предоставляет возможность оплаты через банковские карты, электронные кошельки и другие платежные системы. Комиссии за проведение операций обмена в Яндекс.Деньги могут варьироваться в зависимости от выбранного способа обмена.

Qiwi – это еще один популярный обменный сервис, который предлагает простой и удобный способ обмена валюты. Он поддерживает несколько основных валют, включая доллары США, евро и российские рубли. Qiwi предоставляет возможность совершать операции обмена через интернет-платежные терминалы, мобильное приложение и другие способы. Сервис также позволяет пополнять и снимать средства с помощью банковских карт и электронных кошельков. Однако, Qiwi может взимать комиссию за проведение операций обмена и существуют ограничения по минимальной и максимальной сумме обмена.

Сбербанк Онлайн – это мобильное приложение Сбербанка, которое предоставляет возможность совершать операции по обмену валюты. Приложение поддерживает широкий спектр валют и позволяет быстро и удобно совершать операции обмена через счета и карты Сбербанка. Сбербанк Онлайн также предоставляет возможность просмотра текущих курсов валют и предлагает инструменты для расчета суммы после конвертации. Пользователи отмечают удобство интерфейса и надежность приложения, однако некоторые жалуются на некоторые задержки при обработке операций обмена.

Tinkoff Mobile – мобильное приложение Тинькофф Банка, которое также предоставляет функционал для обмена валюты. Приложение поддерживает различные валюты, включая доллары США, евро и другие. Оно предоставляет удобный интерфейс с интуитивно понятными функциями навигации и различными инструментами для расчета суммы после обмена. Пользователям нравится простота и доступность приложения, а также оперативность обработки заявок. Однако, некоторые отмечают, что приложение может иметь периодические технические сбои.

PayPal – это один из самых популярных сервисов обмена валюты, который предоставляет возможность совершать международные денежные переводы и обменивать валюту между различными странами. PayPal поддерживает широкий спектр валют и предоставляет удобный интерфейс для проведения операций обмена. Сервис также предлагает защищенные методы оплаты и персональные аккаунты для пользователей. Однако, некоторые пользователи отмечают высокие комиссии за проведение операций и длительные сроки обработки заявок.

TransferWise – это сервис, который предлагает обмен валюты по более выгодному курсу, чем у большинства банков и платежных систем. Он поддерживает широкий спектр валют и предоставляет удобный интерфейс для проведения операций обмена. Сервис также предлагает низкие комиссии за проведение операций и быстрые сроки обработки заявок. Пользователи отмечают простоту использования сервиса и прозрачность процесса обмена валюты.

Revolut – это финансовый сервис, который предоставляет бесплатные международные переводы и обмен валюты между различными странами. Revolut поддерживает большое количество валют и предоставляет удобное мобильное приложение для проведения операций обмена. Сервис также предлагает бесплатные пластиковые карты для удобного использования счетов. Пользователи высоко оценивают низкие комиссии и удобство использования Revolut, однако некоторые отмечают некоторые ограничения по минимальной и максимальной сумме обмена.

Исследование популярных обменных сервисов и мобильных приложений банков позволяет сделать следующие обобщенные выводы.

1. Обменные сервисы, такие как WebMoney, Яндекс.Деньги и Qiwi, предлагают пользователю удобный способ обмена валюты и перевода денежных средств. Они поддерживают широкий спектр валют и предоставляют разные способы пополнения и вывода средств.

2. У этих сервисов есть свои преимущества, такие как удобный интерфейс, доступность различных платежных систем и широкий выбор поддерживаемых валют.

3. В то же время, каждый из этих сервисов имеет некоторые недостатки, которые можно реализовать в рамках курсовой работы. Например, одним из недостатков может быть длительное время обработки заявок на обмен валюты или ограничения по максимальной сумме обмена. Также возможны комиссии за проведение операций обмена, которые могут быть существенными для некоторых пользователей.

В итоге, выполнение курсовой работы на тему анализа существующих решений в области обменных сервисов и мобильных приложений банков позволит изучить сильные и слабые стороны этих сервисов, а также предложить возможные пути их улучшения для обеспечения более удобного и эффективного обмена валюты для пользователей.

## 1.3. Используемые технологии и инструменты

Для создания данного обменника валют были использованы следующие технологии и инструменты:

- Python: Python является одним из самых популярных и востребованных языков программирования в мире. Он обладает простым и понятным синтаксисом, что делает его идеальным выбором для небольших проектов.

Одной из ключевых особенностей Python является его универсальность. С помощью Python можно разрабатывать как веб-приложения, так и научные вычисления, обработку данных, искусственный интеллект, создание игр и многое другое. Это делает Python идеальным выбором для широкого спектра проектов на практике.

Также Python обладает огромным количеством библиотек и фреймворков, что упрощает разработку и позволяет значительно ускорить процесс создания программного обеспечения. Кроме того, Python имеет активное сообщество разработчиков, которые готовы помогать друг другу и делиться опытом.

В целом, выбор Python для реализации проекта обусловлен его простотой, мощными возможностями, богатой экосистемой библиотек и активным сообществом разработчиков.

PyQT5: PyQt5 - это библиотека для создания графических пользовательских интерфейсов на языке программирования Python. Выбор пал на PyQt, потому что она обладает большим количеством функциональных возможностей и хорошо поддерживается сообществом разработчиков.

С помощью PyQt5 можно создавать красивые и эффективные интерфейсы, что важно для проекта. Богатый функционал библиотеки позволяет легко взаимодействовать с пользователем, использовать различные виджеты, создавать анимации и многое другое.

Библиотека PyQt5 также является кросс-платформенной, что означает, что разработанное с ее помощью приложение может работать и на Windows, и на MacOS, и на Linux без изменения исходного кода. Это делает PyQt5 удобным выбором для проекта, ориентированного на широкую аудиторию.

Кроме того, PyQt5 имеет хорошо разработанную документацию и активное сообщество, что облегчает процесс изучения и поддержки библиотеки.

Таким образом, выбор библиотеки PyQt5 для реализации проекта обусловлен ее мощными возможностями, кросс-платформенностью, хорошей документацией и поддержкой со стороны сообщества разработчиков.

-Forex\_python: Во-первых, эта библиотека предоставляет удобные инструменты для работы с валютными данными и обменными курсами. Она содержит широкий выбор функций для получения актуальных курсов валют из различных источников и обеспечивает возможность простого преобразования валюты.

Во-вторых, forex-python является популярной библиотекой, имеющей хорошую поддержку и обновления. Она активно разрабатывается и обновляется сообществом разработчиков, что гарантирует надежность, актуальность и стабильность ее функционала. Таким образом, использование forex-python позволит быть уверенным в достоверности получаемых данных и обеспечит стабильную работу обменника валют.

Кроме того, библиотека forex-python предоставляет документацию и примеры кода, что делает ее использование удобным для разработчика. Она предлагает различные методы работы с валютами, кросс-курсами и конвертацией валюты, а также позволяет получать и анализировать исторические данные. Это позволяет легко осуществить все необходимые операции, связанные с обменом валюты, для создания обменника валют.

Еще одним важным фактором, по которому выбор пал на библиотеку forex\_python, является ее простота использования. Она предоставляет интуитивно понятный интерфейс и простые в использовании функции, что позволяет даже относительно новым программистам легко осуществлять операции с валютами.

Библиотека forex-python также поддерживает работу с несколькими источниками данных о курсах валют. Это означает, что есть возможность выбрать наиболее надежный и актуальный источник для получения курсов валют. Это особенно важно для обменника валют, поскольку точность и актуальность данных являются ключевыми факторами для успешной операции обмена.

- SQLite3: Во-первых, SQLite3 является одним из наиболее популярных и широко используемых инструментов для работы с базами данных. Он был разработан таким образом, чтобы быть легким в интеграции, надежным и эффективным при работе с малыми и средними объемами данных. Библиотека SQLite3 имеет простой API и простой способ взаимодействия с базой данных, что делает ее подходящей для курсовой работы.

Во-вторых, SQLite3 является встраиваемой базой данных, что означает, что она не требует отдельного сервера или настроек для работы. Все данные хранятся в одном файле базы данных, что делает ее удобной для переноса и внедрения. Это особенно полезно для моей курсовой работы, так как можно легко передавать базу данных и работать с ней на любом устройстве без необходимости настройки и установки отдельного сервера.

В-третьих, SQLite3 имеет расширенные возможности для работы с данными и запросами. Она поддерживает SQL-стандарт и имеет широкий набор функций и операторов для извлечения, модификации и управления данными. Это позволяет мне эффективно работать с информацией о валютных обменах в моей курсовой работе, выполнять сложные запросы и анализировать данные.

В-четвертых, SQLite3 обеспечивает хорошую производительность при работе с небольшими и средними объемами данных. Это связано с его простотой и легковесностью. SQLite3 не требует дополнительных процессов или серверов, что сокращает накладные расходы на обработку запросов и управление соединениями. Это особенно важно для проектов масштаба обменника валют, где часто требуется быстрый доступ к информации и выполнение транзакций.

В-пятых, SQLite3 обладает надежностью и стабильностью. Библиотека SQLite3 хорошо протестирована и имеет долгую историю использования. Она предоставляет механизмы обработки ошибок, а также возможность резервного копирования и восстановления данных. Это важно для обменника валют, где точность данных и сохранность информации имеют первостепенное значение.

И, наконец, SQLite3 является свободно распространяемым и открытым исходным кодом. Это означает, что я могу легко получить доступ к исходному коду библиотеки, изучить его и внести необходимые изменения или расширения в соответствии с требованиями моей курсовой работы. Более того, SQLite3 имеет активное сообщество разработчиков, которое предоставляет поддержку и регулярные обновления. Это дополнительный бонус, который обеспечивает стабильность и надежность в использовании SQLite3 в моей курсовой работе.

В результате, использование библиотеки SQLite3 для разработки базы данных обменника валют в моей курсовой работе было обоснованным выбором, учитывая ее популярность, простоту интеграции, портабельность, производительность, надежность и открытый исходный код.

## ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1. Функциональные требования

Перед началом разработки необходимо определить функциональные требования к программному модулю. Функциональные требования к программе «обменник валют» включают следующие основные функции:

1. Обмен валют: Пользователи должны иметь возможность обменивать одну валюту на другую по актуальному курсу обмена. Осуществление операций обмена должно быть быстрым, надежным и безопасным.

2. Авторизация: Для доступа к функционалу программы пользователи должны пройти авторизацию. Это гарантирует безопасность и предотвращает несанкционированный доступ к аккаунтам пользователей.

3. Понятный интерфейс: Программа должна иметь интуитивно понятный и удобный интерфейс, который облегчит пользование ею. Пользователи должны легко находить необходимые функции.

4. Логирование операций: Каждая операция обмена валюты должна быть записана и сохранена в лог-файле или базе данных. Логирование операций позволяет отслеживать и контролировать все действия пользователей, а также предоставить возможность восстановления данных в случае сбоев или ошибок.

2.2 Логика работы программы. (рисунок 1)

UML-диаграммы используются для визуального представления архитектуры и структуры программного обеспечения. Они помогают разработчикам лучше понимать взаимосвязи между различными компонентами системы, а также облегчают коммуникацию между участниками разработки. На рисунке 1 представлена UML диаграмма работы с программой «Обменник валют».

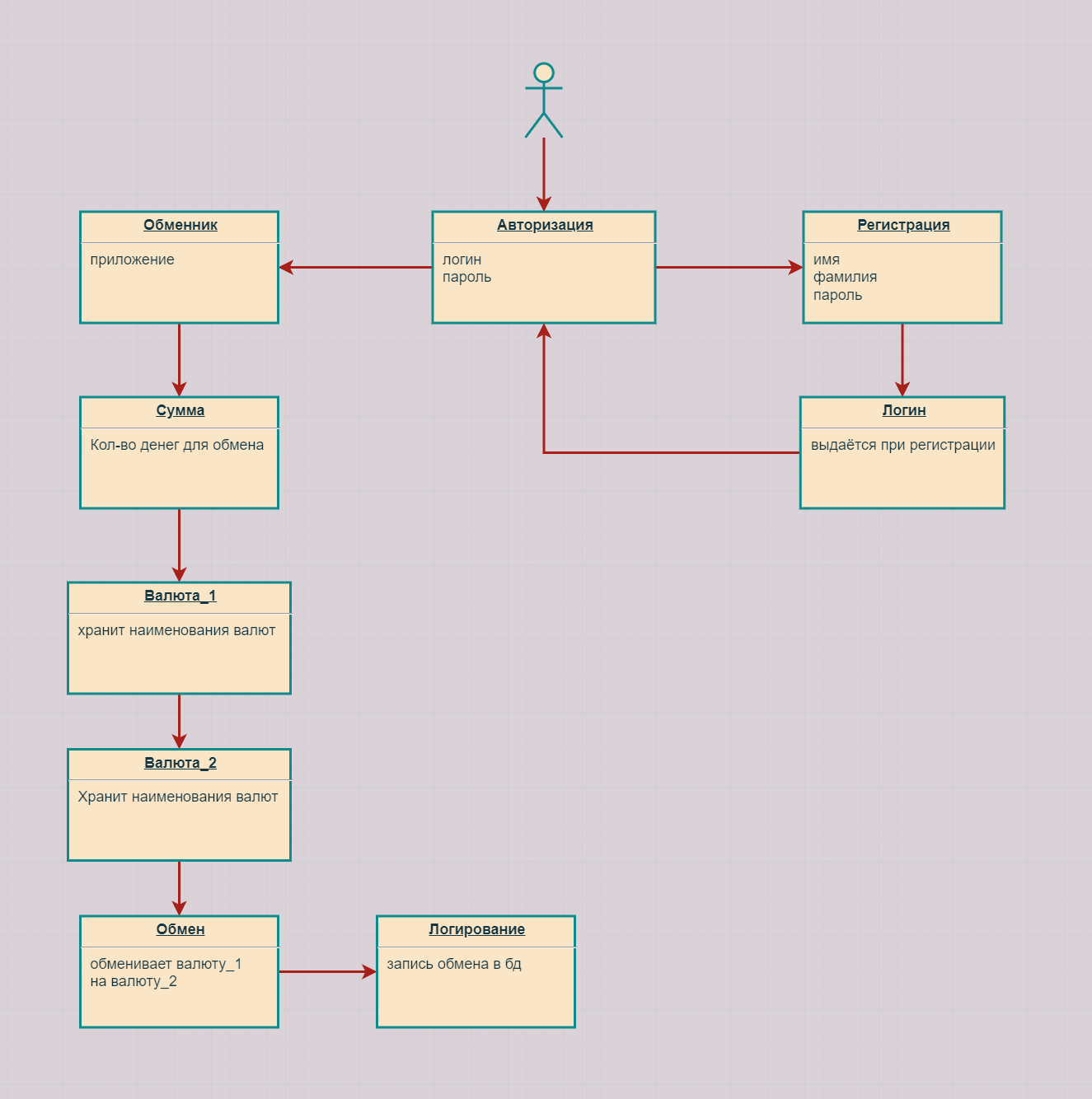


Рисунок 1. Схема работы приложения

2.3 Создание базы данных

Следуя задаче программы, создадим поля таблиц базы данных (таблица 1)

У сущности "Сотрудник" есть некоторые свойства: идентификатор, имя, фамилия и пароль. Идентификатор является первичным ключом.

Связь между «Сотрудником» и «Операциями» определена как «один к одному», так как каждый сотрудник имеет уникальный идентификатор, который служит идентификатором для операций.

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Имя поля | Тип данных | Описание |
| идентификатор | id | числовой | уникальный идентификатор, используемый для идентификации записи. |
| имя | name | текстовый | имя сотрудника |
| фамилия | surname | текстовый | фамилия сотрудника |
| пароль | password | текстовый | Пароль сотрудника |

Таблица 1. Структура отношения Сотрудник (staff)

У сущности «Обмен валют» есть несколько атрибутов: Номер операции, полученная валюта, количество полученной валюты, выданная валюта, курс, количество выданной валюты, дата сделки и кассир. Первичным ключом является номер операции.

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Имя поля | Тип данных | Описание |
| идентификатор | id | числовой | уникальный идентификатор |
| идентификатор сотрудника | staff\_id | числовой | идентификатор сотрудника |
| валюта входа | currency\_in | текстовый | указывает валюту, входной операции |
| валюта выхода | currency\_out | текстовый | указывает валюту, выходной операции. |
| сумма | amount | числовой | указывает сумму |
| дата | date | текстовый | указывает дату операции |
| время | time | текстовый | указывает время операции |

## Таблица 2. Структура отношения «Операции»

## 2.4. Интерфейс программного модуля

Интерфейс разрабатываемого программного модуля должен быть интуитивно понятным и удобным для пользователя. Он должен предоставлять возможность легко авторизовываться и конвертировать валюту.

Основные элементы интерфейса включают:

- Окно авторизации (рисунок 2):

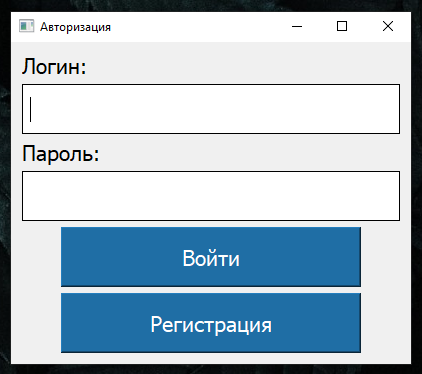


Рисунок 2. Окно авторизации

Окно авторизации предназначено для того, чтобы пользователь мог войти в свою учетную запись. Оно состоит из нескольких элементов, включающих поле ввода для ввода логина, поле ввода для ввода пароля, кнопку «Войти» и ссылки на регистрацию. Важно, чтобы окно авторизации было защищено от несанкционированного доступа и обеспечивало безопасность личных данных пользователя.

- Окно регистрации (рисунок 3):

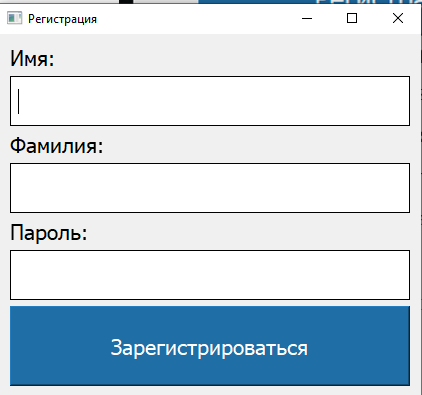


Рисунок 3. Окно регистрации

Окно регистрации предназначено для того, чтобы пользователь мог создать новую учетную запись. Оно содержит поля для ввода необходимой информации, такой как имя, фамилия, пароль, которая может потребоваться для регистрации. Окно регистрации может также содержит кнопку «Зарегистрироваться».

- Окно обмена валют (рисунок 4):

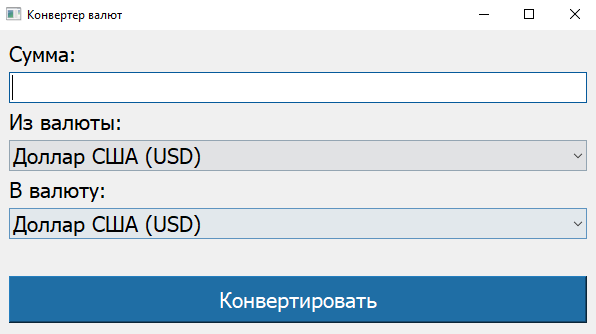


Рисунок 4. Главное окно

Окно обмена валют предоставляет пользователю возможность конвертировать одну валюту в другую. Оно содержит список доступных валют для выбора, поле для ввода суммы, которую пользователь хочет обменять, и поле для отображения результата обмена. Также кнопка «Конвертировать», которая запускает процесс обмена валюты. В нашем случае добавлены дополнительные параметры, такие как комиссионные, актуальность курса и другие настройки обмена.

## 2.5. Описание структуры и основных классов

Разрабатываемый программный модуль можно организовать по следующей структуре:

- Класс «AuthorizationWindow»,

1) Метод «\_\_init\_\_» (рисунок 5)

Код представляет собой инициализацию класса, предназначенного для создания окна конвертера валют. В методе происходит инициализация необходимых элементов интерфейса, таких как виджеты для ввода суммы и выбора валют, метки для отображения информации, и их настройка стилей (размер шрифта).



Рисунок 5. Метод «\_\_init\_\_»

3) Метод «authorization» (рисунок 6)

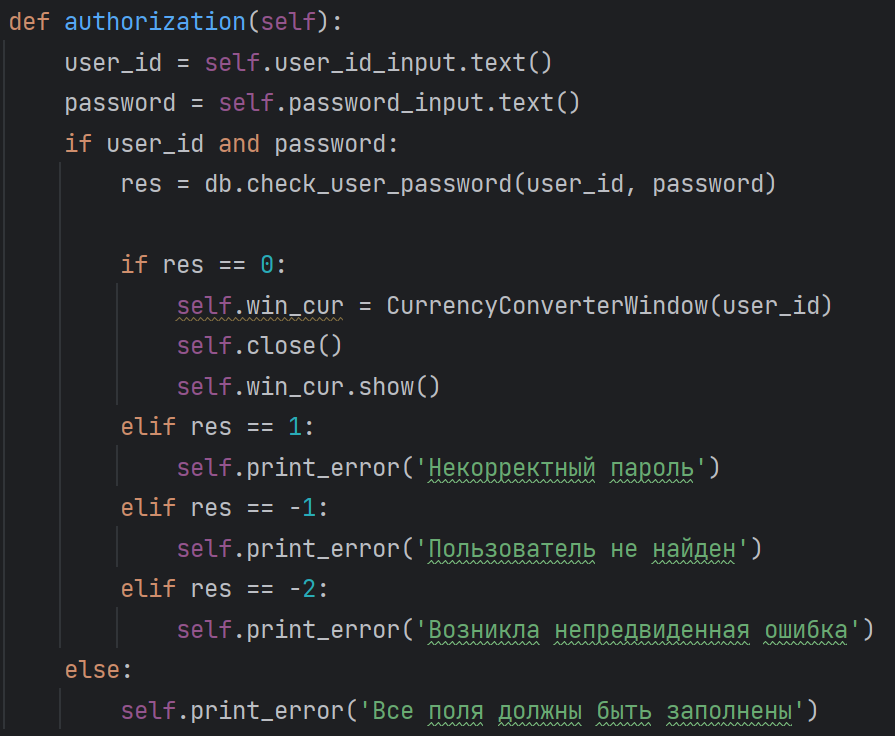


Рисунок 6. Метод «authorization»

Этот код представляет собой метод, который вызывается для процесса аутентификации пользователя. Сначала он извлекает идентификатор пользователя и пароль из полей ввода. Затем он проверяет, что оба поля были заполнены. Если они заполнены, код обращается к базе данных для проверки соответствия пользователя и пароля.

- Класс «RegistrationWindow»

1) Метод «create\_user» (рисунок 7)

Метод предназначен для создания нового пользователя. Он сначала получает данные ввода - имя, фамилию и пароль пользователя. Затем он проверяет, что все три поля заполнены. Если они заполнены, вызывается метод `db.new\_user`.

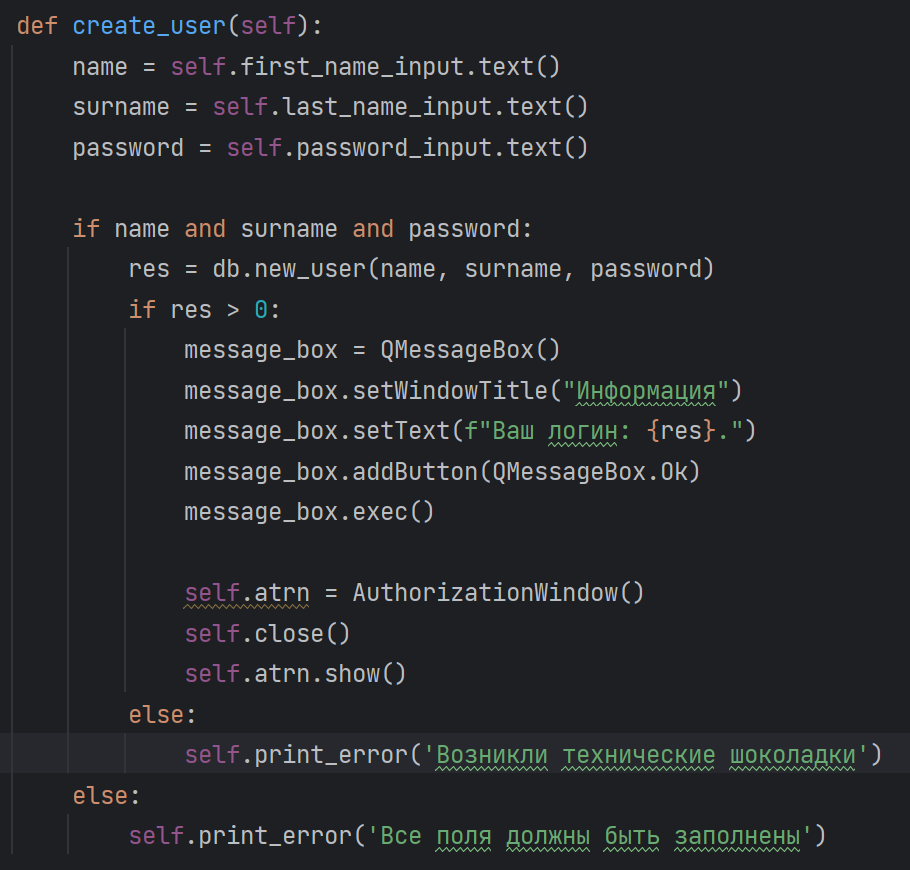


Рисунок 7. Метод «create\_user»

- Класс «CurrencyConverterWindow»

1) Метод «convert» (рисунок 8)

Данный метод `convert` представляет собой функцию, которая осуществляет конвертацию валют. Вначале он извлекает данные о сумме, из какой валюты и в какую валюту происходит конвертация. Далее происходит проверка наличия корректных данных о сумме, начальной и конечной валютах. После происходит конвертация валюты и вывод на экран результата.

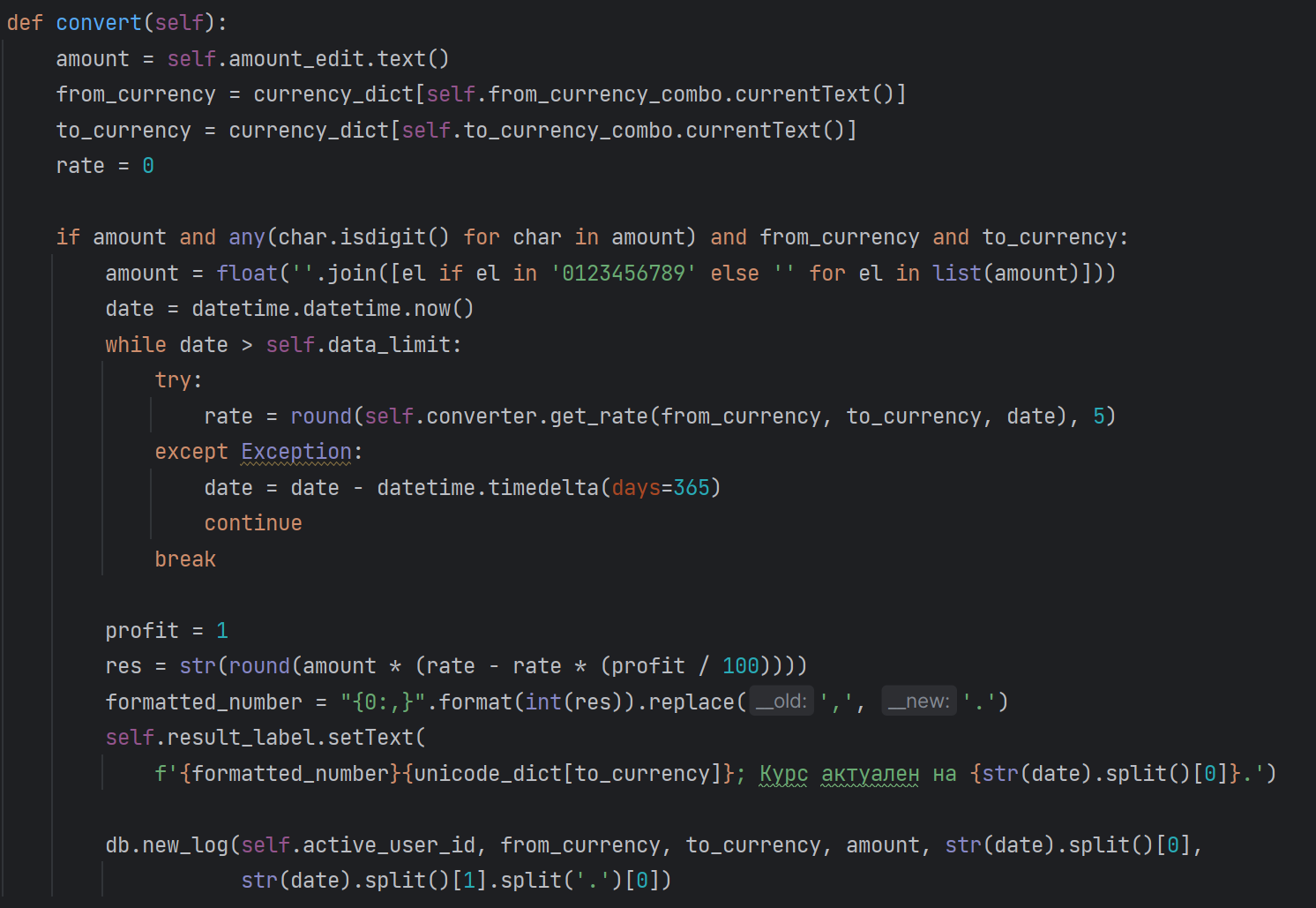


Рисунок 8. Метод «convert»

Класс `DataBase` отвечает за управление базой данных SQLite. Рассмотрим каждый метод подробнее:

1. `\_\_init\_\_(self)` (рисунок 9) - конструктор класса, который устанавливает соединение с базой данных и создаёт объект `cursor`, используемый для выполнения SQL-запросов. Также вызывается метод `create\_tables()` для создания таблиц, если они ещё не существуют.

2. `create\_tables(self)` (рисунок 9) - метод, который создаёт две таблицы в базе данных: `staff` и `operations`. Таблица `staff` содержит следующие поля: `id` (первичный ключ), `name` (имя сотрудника), `surname` (фамилия сотрудника) и `password` (пароль сотрудника). Таблица `operations` содержит следующие поля: `id` (первичный ключ), `staff\_id` (ссылка на идентификатор сотрудника), `currency\_in` (входящая валюта операции), `currency\_out` (исходящая валюта операции), `amount` (сумма операции), `date` (дата операции) и `time` (время операции). Он также устанавливает внешний ключ для поля `staff\_id`, связывающий таблицы `staff` и `operations`.

3. `new\_user(self, name, surname, password)` (рисунок 10) - метод для добавления нового сотрудника в таблицу `staff`. Он принимает имя, фамилию и пароль сотрудника, и добавляет их в базу данных. В случае успешного добавления, метод возвращает идентификатор нового сотрудника (`user\_id`). В случае возникновения ошибки, метод возвращает -1.

4. `check\_user\_password(self, user\_id, password)` (рисунок 10) - метод для проверки пароля сотрудника. Он принимает идентификатор сотрудника (`user\_id`) и пароль, и выполняет запрос к базе данных для получения пароля, связанного с данным идентификатором. Если пароль в базе данных совпадает с переданным паролем, метод возвращает True. В противном случае, метод возвращает False.

Класс `DataBase` обеспечивает функциональность работы с базой данных, включая создание таблиц, добавление новых сотрудников и проверку паролей.

Это основные методы класса DataBase, которые обеспечивают функциональность управления базой данных.

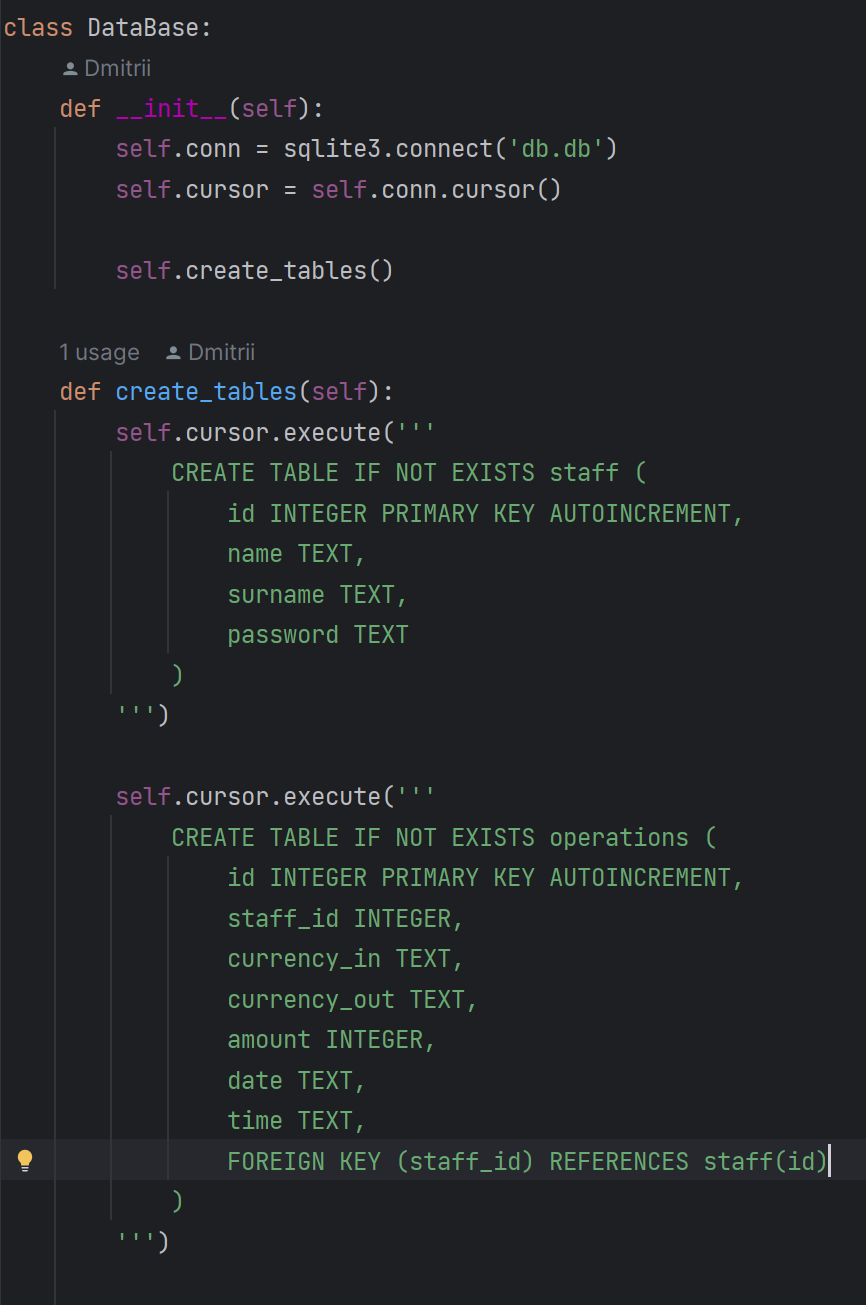


Рисунок 9. Методы «\_\_init\_\_» и «create\_tables»

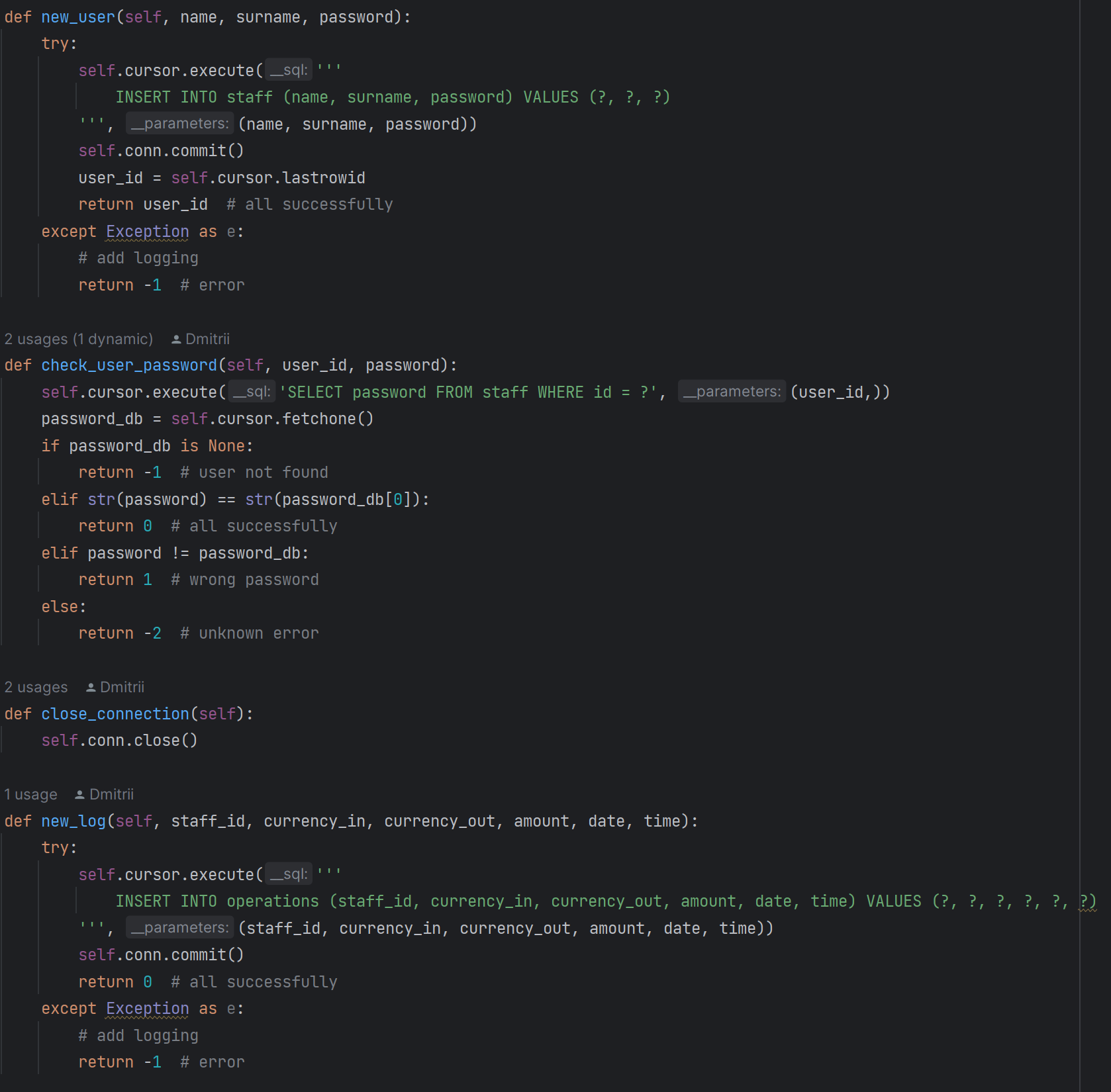


Рисунок 10. Методы «new\_user», «check\_user\_password», «close\_connection» и «new\_log».

2.6 Тестирование программного продукта

1. Тестирование авторизации:

- Проверка правильности ввода логина и пароля (рисунок 11).

- Проверка работы функции «Войти» для зарегистрированных пользователей (рисунок 12).

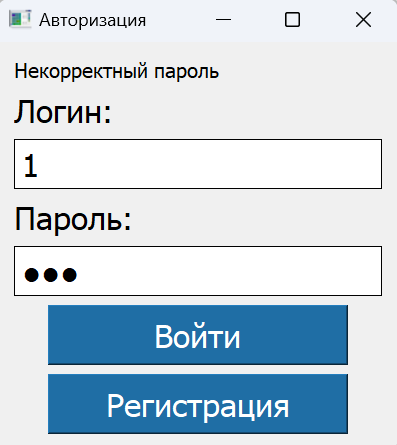


Рисунок 11. Тест окна входа

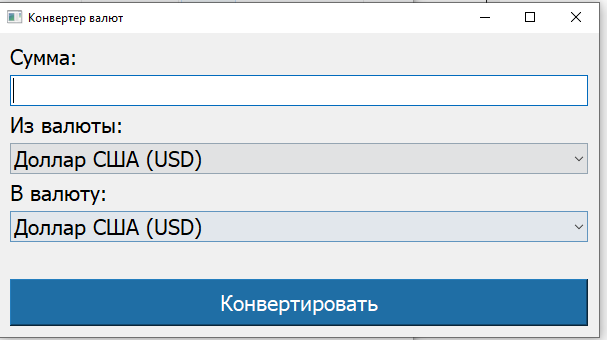


Рисунок 12. Тест входа в систему

2. Тестирование регистрации:

- Проверка успешной регистрации нового пользователя (рисунок 13).

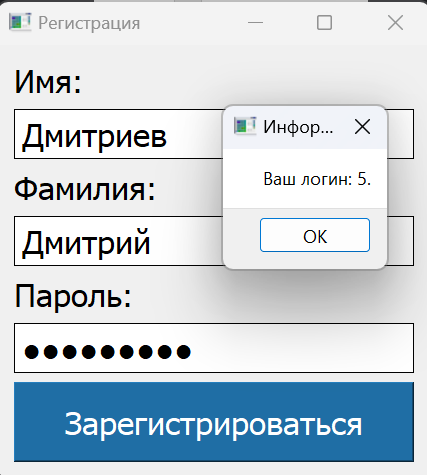


Рисунок 13. Тест окна регистрации

3. Тестирование функции обмена валют:

- Проверка корректности отображения списка доступных валют (рисунок 14).

- Проверка расчета конечной суммы после обмена (рисунок 15).

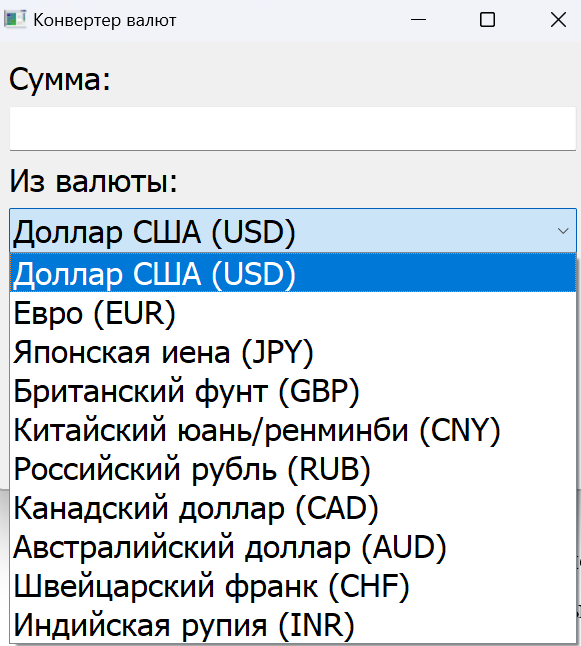


Рисунок 14. Тест функции выбора валют

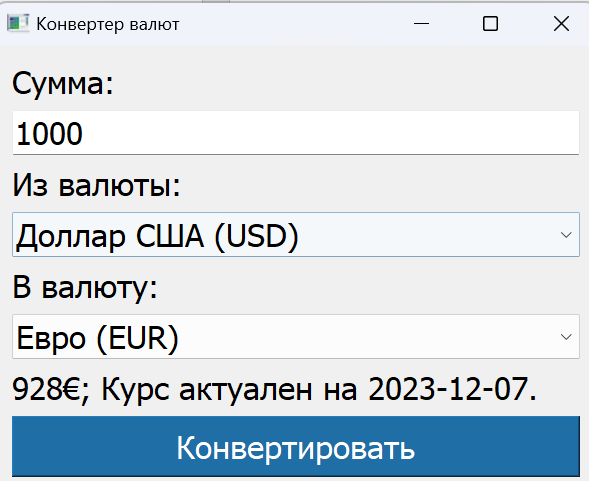


Рисунок 15. Тест функции конвертации валют

4. Тестирование логирования обмена валют:

- Проверка записи каждого успешного обмена в журнал (рисунок 16).

- Проверка правильности данных в журнале: базовая и целевая валюта, сумма обмена, дата и время.

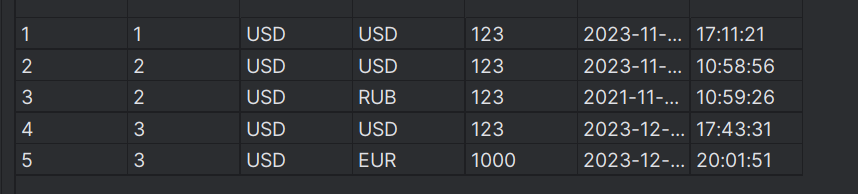


Рисунок 16. Демонстрация работы базы данных

5. Тестирование производительности:

- Проверка времени отклика при различных нагрузках (рисунок 7).

- Проверка стабильной работы при повышенной нагрузке (рисунок 7).

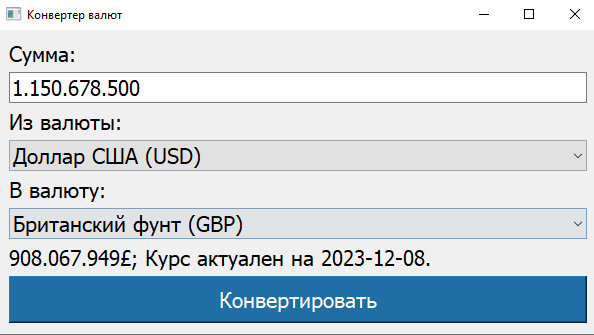


Рисунок 16. Тест работы при повышенной нагрузке

В результате тестирования мы убедились в корректной и безотказной работе программы «Обменник валют». Это означает, что приложение имеет конечный вид и готово к использованию в более сложных и масштабных системах.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе была рассмотрена тема «Программа для пункта обмена валют». Целью работы было разработать и реализовать программное решение, способное автоматизировать процесс обмена валют и обеспечить удобство как для клиентов, так и для сотрудников пункта обмена.

В процессе исследования данной темы был проведен анализ существующих программных решений, а также изучены требования и потребности пользователей. На основе полученных данных были определены основные функциональные и нефункциональные требования к программе для пункта обмена валют.

Подготовленное программное решение предлагает следующий функционал:

- Регистрация и аутентификация клиентов пункта обмена валют.

- Просмотр актуальных курсов валют.

- Оформление заказа на обмен валюты с указанием желаемой суммы.

- Автоматический расчет суммы для обмена на основе текущего курса.

- Печать чека с подробной информацией о сделке.

- Генерация отчетов о выполненных операциях.

- Учет и контроль оборота наличных денежных средств.

Основными преимуществами разработанной программы являются:

- Удобство использования для клиентов, благодаря интуитивно понятному интерфейсу.

- Автоматизация рутинных операций, что способствует повышению эффективности работы сотрудников пункта обмена валют.

- Возможность оперативного отслеживания изменений курсов валют, что позволяет адекватно реагировать на рыночные тенденции.

- Контроль оборота наличных денежных средств и генерация отчетов, что облегчает учетную и аналитическую работу.

В заключение можно отметить, что разработанная программа для пункта обмена валют представляет собой эффективный инструмент для упрощения и автоматизации работы обменного пункта. Она позволяет повысить уровень обслуживания клиентов и оптимизировать управление денежными потоками. Однако, чтобы успешно внедрить данную программу, необходимо провести дополнительное тестирование и адаптировать ее к специфическим требованиям конкретного пункта обмена валют.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. ГОСТ 7.32-2001. Система ГОСТ. Статьи в периодические издания. Структура и правила оформления.
2. Алан Харрис, Кристофер Негус «Python для детей: самое необходимое и полное руководство для начинающих», Питер, 2019, 400 стр.
3. Аллен Б. Дауни «Python. Экспресс-курс», Питер, 2018, 416 стр.
4. Даниэль Карфэ «Natural Language Processing with Python and NLTK», O'Reilly Media, 2016, 512 стр.
5. Джоэл Грус «Изучаем Python — программирование, которое выгодно отличается», ДМК Пресс, 2019, 384 стр.
6. Дэвид Бизли «Python в научных вычислениях», Диалектика, 2016, 600 стр.
7. Жан-Поль Картрон «Programming in Python 3, A Complete Introduction to the Python Language», Springer, 2011, 477 стр.
8. Чарльз Сейверанс «Python для детей: самое нужное и доступное руководство», Диалектика, 2017, 320 стр.
9. Хэрст Бьер «Python Essential Reference», Addission-Wesley Professional, 2018, 744 стр.
10. Майкл Доусон «Python для анализа данных», Диалектика, 2017, 464 стр.
11. Марк Саммерфилд «Python в примерах», Издательский дом «Вильямс», 2019, 768 стр.
12. Мартин Фауст «Introduction to PyQt5», independent, 2021, 388 стр.
13. Марк Лутц «Learning Python», O'Reilly Media, 2013, 1648 стр.
14. <https://pymanual.github.io/>
15. <https://pythontutor.ru/>
16. <https://habr.com/ru/articles/754400/>
17. <https://python-scripts.com/pyqt5>
18. <https://pythonist.ru/rukovodstvo-po-pyqt5/>

Приложение

from PyQt5.QtWidgets import QApplication

from PyQt5.QtWidgets import QWidget, QVBoxLayout, QLabel, QLineEdit, QPushButton, QHBoxLayout, QMessageBox, QComboBox

from PyQt5.QtGui import QFont

import sqlite3

import datetime

from forex\_python.converter import CurrencyRates

from db import DataBase

unicode\_dict = {

"USD": "\u0024", # Доллар США, "EUR": "\u20AC", # Евро

"JPY": "\u00A5", # Японская иена, "GBP": "\u00A3", # Британский фунт

"CNY": "\u00A5", # Китайский юань/ренминби, "RUB": "\u20BD",

"CAD": "\u0024", # Канадский доллар, "AUD": "\u0024", # А. доллар

"CHF": "\u20A3", # Швейцарский франк, "INR": "\u20B9" # Инд. рупия}

currency\_dict = {

'Доллар США (USD)': 'USD', 'Евро (EUR)': 'EUR',

'Японская иена (JPY)': 'JPY', 'Британский фунт (GBP)': 'GBP',

'Китайский юань/ренминби (CNY)': 'CNY', 'Российский рубль (RUB)': 'RUB',

'Канадский доллар (CAD)': 'CAD', 'Австралийский доллар (AUD)': 'AUD',

'Швейцарский франк (CHF)': 'CHF', 'Индийская рупия (INR)': 'INR'}

class CurrencyConverterWindow(QWidget):

def \_\_init\_\_(self, user\_id):

super().\_\_init\_\_()

self.active\_user\_id = user\_id

self.setWindowTitle('Конвертер валют')

self.resize(600, 300)

self.converter = CurrencyRates()

self.data\_limit = datetime.datetime(2015, 1, 1)

self.amount\_label = QLabel('Сумма:')

self.from\_currency\_label = QLabel('Из валюты:')

self.to\_currency\_label = QLabel('В валюту:')

self.result\_label = QLabel('')

self.amount\_label.setStyleSheet("font-size: 16pt;")

self.from\_currency\_label.setStyleSheet("font-size: 16pt;")

self.to\_currency\_label.setStyleSheet("font-size: 16pt;")

self.result\_label.setStyleSheet("font-size: 16pt;")

self.amount\_edit = QLineEdit()

self.from\_currency\_combo = QComboBox()

self.to\_currency\_combo = QComboBox()

self.result\_label = QLabel('')

self.amount\_edit.setStyleSheet("font-size: 16pt;")

self.from\_currency\_combo.setStyleSheet("font-size: 16pt;")

self.to\_currency\_combo.setStyleSheet("font-size: 16pt;")

self.result\_label.setStyleSheet("font-size: 16pt;")

from\_currency\_options = ['Доллар США (USD)', 'Евро (EUR)', 'Японская иена (JPY)', 'Британский фунт (GBP)', 'Китайский юань/ренминби (CNY)', 'Российский рубль (RUB)', 'Канадский доллар (CAD)', 'Австралийский доллар (AUD)', 'Швейцарский франк (CHF)', 'Индийская рупия (INR)']

to\_currency\_options = ['Доллар США (USD)', 'Евро (EUR)', 'Японская иена (JPY)', 'Британский фунт (GBP)', 'Китайский юань/ренминби (CNY)', 'Российский рубль (RUB)', 'Канадский доллар (CAD)', 'Австралийский доллар (AUD)', 'Швейцарский франк (CHF)', 'Индийская рупия (INR)'] self.from\_currency\_combo.addItems(from\_currency\_options)

self.to\_currency\_combo.addItems(to\_currency\_options)

self.convert\_button = QPushButton('Конвертировать')

self.convert\_button.setStyleSheet("font-size: 16pt;")

layout = QVBoxLayout()

layout.addWidget(self.amount\_label)

layout.addWidget(self.amount\_edit)

layout.addWidget(self.from\_currency\_label)

layout.addWidget(self.from\_currency\_combo)

layout.addWidget(self.to\_currency\_label)

layout.addWidget(self.to\_currency\_combo)

layout.addWidget(self.result\_label)

layout.addWidget(self.convert\_button)

self.setLayout(layout)

self.setStyleSheet(

"QPushButton {background-color: #1f6ea5; color: #ffffff; padding: 10px;}"

"QPushButton:hover {background-color: #288fd7;}")

self.convert\_button.clicked.connect(self.convert)

def convert(self):

amount = self.amount\_edit.text()

from\_currency = currency\_dict[self.from\_currency\_combo.currentText()]

to\_currency = currency\_dict[self.to\_currency\_combo.currentText()]

rate = 0

if amount and any(char.isdigit() for char in amount) and from\_currency and to\_currency:

amount = float(''.join([el if el in '0123456789' else '' for el in list(amount)]))

date = datetime.datetime.now()

while date > self.data\_limit:

try:

rate = round(self.converter.get\_rate(from\_currency, to\_currency, date), 5)

except Exception:

date = date - datetime.timedelta(days=365)

continue

break

profit = 1

res = str(round(amount \* (rate - rate \* (profit / 100))))

formatted\_number = "{0:,}".format(int(res)).replace(',', '.')

self.result\_label.setText(

f'{formatted\_number}{unicode\_dict[to\_currency]}; Курс актуален на {str(date).split()[0]}.')

db.new\_log(self.active\_user\_id, from\_currency, to\_currency, amount, str(date).split()[0], str(date).split()[1].split('.')[0])

class RegistrationWindow(QWidget):

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

super().\_\_init\_\_()

self.setWindowTitle("Регистрация")

self.resize(400, 300)

self.layout = QVBoxLayout()

self.error\_flag = True

self.first\_name\_label = QLabel("Имя:")

self.first\_name\_input = QLineEdit()

self.last\_name\_label = QLabel("Фамилия:")

self.last\_name\_input = QLineEdit()

self.password\_label = QLabel("Пароль:")

self.password\_input = QLineEdit()

self.password\_input.setEchoMode(QLineEdit.Password)

self.register\_button = QPushButton("Зарегистрироваться")

self.button\_layout = QHBoxLayout()

self.button\_layout.addStretch()

self.button\_layout.addWidget(self.register\_button)

self.button\_layout.addStretch()

self.layout.addWidget(self.first\_name\_label)

self.layout.addWidget(self.first\_name\_input)

self.layout.addWidget(self.last\_name\_label)

self.layout.addWidget(self.last\_name\_input)

self.layout.addWidget(self.password\_label)

self.layout.addWidget(self.password\_input)

self.layout.addLayout(self.button\_layout)

self.first\_name\_input.setFixedHeight(50)

self.last\_name\_input.setFixedHeight(50)

self.password\_input.setFixedHeight(50)

font = QFont()

font.setPointSize(16)

self.first\_name\_input.setFont(font)

self.last\_name\_input.setFont(font)

self.password\_input.setFont(font)

self.register\_button.setFont(font)

label\_font = QFont()

label\_font.setPointSize(16)

for i in range(self.layout.count()):

widget = self.layout.itemAt(i).widget()

if isinstance(widget, QLabel):

widget.setFont(label\_font)

self.register\_button.setMinimumSize(400, 80)

# Применяем стили

self.setStyleSheet(

"QWidget {background-color: #f0f0f0;}"

"QLabel {color: #000000;}"

"QLineEdit {background-color: #ffffff; color: #000000; border: 1px solid #000000; padding: 5px;}"

"QPushButton {background-color: #1f6ea5; color: #ffffff; padding: 10px;}"

"QPushButton:hover {background-color: #288fd7;})

self.setLayout(self.layout)

self.register\_button.clicked.connect(self.create\_user)

def print\_error(self, text):

if self.error\_flag:

self.error = QLabel()

self.error\_font = QFont()

self.error\_font.setPointSize(10)

self.error.setFont(self.error\_font)

self.layout.insertWidget(0, self.error)

self.error\_flag = False

self.error.setText(text)

def create\_user(self):

name = self.first\_name\_input.text()

surname = self.last\_name\_input.text()

password = self.password\_input.text()

if name and surname and password:

res = db.new\_user(name, surname, password)

if res > 0:

message\_box = QMessageBox()

message\_box.setWindowTitle("Информация")

message\_box.setText(f"Ваш логин: {res}.")

message\_box.addButton(QMessageBox.Ok)

message\_box.exec()

self.atrn = AuthorizationWindow()

self.close()

self.atrn.show()

else:

self.print\_error('Возникли технические шоколадки')

else:

self.print\_error('Все поля должны быть заполнены')

class AuthorizationWindow(QWidget):

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

super().\_\_init\_\_()

self.setWindowTitle("Авторизация")

self.resize(400, 300)

self.layout = QVBoxLayout()

self.error\_flag = True

self.user\_id\_label = QLabel("Логин:")

self.user\_id\_input = QLineEdit()

self.password\_label = QLabel("Пароль:")

self.password\_input = QLineEdit()

self.password\_input.setEchoMode(QLineEdit.Password)

self.authorization\_button = QPushButton("Войти")

self.register\_button = QPushButton("Регистрация")

self.button\_authorization\_layout = QHBoxLayout()

self.button\_authorization\_layout.addStretch()

self.button\_authorization\_layout.addWidget(self.authorization\_button)

self.button\_authorization\_layout.addStretch()

self.button\_register\_layout = QHBoxLayout()

self.button\_register\_layout.addStretch()

self.button\_register\_layout.addWidget(self.register\_button)

self.button\_register\_layout.addStretch()

self.layout.addWidget(self.user\_id\_label)

self.layout.addWidget(self.user\_id\_input)

self.layout.addWidget(self.password\_label)

self.layout.addWidget(self.password\_input)

self.layout.addLayout(self.button\_authorization\_layout)

self.layout.addLayout(self.button\_register\_layout)

self.user\_id\_input.setFixedHeight(50)

self.password\_input.setFixedHeight(50)

font = QFont()

font.setPointSize(16)

self.user\_id\_input.setFont(font)

self.password\_input.setFont(font)

self.register\_button.setFont(font)

self.authorization\_button.setFont(font)

label\_font = QFont()

label\_font.setPointSize(16)

for i in range(self.layout.count()):

widget = self.layout.itemAt(i).widget()

if isinstance(widget, QLabel):

widget.setFont(label\_font)

self.register\_button.setMinimumSize(300, 60)

self.authorization\_button.setMinimumSize(300, 60)

self.setStyleSheet(

"QWidget {background-color: #f0f0f0;}"

"QLabel {color: #000000;}"

"QLineEdit {background-color: #ffffff; color: #000000; border: 1px solid #000000; padding: 5px;}"

"QPushButton {background-color: #1f6ea5; color: #ffffff; padding: 10px;}"

"QPushButton:hover {background-color: #288fd7;}"

)

self.setLayout(self.layout)

self.authorization\_button.clicked.connect(self.authorization)

self.register\_button.clicked.connect(self.register)

def authorization(self):

user\_id = self.user\_id\_input.text()

password = self.password\_input.text()

if user\_id and password:

res = db.check\_user\_password(user\_id, password)

if res == 0:

self.win\_cur = CurrencyConverterWindow(user\_id)

self.close()

self.win\_cur.show()

elif res == 1:

self.print\_error('Некорректный пароль')

elif res == -1:

self.print\_error('Пользователь не найден')

elif res == -2:

self.print\_error('Возникла непредвиденная ошибка')

else:

self.print\_error('Все поля должны быть заполнены')

def register(self):

self.regw = RegistrationWindow()

self.close()

self.regw.show()

class DataBase:

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

self.conn = sqlite3.connect('db.db')

self.cursor = self.conn.cursor()

self.create\_tables()

def create\_tables(self):

self.cursor.execute('''

CREATE TABLE IF NOT EXISTS staff (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT,

surname TEXT,

password TEXT) ''')

self.cursor.execute('''

CREATE TABLE IF NOT EXISTS operations (

id INTEGER PRIMARY KEY AUTOINCREMENT,

staff\_id INTEGER,

currency\_in TEXT,

currency\_out TEXT,

amount INTEGER,

date TEXT,

time TEXT,

FOREIGN KEY (staff\_id) REFERENCES staff(id)

)

''')

self.conn.commit()

def new\_user(self, name, surname, password):

try:

self.cursor.execute('''

INSERT INTO staff (name, surname, password) VALUES (?, ?, ?)

''', (name, surname, password))

self.conn.commit()

user\_id = self.cursor.lastrowid

return user\_id # all successfully

except Exception as e:

return -1 # error

def check\_user\_password(self, user\_id, password):

self.cursor.execute('SELECT password FROM staff WHERE id = ?', (user\_id,))

password\_db = self.cursor.fetchone()

if password\_db is None:

return -1 # user not found

elif str(password) == str(password\_db[0]):

return 0 # all successfully

elif password != password\_db:

return 1 # wrong password

else:

return -2 # unknown error

def close\_connection(self):

self.conn.close()

def new\_log(self, staff\_id, currency\_in, currency\_out, amount, date, time):

try:

self.cursor.execute('''

INSERT INTO operations (staff\_id, currency\_in, currency\_out, amount, date, time) VALUES (?, ?, ?, ?, ?, ?)

''', (staff\_id, currency\_in, currency\_out, amount, date, time))

self.conn.commit()

return 0 # all successfully

except Exception as e:

return -1 # error

if \_\_name\_\_ == '\_\_main\_\_':

db = DataBase()

app = QApplication([])

win = AuthorizationWindow()

win.show()

app.exec\_()