**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное учреждение Московской области «Люберецкий техникум имени Героя Советского Союза, летчика-космонавта Ю.А.Гагарина»**

**КУРСОВАЯ РАБОТА**

Ивановой Сабрины Сергеевны

 по профессиональному модулю (дисциплине)

ОП.08 Основы проектирования баз данных

Курс 2 Группа № ИС-21

По теме: Проектирование базы данных автоматизированного учета займов и вкладов в коммерческом банке

Выполнила студентка     \_\_\_\_\_\_\_\_\_\_\_\_\_    \_Иванова Сабрина Сергеевна \_

                                              (подпись)               (ФИО полностью)

Руководитель                                               Тарджиманян Лия Николаевна

                                             (подпись)                (ФИО полностью)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_

  Дзержинский 2023

**Содержание**

[**ВВЕДЕНИЕ** 2](#_Toc135513265)

[**Глава 1 Теоретическая часть** 6](#_Toc135513266)

[**Глава 2 (Практическая часть) База данных автоматизированного учёта займов и вкладов в коммерческом банке** 17](#_Toc135513267)

[**2.1.  Выбор методологии проектирования** 17](#_Toc135513268)

[**2.2.  Построение концептуальной модели предметной области.** 17](#_Toc135513269)

[**2.3.  Проектирование логической структуры базы данных.** 18](#_Toc135513270)

[**2.4. Выявление полного перечня ограничений целостности, присущего данной предметной области.** 19](#_Toc135513271)

[**2.5. Организация ввода и корректировки данных в БД** 20](#_Toc135513272)

[**2.6 Описание информационных потребностей пользователей и их реализации.** 20](#_Toc135513273)

[**2.7. Разработка интерфейса** 22](#_Toc135513274)

[**2.8 Разбор кода** 23](#_Toc135513275)

[**ЗАКЛЮЧЕНИЕ** 26](#_Toc135513276)

[**СПИСОК ИСТОЧНИКОВ** 27](#_Toc135513277)

# **ВВЕДЕНИЕ**

**Цель курсового проектирования:**

Спроектировать БД автоматизированного учёта займов и вкладов в коммерческом банке. Для достижения поставленной цели необходимо выполнить следующие задачи:

**Задачи**

* Создать Er-диаграмму по техническому заданию.
* Создать базу данных по Er-диаграмме.
* Заполнить базу данных тестовыми данными.
* Создать авторизацию.
* Разработать формы для автоматизированного заполнения данными БД.
* Тестирование созданных форм на работоспособность.

**Инструменты:**

Visual Studio Code (VS Code) - интегрированная среда разработки (IDE) для программирования.

Для выполнения перечисленных задач также потребуется использование следующей СУБД:

SQLite - легкая и удобная система управления базами данных с простым функционалом и интерфейсом, что делает ее очень удобной для разработки и тестирования баз данных.

Для выполнения работы вы можете использовать VS Code в качестве IDE и SQLite в качестве СУБД для разработки, заполнения и выполнения запросов к базе данных.

**Выбор СУБД.**

Для разработки базы данных мною была выбрана СУБД SQLite.

Причины выбора данной СУБД:

1. Экономичность
2. Надежность
3. Портативность

**ПОСТАНОВКА ЗАДАЧИ:**

В данной курсовой работе будет рассмотрен весь жизненный цикл разработки базы данных: начиная от анализа требований и проектирования структуры базы данных, заканчивая созданием схемы базы данных и написанием запросов на языке SQL.

Окончательный результат данной работы будет представлять собой готовую базу данных, которая будет соответствовать требованиям предприятия и позволит эффективно регистрировать и обрабатывать документы, связанные с управленческой деятельностью и (или) кадровым учетом.

**Техническое задание:**

Общее описание проекта: Требуется разработать информационную систему для ведения учета вкладов в банке. Система должна предусматривать проверки данных клиентов, учет вкладов. В информационной системе необходимо хранить большое количество информации, вследствие чего разработка программного продукта начинается с создания базы данных, в которой хранится информация о клиентах.

1. спроектировать базу данных;
2. создать формы для работы с базой;
3. организовать пользовательское меню;
4. создать поисковые системы;
5. организовать сортировку данных;
6. сформировать отчеты.

В интерфейсе должно быть реализовано добавление новых данных в базу данных, возможность редактировать уже существующие данные. Также пользователь может удалять ненужные записи из базы данных. Должна быть реализована возможность просмотра полных таблиц, а также возможность выводить данные по определенным значениям. Интерфейс должен быть защищен от стороннего доступа к изменению целостности данных в БД.

# **Глава 1 Теоретическая часть**

Современный мир информационных технологий трудно представить себе без использования баз данных. Практически все системы в той или иной степени связаны с функциями долговременного хранения и обработки информации. Фактически информация становится фактором, определяющим эффективность любой сферы деятельности. Увеличились информационные потоки и повысились требования к скорости обработки данных, и теперь уже большинство операций не может быть выполнено вручную, они требуют применения наиболее перспективных компьютерных технологий. Базы данных создаются обычно не для решения какой-либо одной задачи для одного пользователя, а для многоцелевого использования. Базы данных отражают определенную часть реального мира. Эта информация должна по возможности фиксироваться в базе данных однократно, и все пользователи, которым эта информация нужна, должны иметь возможность работать с ней. Отсутствие централизованных методов управления доступом к информации послужило причиной разработки систем управления базой данных, а сами хранилища информации, которые работали под управлением данных систем, назывались базами или банками данных. Автоматизированные информационные системы ─ это совокупность информации экономико-математических методов и моделей технических, программных, технологических средств и специалистов, предназначенных для автоматизации деятельности, связанной с хранением, передачей и обработкой информации. Классическими примерами автоматизированных информационных систем являются банковские системы, автоматизированные системы управления предприятиями, системы резервирования авиационных или железнодорожных билетов и т. д. Целью данного дипломного проекта является разработка автоматизированной системы учета вкладов в банке. Автоматизация позволит работникам избавиться от многих рутинных процессов, что обеспечит наиболее быстрое, полное и качественное обслуживание клиентов.

Традиционные файловые системы предназначены для хранения небольших объемов данных и обеспечивают простоту в использовании. Однако, при увеличении объема данных, эти системы становятся неэффективными и не могут обеспечить необходимый уровень безопасности и устойчивости к сбоям. Именно поэтому информационные системы с базами данных (БД) стали более популярными. Базы данных обеспечивают более эффективное управление большими объемами данных и сложными приложениями, однако могут быть более сложными в использовании и затратными. Для работы с БД существуют различные системы управления базами данных (СУБД)

Архитектура СУБД определяет, как они организуют, хранят, управляют и обрабатывают данные в базе данных через три уровня: внешние схемы, концептуальная схема и физическая схема. Это позволяет определить логическую модель данных, структуру хранения данных и управлять транзакциями, обеспечивая безопасность и оптимизацию запросов. Цель архитектуры СУБД - обеспечить эффективное управление данными и защиту их конфиденциальности, целостности и доступности.

С развитием технологий базы данных, появились новые перспективы развития. Например, расширение облачных баз данных для масштабирования и уменьшения нагрузки на локальные серверы, использование искусственного интеллекта и машинного обучения для улучшения производительности.

Реляционная модель данных и объектно-ориентированная модель данных - это два разных подхода к хранению и управлению данными. В реляционной модели данные представлены в виде таблиц, а связи между данными описываются отношениями между таблицами. В объектно-ориентированной модели данных данные представлены в виде объектов, а связи между данными описываются ссылками между объектами.

Объектно-ориентированный подход к проектированию баз данных (ООП БД) использует концепции объектно-ориентированного программирования для проектирования баз данных. Классы объектов представляют собой структуры данных, а методы классов определяют операции, которые можно выполнять с этими данными. Наследование позволяет создавать более сложные структуры данных. ООП БД более гибок и расширяем, что делает его популярным для обработки больших объемов данных.

Объектно-ориентированный подход к проектированию ПО (ООП) также использует объекты для инкапсуляции данных и функций. ООП предоставляет множество механизмов для управления данными и моделями, включая наследование и полиморфизм. Инкапсуляция скрывает детали реализации от пользователей, что делает код более безопасным и легко поддерживаемым. ООП - популярный подход для разработки ПО в различных областях, включая создание приложений для работы с данными. Однако при проектировании баз данных также можно столкнуться с различными проблемами.

Сетевая модель данных - модель хранения данных в виде графа. Её преимущества: гибкость, быстрый доступ и эффективность. Недостатки: сложность, неоднородность и ограниченность.

Проблема слабых объектов возникает в моделировании баз данных, когда объекты не могут существовать без родительских объектов и не имеют своего уникального идентификатора. Примером может служить информация о доставке и товарах в базе данных интернет-магазина, которые привязаны к конкретному заказу. Решить эту проблему можно двумя способами: создать отдельные таблицы для каждого слабого объекта или добавить атрибуты слабых объектов в таблицу родительского объекта, что может усложнить запросы и привести к дублированию данных. Выбор подхода зависит от конкретной ситуации и требований к базе данных. Однако, использование слабых объектов может затруднить поддержку и развитие базы данных в будущем. При проектировании БД можно столкнуться с иными проблемами.

 Концептуальное проектирование баз данных - процесс разработки модели данных для описания предметной области и ее отношений. Недостаточная понятность требований, недостаток обратной связи с заказчиком и неправильное использование методологий могут стать методологическими проблемами. Проблемы с интеграцией, производительностью и безопасностью данных могут возникнуть в результате технологических проблем. Для снижения вероятности этих проблем, необходимо правильно выбирать методологию, технологии и инструменты, учитывать требования заказчика и обеспечивать обратную связь на всех этапах разработки.

Функциональная зависимость - это связь между атрибутами в отношении, где значение одного определяет значение другого. Она является важным аспектом проектирования баз данных и может быть найдена с помощью методов декомпозиции или алгоритмов, таких как Армстронга, основанных на анализе статистических свойств данных.

Удаление избыточных функциональных зависимостей - это процесс оптимизации базы данных, который заключается в удалении избыточных или не нужных для обеспечения целостности данных функциональных зависимостей. Чтобы удалить такие зависимости, необходимо проанализировать структуру базы данных и использовать нормальные формы, такие как 1NF, 2NF и 3NF. Они помогут избежать дублирования данных и улучшить эффективность запросов.

Многозначные зависимости и нормальные формы высокого порядка - это концепции, связанные с проектированием баз данных и их нормализацией. Многозначная зависимость возникает, когда один атрибут определяет значения двух или более других атрибутов. Нормальные формы высокого порядка расширяют основные нормальные формы (1NF, 2NF и 3NF) и используются для устранения многозначных зависимостей. Однако, использование нормальных форм высокого порядка может привести к усложнению структуры базы данных и увеличению количества таблиц. Поэтому, при проектировании базы данных, необходимо выбирать подходящие нормальные формы в зависимости от требований и задач базы данных.

Декомпозиция без потерь - это разбиение отношения в базе данных на более мелкие отношения, сохраняя все функциональные зависимости. Таким образом, можно восстановить исходное отношение путем объединения разбитых отношений. Функциональная зависимость - это отношение между атрибутами в таблице, где значение одного атрибута определяет значение другого. Для декомпозиции без потерь необходимо анализировать функциональные зависимости и разбивать отношение таким образом, чтобы каждый новый относительно был в нормальной форме и соответствовал одной или нескольким функциональным зависимостям. Однако, декомпозиция без потерь может увеличить количество таблиц и усложнить структуру базы данных, что может повлиять на производительность запросов. Необходимо выбирать подходящие функциональные зависимости и декомпозицию в зависимости от требований и задач базы данных.

Реляционная алгебра и реляционное исчисление - это два основных подхода к работе с реляционными базами данных. Реляционная алгебра предлагает набор операций для извлечения данных из таблиц, включая объединение, пересечение, выборку, проекцию и соединение. Реляционное исчисление - это формальный язык запросов для выборки данных, который включает в себя исчисление кортежей и исчисление доменов.

Оба подхода имеют свои преимущества и могут использоваться в различных сценариях. Простые операции, такие как выборка и проекция, часто выполняются с помощью реляционной алгебры, тогда как более сложные операции, такие как соединение, могут использоваться с помощью реляционного исчисления. Кроме того, операции реляционной алгебры могут использоваться для реализации запросов в реляционном исчислении.

Реляционное исчисление представляет собой формальный язык запросов для манипулирования данными в реляционных базах данных. Есть два варианта реляционного исчисления: исчисление кортежей и исчисление доменов. В исчислении кортежей запросы формулируются в терминах строк таблицы, а в исчислении доменов - в терминах значений столбцов таблицы. Оба варианта могут использоваться для формулирования запросов к базам данных, но на практике чаще используется исчисление кортежей.

Существует несколько способов доказательства эквивалентности реляционной алгебры и реляционного исчисления, один из которых - доказательство того, что любой запрос, написанный в одном формализме, может быть переписан в другом формализме без изменения его семантики. Это означает, что любой запрос на выборку данных, написанный в реляционной алгебре, может быть записан и выполнен в реляционном исчислении и наоборот, не нарушая его смыслового значения.

SQL89 и SQL92 - это стандарты языка SQL для работы с реляционными базами данных. SQL92 является более расширенной версией SQL89, вводящей новые функции и операторы. Одним из ключевых отличий между SQL89 и SQL92 является оператор JOIN, который был введен в SQL92 для объединения данных из нескольких таблиц, в то время как SQL89 использовал только WHERE и AND.

SQL92 также ввел понятие внешних ключей для связывания данных из разных таблиц, чего не было в SQL89. Этот стандарт также позволяет использовать подзапросы внутри других запросов и вводит новые строковые функции, такие как SUBSTRING и TRIM, которые не были определены в SQL89.

Другое отличие между SQL89 и SQL92 заключается во введении системных таблиц для хранения информации о базе данных и ее объектах, а также в добавлении новых типов данных, таких как BOOLEAN и INTERVAL, которых не было в SQL89. Несмотря на это, многие базы данных до сих пор используют SQL89, поскольку он более прост в использовании.

MS SQLite - это компактная встраиваемая библиотека, реализующая SQL-диалект, совместимый со стандартом SQL-92. Она поддерживает операторы SQL, включая операторы создания и управления таблицами, индексами и триггерами, а также множество функций, таких как математические и агрегатные функции. MS SQLite не имеет отдельного сервера баз данных, поэтому не поддерживает полноценную многопользовательскую работу и некоторые расширения SQL, такие как хранимые процедуры и функции. MS SQLite может быть использована в различных приложениях для работы с локальными базами данных.

При выборе базы данных для проекта необходимо учитывать такие критерии, как производительность, масштабируемость, надежность и стоимость. Майнфреймы имеют высокую производительность, но могут быть дорогими в обслуживании и разработке, в то время как файловые серверы более доступны, но могут потребовать дополнительных затрат на оборудование и сетевые ресурсы. Базы данных клиент/сервер могут быть более доступными, но могут потребовать дополнительных затрат на сетевое оборудование и программное обеспечение.

Логика приложения - это неотъемлемая часть любого программного продукта, определяющая его функциональность и способность взаимодействовать с пользователем. Код, написанный в рамках логики приложения, обрабатывает данные, осуществляет операции и взаимодействует с пользователем, делая приложение полезным и удобным в использовании.

Чтобы повысить масштабируемость и сопровождаемость приложения, используется многоуровневая архитектура, разделяющая его на различные слои. Клиентский слой обеспечивает взаимодействие с пользователем, бизнес-логический слой содержит логику приложения, а слой данных управляет информацией. Разделение на слои позволяет легко изменять и модифицировать каждый из них без изменения всего приложения.

Клиентский и бизнес-логический слои работают на разных устройствах и взаимодействуют посредством протоколов и интерфейсов, повышая эффективность и надежность приложения. Важно придерживаться принципов систем клиент/сервер, включающих разделение обязанностей, прозрачность распределения, масштабируемость, надежность и безопасность.

При оценке систем клиент/сервер принимаются во внимание критерии производительности, надежности, масштабируемости, безопасности и удобства использования. Все эти критерии помогают определить эффективность системы и ее пригодность для использования в конкретных условиях.

Логика приложения - это неотъемлемая часть любого программного продукта, определяющая его функциональность и способность взаимодействовать с пользователем. Код, написанный в рамках логики приложения, обрабатывает данные, осуществляет операции и взаимодействует с пользователем, делая приложение полезным и удобным в использовании.

Чтобы повысить масштабируемость и сопровождаемость приложения, используется многоуровневая архитектура, разделяющая его на различные слои. Клиентский слой обеспечивает взаимодействие с пользователем, бизнес-логический слой содержит логику приложения, а слой данных управляет информацией. Разделение на слои позволяет легко изменять и модифицировать каждый из них без изменения всего приложения.

Клиентский и бизнес-логический слои работают на разных устройствах и взаимодействуют посредством протоколов и интерфейсов, повышая эффективность и надежность приложения. Важно придерживаться принципов систем клиент/сервер, включающих разделение обязанностей, прозрачность распределения, масштабируемость, надежность и безопасность.

При оценке систем клиент/сервер принимаются во внимание критерии производительности, надежности, масштабируемости, безопасности и удобства использования. Все эти критерии помогают определить эффективность системы и ее пригодность для использования в конкретных условиях.

Распределенные системы баз данных (РСБД) - это сложные компьютерные системы, которые управляют хранением и обработкой данных в распределенной среде. Централизованные РСБД обеспечивают высокую производительность и надежность, но могут иметь проблемы с масштабируемостью и доступностью. С другой стороны, распределенные РСБД обладают высокой масштабируемостью и доступностью, но могут столкнуться с проблемами согласования и синхронизации данных.

Для успешной разработки РСБД необходимо учитывать ряд принципов. Важно обеспечивать разделение данных, репликацию данных, согласованность данных, выполнение транзакций в соответствии с принципами ACID, масштабируемость, безопасность данных, отказоустойчивость и управление конфликтами. Каждый из этих принципов играет важную роль в обеспечении высокой доступности, масштабируемости, отказоустойчивости и безопасности данных в распределенной среде.

Управление распределенной информацией может стать сложной задачей. Некоторые из проблем, которые могут возникнуть, включают согласованность данных, безопасность, управление версиями, сложность интеграции и масштабируемость. Несмотря на эти проблемы, РСБД остаются необходимой частью современных информационных технологий, и правильное управление данными может значительно повысить эффективность и надежность бизнес-процессов.

Хранилище данных - это система, которая позволяет организовывать и хранить большие объемы данных, доступные для аналитики и бизнес-пользователей. При проектировании такой системы учитываются принципы целостности, единообразия, интеграции, гибкости, производительности, безопасности, управления метаданными и поддержки различных запросов. Это позволяет эффективно использовать данные для принятия решений в бизнесе.

Однако для обработки и хранения больших объемов данных могут потребоваться распределенные хранилища данных. Это системы хранения, которые объединяют несколько узлов на разных компьютерах. Такие системы обеспечивают быстрый доступ к данным, высокую отказоустойчивость и балансировку нагрузки. Важными факторами при проектировании таких систем являются согласованность данных, управление репликацией и целостность данных. Распределенные хранилища данных широко применяются в различных отраслях.

Однако проектирование хранилищ данных может быть сложным процессом. Недостаточное понимание бизнес-требований и данных, неправильная модель данных, проблемы с производительностью и безопасностью, неправильный выбор технологий и структуры данных могут привести к проблемам в дальнейшей работе системы. Поэтому при проектировании хранилищ данных необходимо учитывать все эти факторы, чтобы создать эффективную и надежную систему хранения и обработки данных.

Администрирование баз данных (БД) является критической функцией для эффективности, безопасности и целостности информационных систем. Оно включает установку, настройку, мониторинг производительности, резервное копирование, управление безопасностью и схемой. Важно учитывать стандарты безопасности, производительности, резервного копирования и управления схемой при администрировании БД.

Безопасность и целостность данных также являются важными аспектами для БД. Несанкционированный доступ, ошибки проектирования, SQL-инъекции и другие атаки могут угрожать безопасности и целостности БД. Для этого нужно использовать аутентификацию, авторизацию, регулярное резервное копирование, шифрование данных и программное обеспечение для защиты от уязвимостей. Также необходимо устанавливать обновления и патчи для компонентов системы.

Сжатие данных может улучшить производительность и экономить место на диске. Для этого используются различные методы сжатия, включая алгоритмы сжатия без потерь и с потерями. Однако, создание и сжатие больших информационных массивов может приводить к сложностям с доступом к данным, управлению ими и безопасности. Поэтому важно использовать современные методы и инструменты для решения этих проблем.

Среда WEB отлично подходит для создания приложений баз данных, которые могут быть использованы через Интернет с помощью веб-браузера. Для разработки таких приложений используются различные языки программирования, такие как PHP, JavaScript, Ruby, Python и другие, а также специализированные фреймворки и библиотеки, которые значительно упрощают процесс разработки.

Для того чтобы создать структуру базы данных, используются ER-диаграммы, которые помогают понять связи между таблицами и полями. Для проектирования, анализа, тестирования и документирования баз данных используются CASE-средства.

Для тестирования новых приложений и изменений в существующих используются специальные базы данных для тестирования. Для управления и обработки данных в базе данных используются Системы управления базами данных (СУБД), которые позволяют эффективно управлять и обрабатывать данные.

Оптимизация производительности базы данных - важный аспект ее работы. Для этого используются инструменты мониторинга производительности, которые помогают оптимизировать ее работу.

Резервное копирование и восстановление данных - это важный аспект безопасности базы данных. Для этого используются специальные инструменты, которые создают резервные копии и восстанавливают данные при необходимости.

Анализ данных - это еще один важный аспект использования баз данных. Для этого используются инструменты анализа данных, которые помогают извлекать информацию из базы данных и анализировать ее.

Безопасность баз данных - это комплекс мер, направленных на защиту данных от несанкционированного доступа, изменений, кражи или уничтожения. Основные меры безопасности включают аутентификацию и авторизацию пользователей, шифрование данных, аудит базы данных, резервное копирование и восстановление данных, правильную конфигурацию системы, физическую безопасность и обучение пользователей.

Для обеспечения безопасности серверов баз данных используются меры, такие как резервирование, кластеризация, мониторинг, защита от несанкционированного доступа и злоупотребления, регулярное обновление и обслуживание, а также физическая защита от пожаров, наводнений и краж.

# **Глава 2 (Практическая часть) База данных автоматизированного учёта займов и вкладов в коммерческом банке**

## **2.1.  Выбор методологии проектирования**

Для проектирования базы данных автоматизированного учета займов и вкладов в коммерческом банке можно использовать методологию Entity-Relationship (ER). Эта методология позволяет описать сущности, атрибуты и связи между ними.

## **2.2.  Построение концептуальной модели предметной области.**

На основе технического задания в базе данных должны присутствовать следующие сущности: сотрудники, клиенты, данные о видах вкладов, вклады и договора. У которых должны присутствовать атрибуты и связи, точная схема базы данных.

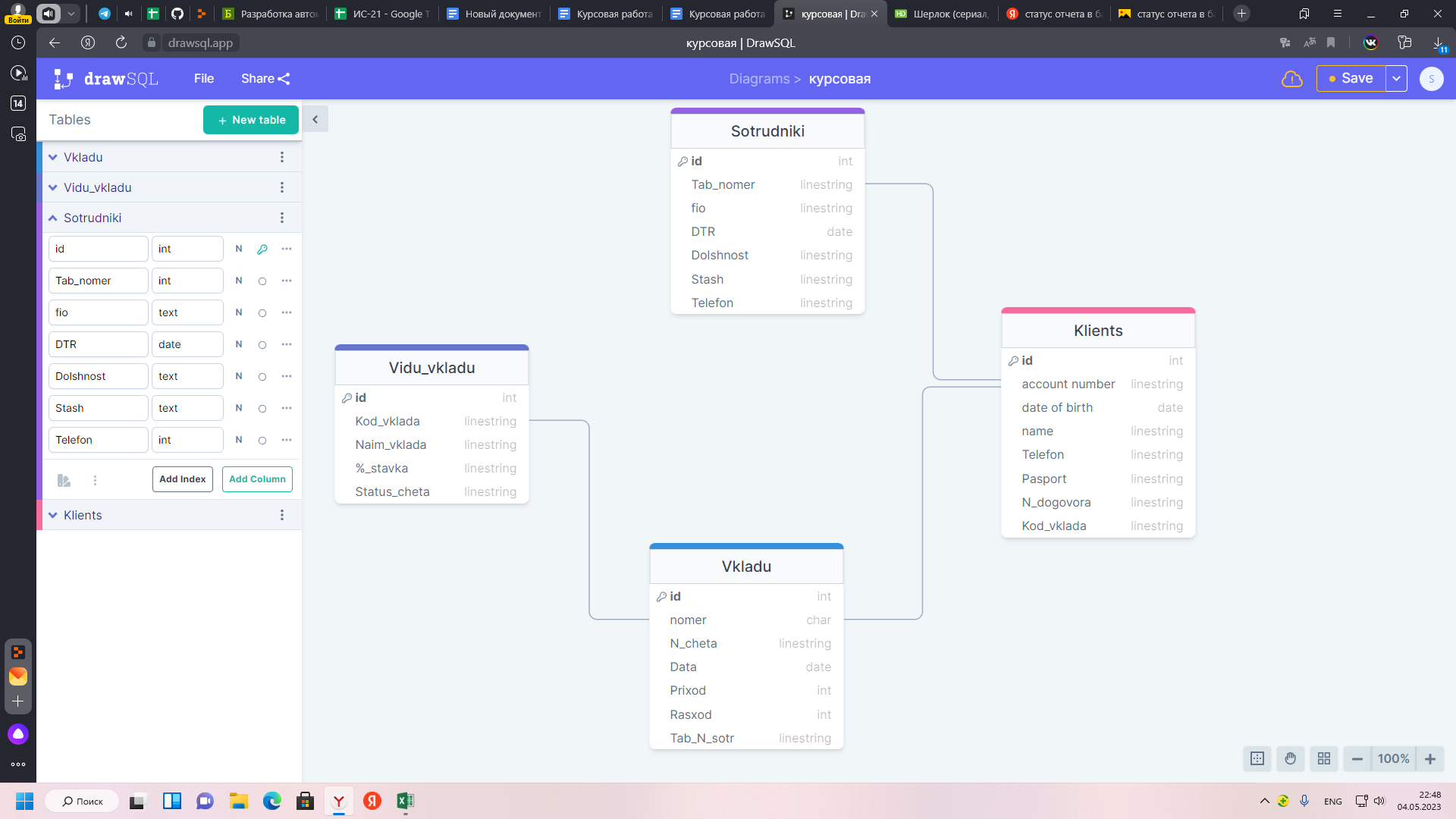
****

Рисунок 1. ER-диаграмма базы данных.

## **2.3.  Проектирование логической структуры базы данных.**

На базе er-диаграммы создаём базу данных в Sqlite. У нас есть четыре сущности: сотрудники, договора, клиенты, вклады, данные о видах вкладов.

Таблица “Сотрудники” имеет следующие атрибуты:

* ID
* Табельный номер
* ФИО
* Дата рождения
* Должность
* Стаж
* Телефон

Таблица “Клиенты” имеет следующие атрибуты:

* Номер счета
* ФИО
* Дата рождения
* Телефон
* Паспорт
* Номер договора
* Код вклада

Таблица “Данные о видах вклада” имеет следующие атрибуты:

* Код вклада
* Наименование вклада
* Процентная ставка
* Статус счета

Таблица “Данные о вкладах” имеет следующие атрибуты:

* Номер
* Номер счета
* Дата
* Приход
* Расход
* Табельный номер сотрудника

Для реализации контроля целостности были использованы встроенные средства СУБД SQLite, такие как ограничения NOT NULL, UNIQUE и FOREIGN KEY.

В качестве типов данных были выбраны наиболее подходящие для каждого поля: INTEGER для идентификаторов, TEXT для строковых значений, REAL для дат и времени.

## **2.4. Выявление полного перечня ограничений целостности, присущего данной предметной области.**

В предметной области автоматизированного учета займов и вкладов в коммерческом банке могут быть следующие ограничения целостности:

1. Уникальность id клиента
2. Обязательность заполнения поля "номер счета ", "паспортные данные", "ФИО"

Для контроля уникальности id клиента можно использовать ограничение PRIMARY KEY на соответствующем поле таблицы. Для контроля обязательности заполнения полей можно использовать ограничение NOT NULL на соответствующих полях.

## **2.5. Организация ввода и корректировки данных в БД**

Первичный ввод записей в базу данных будет производится при помощи импортирования данных из Excel. Далее ввод данных будет производится непосредственно с формы.

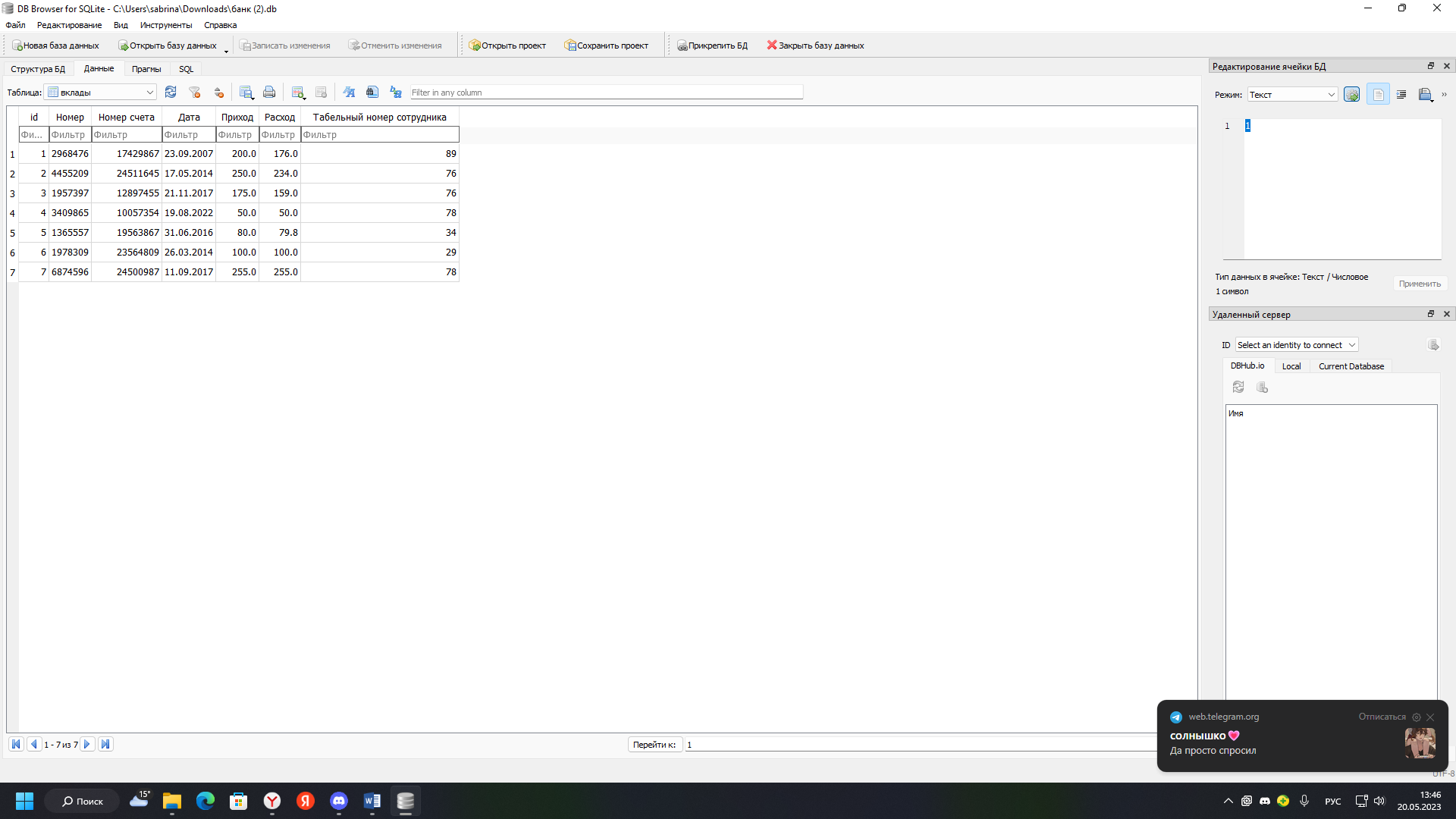


Рисунок 2. Ввод данных в Excel и корректировка данных в SQLIte

## **2.6 Описание информационных потребностей пользователей и их реализации.**

У пользователя могут возникнуть следующие потребности при пользовании интерфейсом:

* авторизация на форме;
* добавление данных в базу данных;
* изменение данных в таблице;
* удаление данных из базы данных;
* поиск информации по определенным значениям.

Общие требования к программному продукту Программный продукт должен обладать высокой степенью надежности. Во всем программном продукте должно использоваться ограничение ввода данных в поля таблиц. Перед занесением новых данных должна производиться проверка введенной информации на соответствие.

Программный продукт должен обладать высоким быстродействием, производит все действия за короткий промежуток времени. Создаваемый программный продукт позволяет:

* вести автоматизированный учет вкладов в банке;
* составлять отчетную ведомость.

Перед работой с интерфейсом нужно запустить ее, для этого необходимо нажать на кнопку “Открыть”, после чего в нижнем окне должна отобразиться таблица с данными.

Чтобы добавить новую запись в базу данных необходимо заполнить все необходимые ячейки, после чего нажать на кнопку добавить, данные будут успешно добавлены в таблицу.

Для изменения уже существующих данных необходимо заполнить все поля и указать id записи из таблицы, в которой будут производится изменения.

Чтобы удалить ненужную информацию из базы данных необходимо в окне, в котором отображается таблица, выбрать запись и нажать на нее, после чего кликнуть на кнопку “Удалить”.

Для вывода данных по определенным значением нужно переместится в блок поиска. Нажав на выпадающий список нужно выбрать столбец и таблицы по которому будет производиться поиск, после чего нужно ввести необходимые значения для поиска в строку ниже и нажать на кнопку “Найти”. В окне, где отображалась таблица появятся данные, которые Вы искали.

Для переключения с одной формы на другую внизу страницы предусмотрены кнопки, при нажатии на необходимую кнопку будет осуществляться переход на соответствующую страницу интерфейса.

## **2.7. Разработка интерфейса**

Для начала необходимо создать внешний вид форм, перенести на окно необходимые кнопки, поля для ввода данных, текстовые значения.

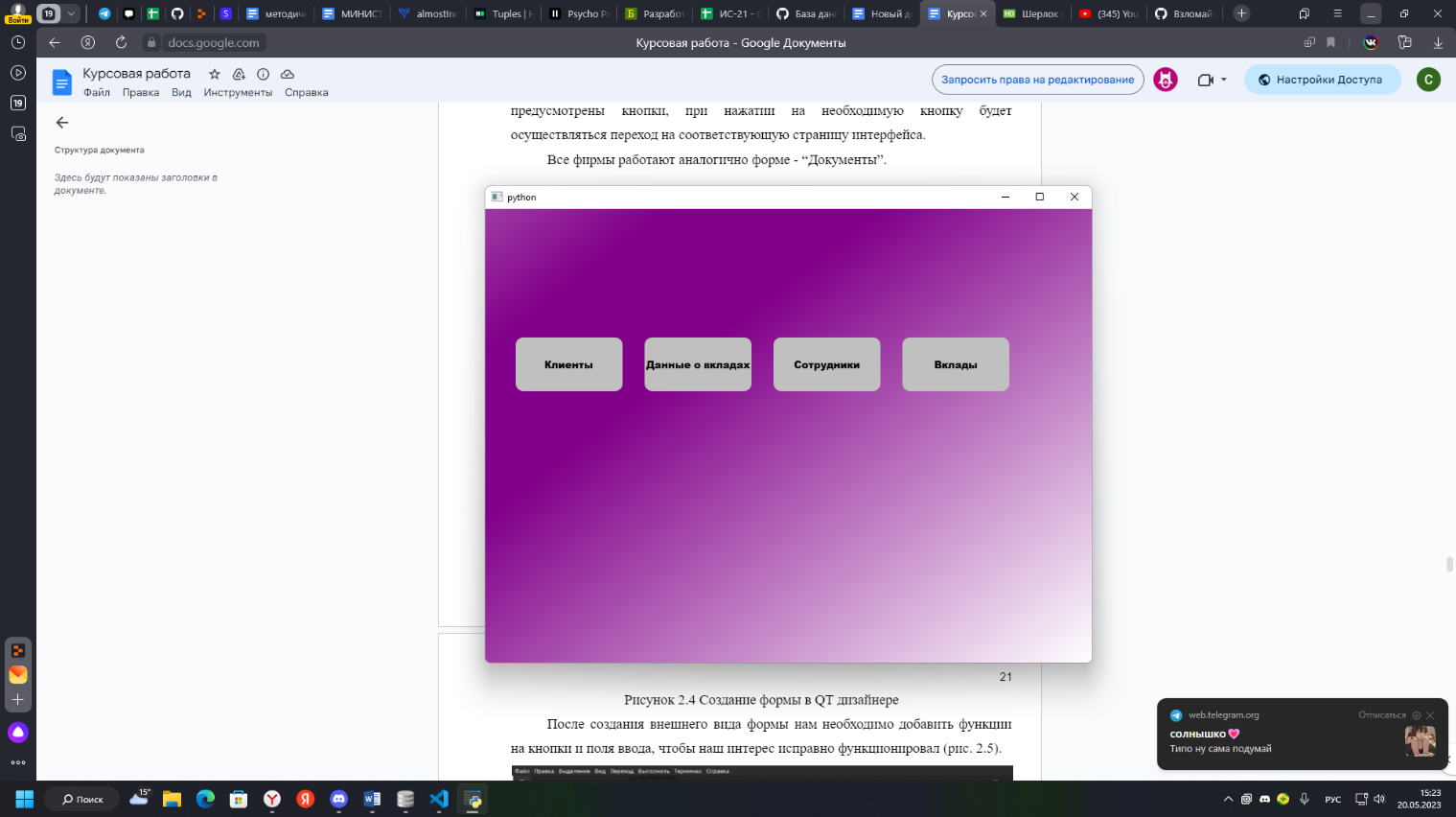


Рисунок 3. Внешний вид форм

После создания внешнего вида формы нам необходимо добавить функции на кнопки и поля ввода, чтобы наш интерес исправно функционировал.

## 

## **2.8 Разбор кода**

Создание класса и вызов функций (Рисунок 5.)

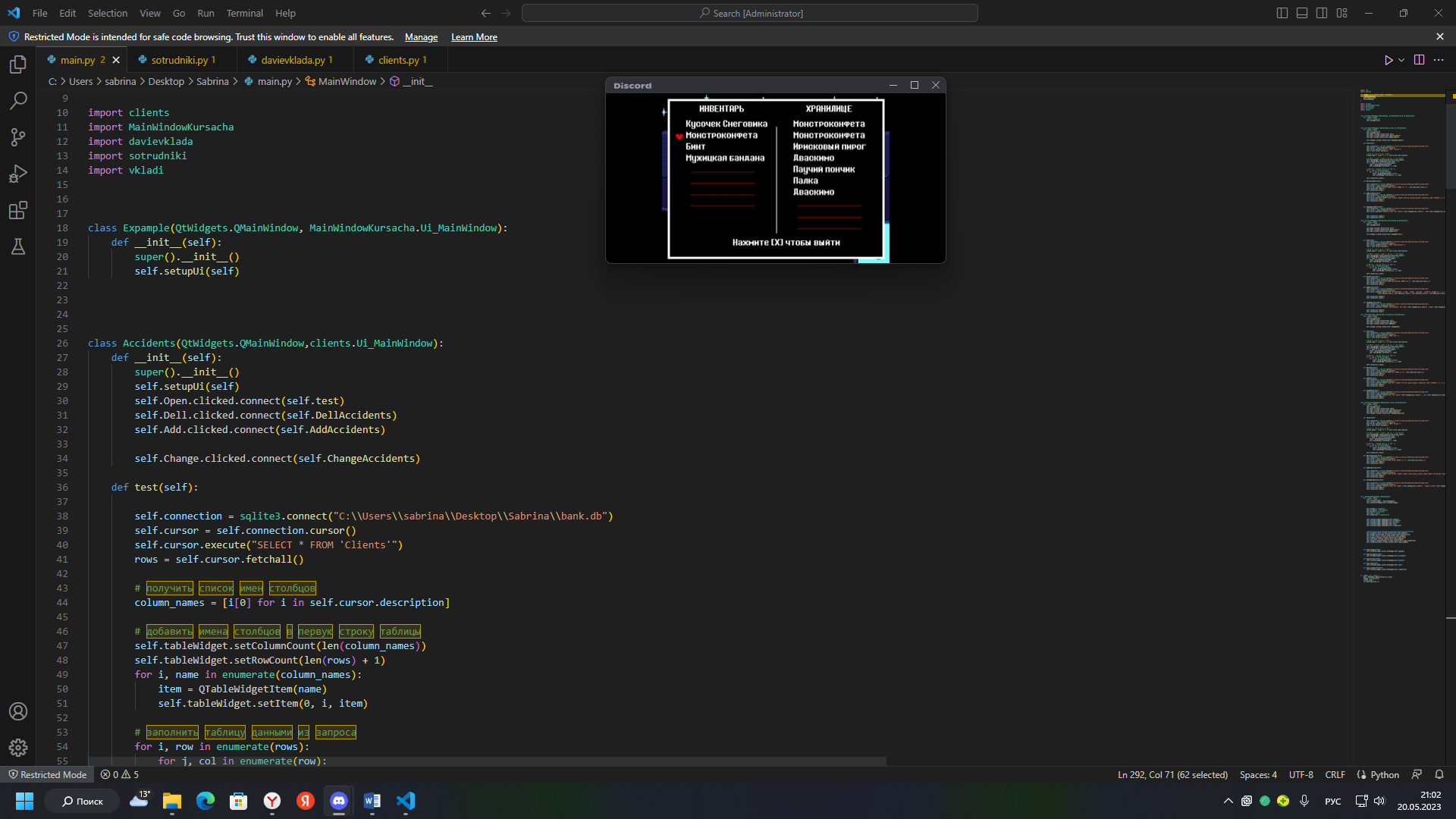


Рисунок 5. Создание класса и вызов функции.

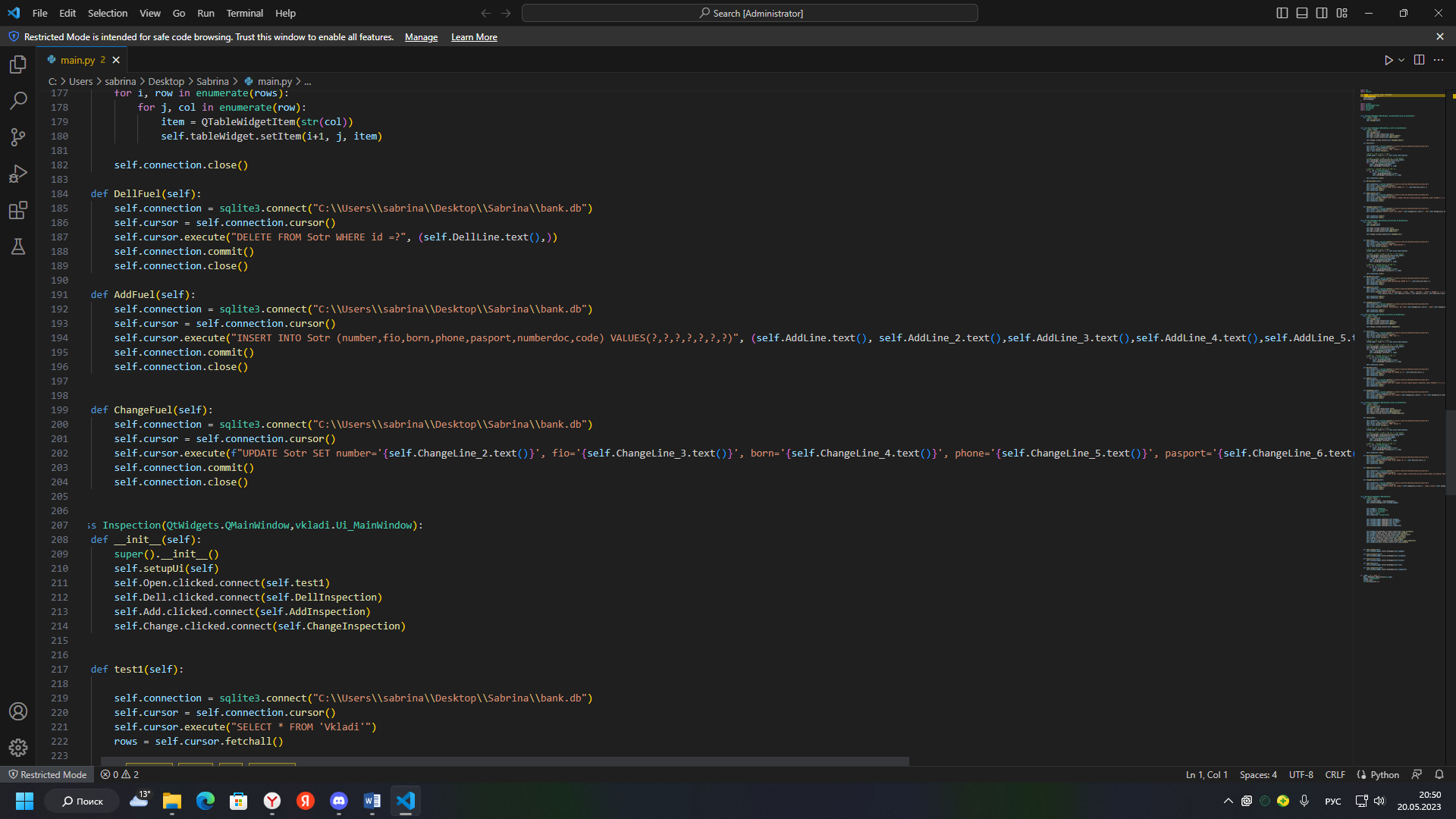


Рисунок 6. Удаление данных в таблице через форму базы данных.

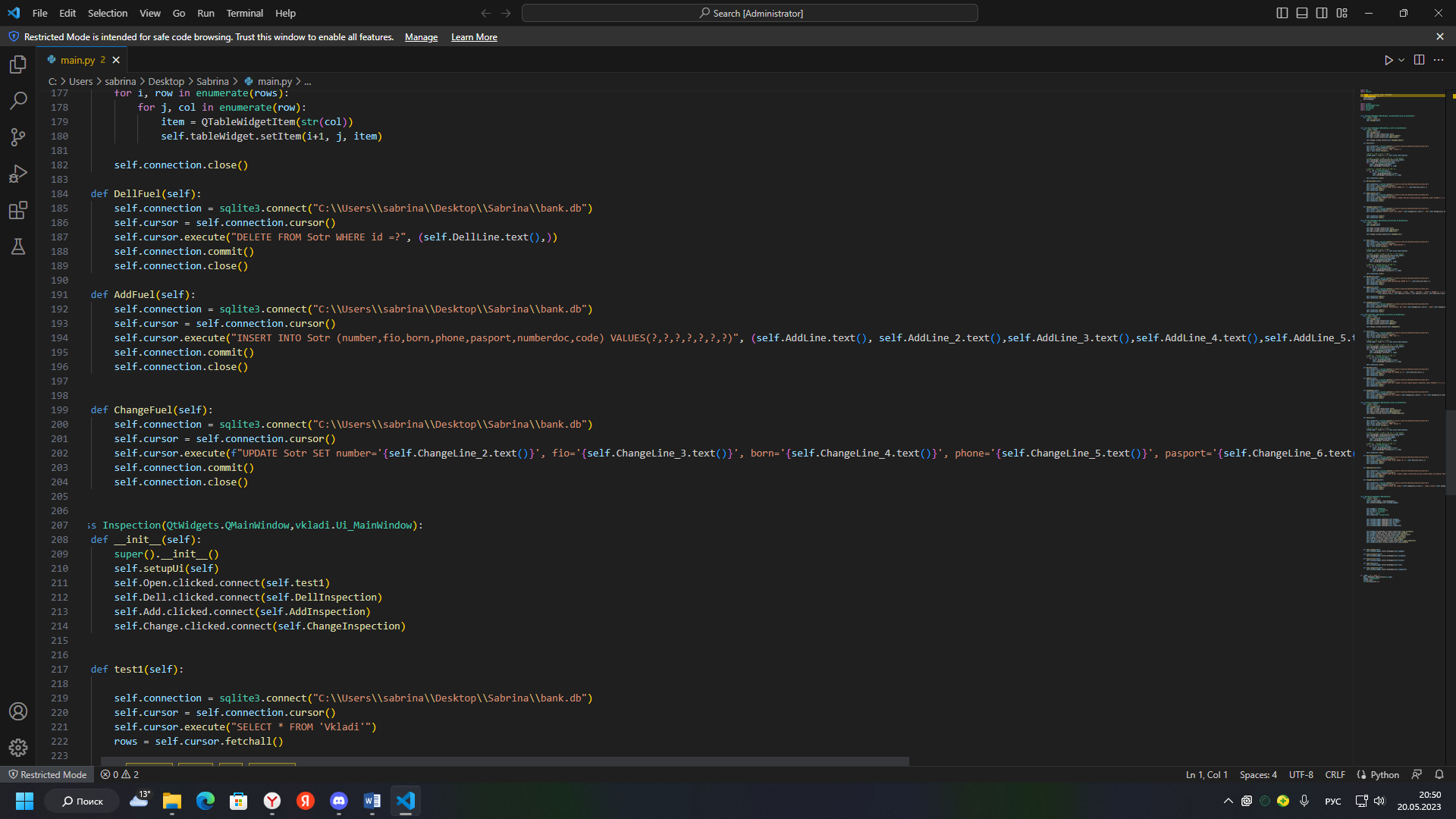


Рисунок 7. Добавление данных в таблице через форму базы данных.

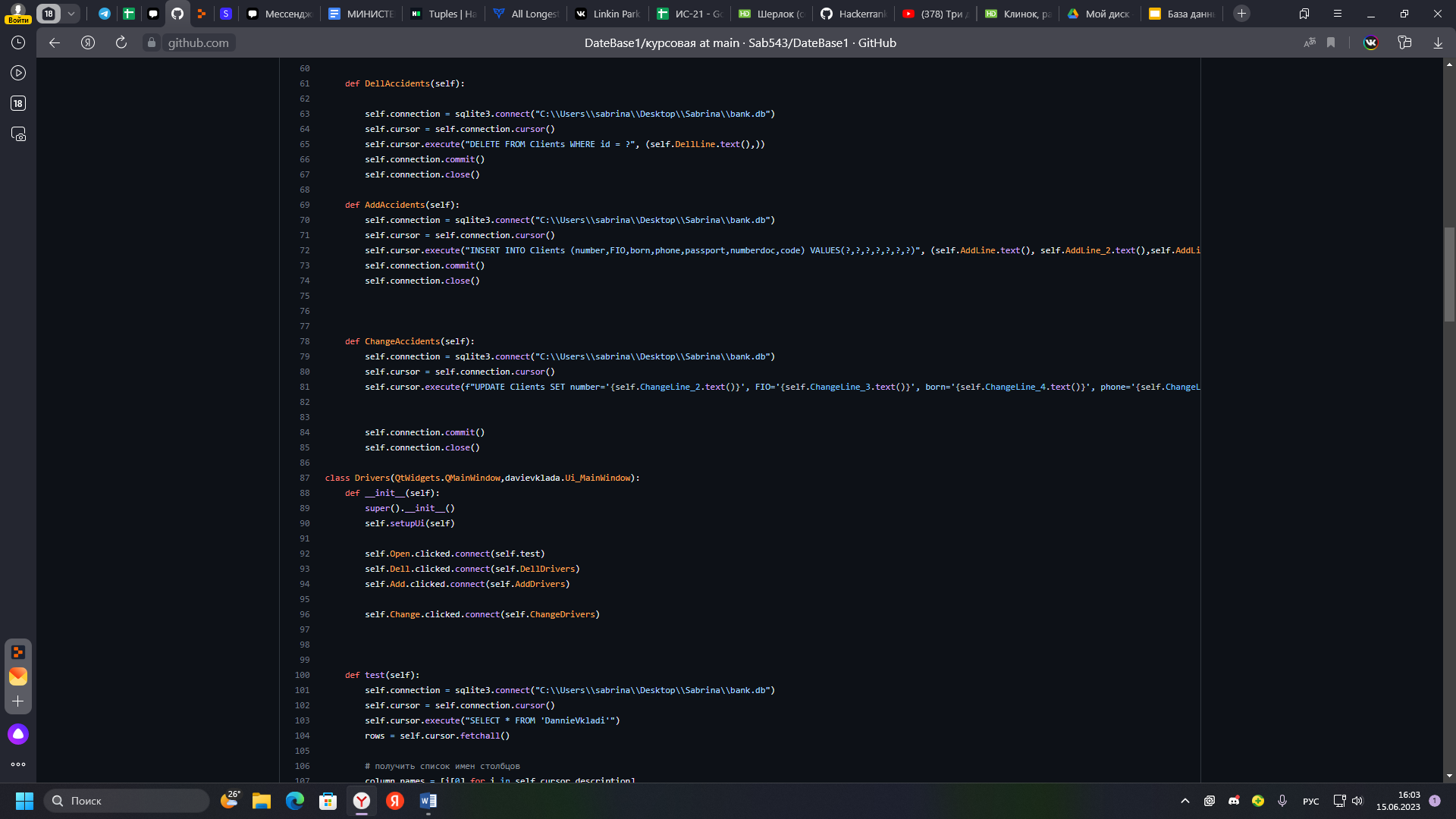


Рисунок 8. Изменение данных в таблице через форму базы данных.

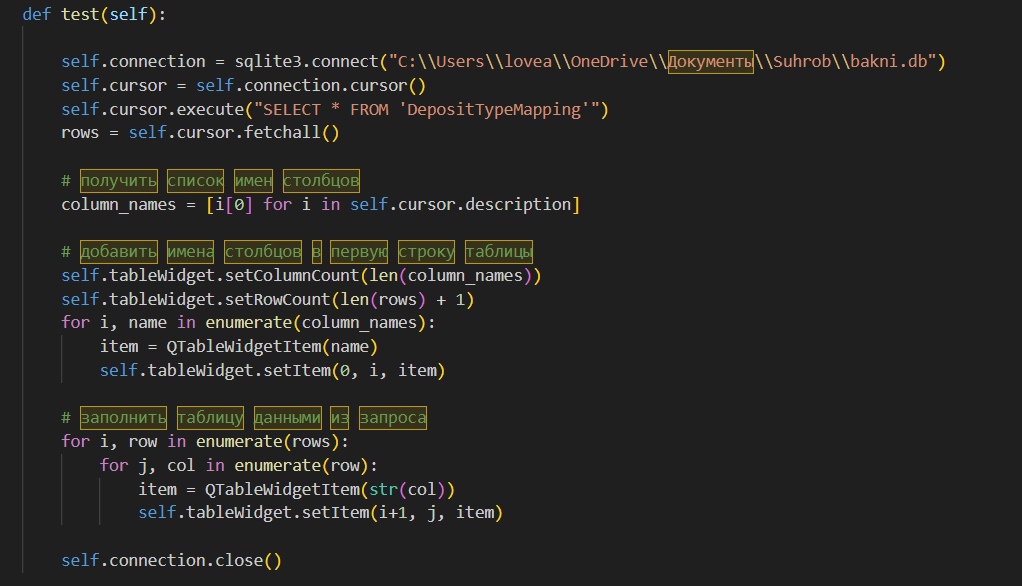


Рисунок 9. Вывод данных из таблицы на форму (открытие)

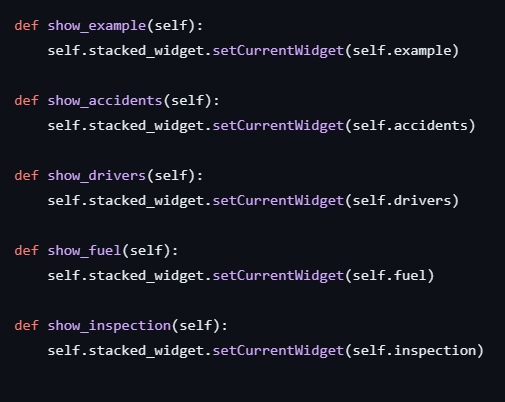


Рисунок 8. Функции перехода между формами

Все формы работают на основе кода, который показан выше, изменения происходят только в SQL запросах.

# **ЗАКЛЮЧЕНИЕ**

Цельюкурсового проектированияявляласьразработка базы данных (БД) автоматизированного учета займов и вкладов в коммерческом банке, сопровождающих управленческую деятельность и клиентский учёт для создания более комфортных условий работы сотрудников коммерческого банка.

Для достижения поставленной цели были решены следующие задачи:

* была создана ER-диаграмма по техническому заданию (в Draw SQL);
* была создана база данных по ER-диаграмме (в SQLite);
* БД была заполнена тестовыми данными;
* была создана авторизация (в Qt5 дизайнер);
* разработаны формы для автоматизированного заполнения данными БД (в Qt5 дизайнер);
* подключены формы к базе данных (в Visual Studio Code)
* были протестированы на работоспособность созданные формы.

# **СПИСОК ИСТОЧНИКОВ**

1. «Системы управления базами данных» К. Дейт;

<https://djvu.online/file/xvtymREmD4CKu?ysclid=lhbl0vsqb2884376138>

(16 апреля 2023)

1. «Реляционные базы данных»

<https://cloud.yandex.ru/docs/glossary/relational-databases>

(1 мая 2023)

1. «Проектирование базы данных»

<https://studfile.net/preview/2805563/page:9/>

(7 мая 2023)

1. «Официальная документация Python»

<https://docs.python.org/3/tutorial/index.html>

(19 мая 2023)