Департамент образования и науки города Москвы

Государственное бюджетное профессиональное образовательное учреждение

города Москвы

«Колледж малого бизнеса № 4»

(ГБПОУ КМБ № 4)

**КУРСОВОЙ ПРОЕКТ**

по МДК05.02 Разработка кода информационных систем, МДК 05.03 Тестирование информационных систем

для специальности09.02.07 Информационные системы и программирование

Базовая подготовка

Тема: Разработка информационной системы для учета выданных книг в библиотеке

Выполнил(а) студент(ка)

2 курса группы № ИПС-21.22

Павликова Варвара Павловна

(Фамилия, имя, отчество студента)

Проверил

руководитель курсовой работы

Минеева Кристина Константиновна

Работа защищена с оценкой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024

Москва, 2024

**СОДЕРЖАНИЕ**

**Оглавление**

[**ВВЕДЕНИЕ** 3](#_Toc167365855)

[**ГЛАВА 1. Теоретическая часть программного обеспечения** 5](#_Toc167365856)

[**1.1.** **Понятие, актуальные виды и особенности программного обеспечения** 5](#_Toc167365857)

[**1.2.** **Понятие, актуальные виды и особенности информационных систем** 7](#_Toc167365858)

[**1.3.** **Особенности программных средств разработки и программирования** 10](#_Toc167365859)

[**1.4.** **Анализ предметной области** 12](#_Toc167365860)

[**1.5.** **Определение Visual Studio Code** 15](#_Toc167365861)

[**1.6.** **Функции и особенности приложения Visual Studio Code** 16](#_Toc167365862)

[**ГЛАВА 2. Практическая часть создания проекта** 20](#_Toc167365863)

[**2.1.** **Обоснование выбора программы для разработки информационной системы** 20](#_Toc167365864)

[**2.2.** **Почему Visual Studio Code. Анализ приложений конкурентов** 20](#_Toc167365865)

[**2.3.** **Определение языков программирования, используемых в проекте** 23](#_Toc167365866)

[**2.4.** **Определение выбора метода для разработки** 24](#_Toc167365867)

[**2.5.** **Поэтапная разработка информационной системы** 27](#_Toc167365868)

[**2.6.** **Итоговая версия десктоп-приложения** 46](#_Toc167365869)

[**ЗАКЛЮЧЕНИЕ** 47](#_Toc167365870)

[**ИСПОЛЬЗУЕМАЯ** **ЛИТЕРАТУРА** 48](#_Toc167365871)

[**ПРИЛОЖЕНИЕ** 50](#_Toc167365872)

### **ВВЕДЕНИЕ**

В современном информационном обществе эффективное управление информацией становится все более важным в различных областях деятельности. Одной из таких областей является библиотечное дело, где необходимость в систематизации и учете выданных книг становится критической для эффективного функционирования. В данном контексте представляется актуальной задача разработки информационной системы для учета выданных книг в библиотеке.

Такая информационная система будет востребована в любой библиотеке, будь то общественная, учебная или специализированная. Она обеспечит автоматизацию процесса учета и контроля за движением книжного фонда, сократит время, затрачиваемое на административные процедуры, и повысит качество обслуживания пользователей.

Актуальность данного исследования подчеркивается не только ростом количества книг и читателей в современных библиотеках, но и необходимостью внедрения современных информационных технологий для оптимизации работы учреждений культуры.

Новизна данной работы заключается в разработке индивидуальной информационной системы, адаптированной к конкретным потребностям библиотеки и её аудитории. Это предполагает не только техническую реализацию, но и учет особенностей организации и работы библиотечного учреждения.

Основной целью данной курсовой работы является разработка информационной системы для учета выданных книг в библиотеке с целью оптимизации и упрощения процесса управления библиотечным фондом.

Для достижения поставленной цели необходимо рассмотреть следующие задачи:

• Изучение предметной области библиотечного дела и процесса учета книг.

• Изучение понятия «информационная система».

• Анализ требований пользователей и функциональных возможностей информационной системы.

• Выбор подходящих технологий и программных решений для реализации системы (рассмотреть язык программирования Python, программу Visual Studio Code для разработки приложения, базы данных, для внедрения информационной системы).

• Описание программы для создания информационной системы для учета выданных книг в библиотеке.

• Разработка программы для создания информационной системы для управления.

• Тестирование разработанной системы.

Практическая значимость данной работы заключается в том, что разработанная информационная система для учета выданных книг в библиотеке представляет собой инструмент, который значительно упростит и оптимизирует работу библиотечного учреждения, повысит эффективность работы библиотеки и ускорит обслуживание пользователей; повышение уровня профессионализма в программировании, результаты курсовой работы могут быть использованы как дополнительная работа к портфолио (резюме).

Выполненная работа состоит из введения, основной части из двух разделов (теоретической и практической), заключения и списка литературы.

### **ГЛАВА 1. Теоретическая часть программного обеспечения**

Данная глава представляет собой описание существующих и актуальных видов программного обеспечения, особенности программных средств разработки и программирования.

## **Понятие, актуальные виды и особенности программного обеспечения**

Программное обеспечение — программа или множество программ, используемых для управления компьютером. Имеются и другие определения из международных и российских стандартов: совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ.

Цель программного обеспечения — осуществление управления электронными и механическими частями компьютера. ПО – это весьма гибкая система, имеющая возможность усовершенствования. Программный продукт, использующий информацию, превращает аппаратную часть ПК в комплекс, с помощью которого возможно решать самые разнообразные задачи.

Жизненный цикл любого компьютерного программного обеспечения состоит из трех основных стадий: написание, работа и поддержка.

Сам процесс написания условно подразделяется на 6 этапов:

* Постановка задачи.
* Создание концепции.
* Выработка алгоритма.
* Формирование и компиляция.
* Выявление ошибок и отладка.
* Завершение процедур по оформлению документации.

Особенности ПО:

* Практически любым процессом можно управлять определённым набором инструкций.
* Затруднение разработки состоит в сложности его оцифровки.
* Создание нового ПО осуществляется на ПК с установленным программным обеспечением.
* Написание кода менее трудоёмкая операция, чем его отладка и настройка управления.
* Программное обеспечение пользователя является методом реализации поставленных задач.

Можно выделить три основных вида программного обеспечение:

* Системное программное обеспечение:

Это комплекс программ, которые обеспечивают управление компонентами компьютерной системы, такими как процессор, оперативная память, устройства ввода-вывода, сетевое оборудование, выступая как «межслойный интерфейс», с одной стороны которого аппаратура, а с другой — приложения пользователя. Системное программное обеспечение управляет всеми ресурсами ЭВМ (центральным процессором, памятью, вводом-выводом) и реализовывает общую организацию процесса обработки информации и интерфейсы между ЭВМ, пользователем, аппаратными и программными средствами. Оно разрабатывается так, чтобы компьютер мог эффективно выполнять прикладные программы.

* Прикладное программное обеспечение:

Это программы, предназначенные для решения задач пользователя. Их также называют приложениями. Этот тип используемого программного обеспечения обладает огромным разнообразием. К таким приложениям можно отнести: универсальное ПО, специализированные программы, программное обеспечение для развлечения. В качестве примеров прикладного программного обеспечения можно выделить следующие: текстовые, графические, видео- и аудиоредакторы, веб-обозреватели, антивирусные пакеты, комплексы для проектирования в различных профессиональных областях, проигрыватели, архиваторы и огромное множество других.

* Инструментальное программное обеспечение:

Оно необходимо при проектировании приложений. Такой вид ПО предназначен для разработчиков софта. Среди них: линковщики, компиляторы, трансляторы, сборники подпрограмм и пр. (Например, Turbo Paskal, Borland C++, Embarcadero Delphi, Microsoft Visual Studio.)

## **Понятие, актуальные виды и особенности информационных систем**

Понятие информационной системы:

Информационная система (ИС) — это совокупность взаимосвязанных элементов, включающая в себя людей, процессы, данные и технологии, направленных на сбор, хранение, обработку и передачу информации для достижения определенных целей. ИС включает в себя как технические устройства (аппаратное обеспечение), так и программное обеспечение, данные, процедуры и людей, которые взаимодействуют с данным комплексом.

Информационная система представляет собой хранилище информации, снабженное процедурами ввода, поиска, размещения и выдачи информации. Наличие таких процедур – главная особенность информационных систем, отличающих их от простых скоплений информационных материалов.  
Информационная система определяется следующими свойствами:

- любая информационная система может быть подвергнута анализу, построена и управляема на основе общих принципов построения систем;

- информационная система является динамичной и развивающейся;

- при построении информационной системы необходимо использовать системный подход;

- выходной продукцией информационной системы является информация, на основе которой принимаются решения;

- информационную систему следует воспринимать как человекокомпьютерную систему обработки информации.

Процессы, обеспечивающие работу информационной системы любого назначения, условно можно представить в виде схемы (Рисунок 1), состоящей из блоков:

- ввод информации из внешних или внутренних источников;

- обработка входной информации и представление ее в удобном виде;

- вывод информации для представления потребителям или передачи в другую систему;

- обратная связь – это информация, переработанная людьми данной организации для коррекции входной информации.

На рисунке 1 изображены процессы в информационной системе.

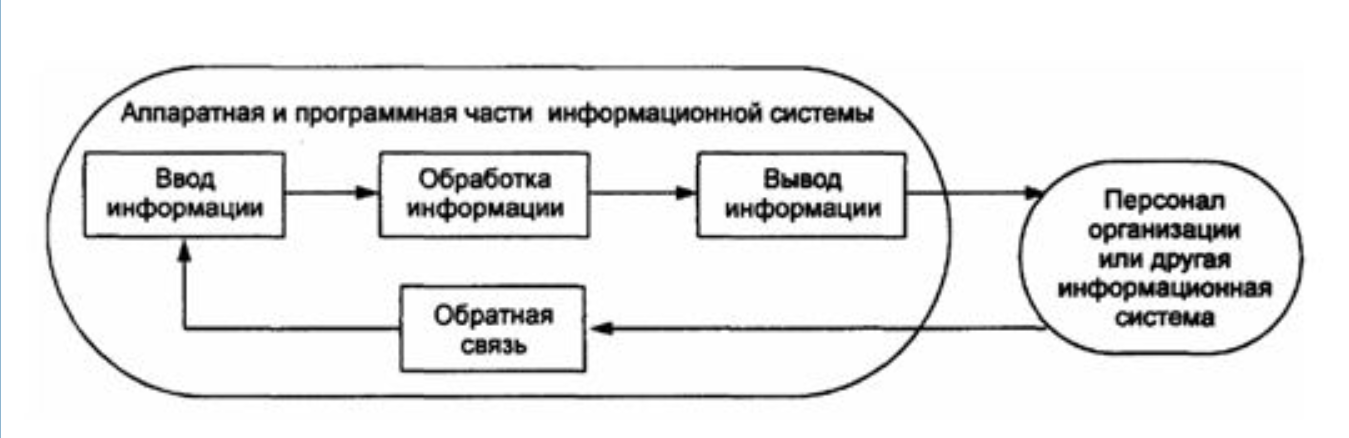


Рисунок 1. Процессы в информационной системе

Работа информационных систем заключается в обслуживании двух встречных потоков информации: ввода новой информации и выдачи текущей информации по запросам. Поскольку главная задача информационной системы: обслуживание клиентов, система должна быть устроена так, чтобы ответ на любой запрос выдавался быстро и был достаточно полным. Эти требования обеспечиваются наличием стандартных процедур поиска информации и тем, что данные системы расположены в определенном порядке.

Типы существующих информационных систем:

1. Информационная система транзакций (ИСТ):

- Предназначена для автоматизации повседневных операций, таких как обработка заказов, начисление заработной платы и учет финансовых транзакций.

- Примеры: система управления заказами, кассовая система в розничной торговле.

2. Информационная система управления (ИСУ):

- Используется для управления организацией и предоставления администраторам доступа к отчетам и данным, необходимым для принятия решений.

- Примеры: система управления персоналом, финансовая система.

3. Система поддержки принятия решений (СППР):

- Поддерживает принятие управленческих решений путем анализа данных и создания модели принятия решений.

- Примеры: система прогнозирования продаж, система анализа инвестиционных рисков.

4. Система выполнения стратегических задач (СВСЗ):

- Ориентирована на высшее руководство компании и предоставляет данные для стратегического планирования и анализа.

- Примеры: система стратегического планирования, система балансовых показателей.

5. Экспертная система (ЭС):

- Использует знания и правила для моделирования работы профессионалов и решения сложных задач в конкретной области.

- Примеры: медицинские диагностические системы, системы технической диагностики оборудования.

6. Система управления контентом (СУК):

- Предназначена для создания, управления и публикации контента на веб-сайтах и других цифровых носителях.

- Примеры: wordpress, joomla, drupal.

7. Система планирования ресурсов предприятия (ERP):

- Интегрирует все аспекты вашего бизнеса в единую систему, включая планирование, производство, продажи и финансы.

- Примеры: sap, oracle erp, microsoft dynamics.

8. Система управления взаимоотношениями с клиентами (CRM):

- Улучшает обслуживание клиентов, увеличивает продажи и помогает управлять взаимодействием с текущими и потенциальными клиентами.

- Пример: salesforce, hubspot, zoho crm.

Информационные системы играют важную роль в современной экономике, поддерживая все уровни управления и повышая производительность организации.

## **Особенности программных средств разработки и программирования**

Средства разработки программного обеспечения — это совокупность методов, методик, приёмов и набора инструментальных программ (компиляторы, прикладные/системные библиотеки и т.п.), используемых разработчиком для создания программного кода программы, соответствующего установленным требованиям.

С учётом этого определения термин «Разработка программ» будет сформулирован следующим образом:

Разработка программ — это процесс, требующий значительных усилий и направленный на создание и поддержку программного кода, обеспечивающего нужный уровень качества и надежности. Для этого используются программные средства разработки. Важно отметить, что, хотя условия задачи могут быть неформальными, объектом разработки являются формализованные программы, что требует определенного уровня формализации на различных этапах разработки. Процесс разработки программ носит креативный характер и требует принятия решений на каждом этапе. Продукт разработки представляет собой комплекс текстовых и динамических объектов, которые обрабатывают данные и реагируют на действия пользователей. При этом программы не расходуют используемые ресурсы в процессе работы.

Программирование — это создание компьютерных программ, которые представляют собой последовательность команд, написанных на определенном языке программирования. Каждое слово, функция и оператор в коде имеют строго определенное значение, которое понимает компьютер.

Язык программирования — это технический язык, имеющий свою лексику, семантику и синтаксис. Операторы являются основными элементами языков программирования, которые предписывают компьютеру, какие операции выполнять. Хотя существует большое количество различных языков программирования, они обычно похожи друг на друга по синтаксису и основным конструкциям.

Язык программирования выбирается в зависимости от требований проекта, целевой платформы и задач. Например, Python подходит для быстрой разработки и прототипирования, тогда как C++ используется для системного программирования и разработки высокопроизводительных приложений. Библиотеки и фреймворки упрощают разработку, предоставляя готовые решения для распространенных задач, что позволяет сосредоточиться на логике приложения, не тратя время на написание базовой функциональности с нуля. Базы данных обычно создаются и управляются с использованием языка структурированных запросов SQL.

Эффективная документация кода является важным аспектом поддержки и развития проекта, обеспечивая понимание его логики и структуры другими разработчиками. Использование автоматизированных инструментов для создания и поддержания документации упрощает этот процесс. Оптимизация производительности включает в себя профилирование кода для выявления узких мест и применение оптимизаций для повышения скорости и эффективности работы приложения. Внедрение безопасных практик программирования, таких как проверка входных данных и шифрование конфиденциальной информации, помогает защитить приложения от уязвимостей и атак.

Для совместной разработки необходимо использование систем контроля версий и инструментов для управления задачами и коммуникации. Это способствует согласованной работе команды и обеспечивает прозрачность в процессах разработки. После выпуска программного продукта важно обеспечить его поддержку и регулярные обновления, включая исправление ошибок, добавление новой функциональности и адаптацию к требованиям пользователей и рынка. Эти факторы играют важную роль в успешной разработке, обеспечивая высокое качество и надежность программного обеспечения.

Программные средства разработки и программирования играют ключевую роль в процессе создания высококачественного программного обеспечения, обеспечивая разработчикам эффективные инструменты и средства для реализации их идей и проектов.

## **Анализ предметной области**

Анализ предметной области библиотечного дела позволяет лучше понять его значимость для общества, выявить потенциальные направления развития и улучшения, а также разработать стратегии для улучшения доступа к информации и повышения эффективности работы библиотек.

Библиотека – это учреждение культуры, организующее сбор, хранение и общественное пользование произведениями печати и другими документами. Библиотеки систематически занимаются сбором, хранением, пропагандой и выдачей читателям произведений печати, а также информационно-библиографической работой, являются общедоступным источником знаний и основной базой для самообразования.

Основными направлениями работы любой библиотеки являются: комплектование и организация книжного фонда; обслуживание читателей.

Комплектование фондов библиотеки состоит из систематического выявления (путем просмотра библиографических источников и литературы) нужных для данной библиотеки изданий и приобретения их. От своевременности и полноты комплектования библиотеки в значительной мере зависит уровень обслуживания читателей.

Организация книжного фонда включает вопросы учета, расстановки, хранения литературы и доставки ее читателю.

Учет библиотечного фонда — это прием поступивших в библиотеку изданий, запись прибывших и исключение по тем или иным причинам непригодных для читателей изданий в документах библиотеки, которые дают точные сведения о наличном составе фонда и помогают обеспечить его сохранность.

Правильная организация фонда облегчает читателю пользование литературой, библиотекарю - быстрое выполнение читательских требований, а также обеспечивает сохранность фондов как общественной собственности.

Обслуживание читателей библиотеки осуществляется различными путём выдачи литературы и помощи отдельным читателям в подборе необходимой им литературы.

Читатели, приходящие в библиотеку, обязаны иметь при себе читательский билет. При выбытии из вуза (отчисление, окончание обучения, увольнение) читатели обязаны вернуть числящиеся за ними издания и сдать читательские билеты. За нарушение правил пользования библиотекой читатели лишаются права пользования всеми пунктами обслуживания библиотеки на установленные администрацией сроки (1 месяц).

Срок пользования литературой для различных категорий читателей и количество выдаваемых изданий на каждом абонементе определяется администрацией, исходя из вида литературы и категории читателя. Число книг, выдаваемых в читальных залах, не ограничивается.

Библиотечная деятельность связана с учетом большого количества операций, множество книг и читателей серьезно замедляют работу библиотекарей. Сложность поиска нужной книги в каталоге занимает длительное время, и целиком опирается на компетентность работников библиотеки.

Для ведения библиотечных каталогов, организации поиска требуемых изданий и библиотечной статистики в базе должны храниться сведения, большая часть которых размещаются в аннотированных каталожных карточках.

В библиотеку приходит много книг из различных издательств. Каждая книга, хранящаяся в библиотеке, имеет следующие параметры:

* название книги;
* уникальный шифр (ISBN);
* автор;
* издательство;
* год издания.

Каждая книга может присутствовать в нескольких экземплярах.

Все книги различаются по своему уникальному шифру - ISBN.

Библиотечно-библиографическая классификация (ББК) распределяет издания по отраслям знания в соответствии с их содержанием. В ней используется цифробуквенные индексы ступенчатой структуры (например, ББК 32.973 Электронные вычислительные машины и устройства). Шифр ББК используется при выделении хранимым изданиям определенных комнат, стеллажей и полок, а также для составления каталогов и статистических отчетов. В библиотеке имеется база данных (БД). Она представляет собой совокупность данных обо всех происходящих процессах в библиотеке (книгах, читателях, проведенных мероприятиях и др.), построенную в хронологической последовательности, в виде таблиц и списков данных. Основное ее назначение - хранение данных, для их последующего целесообразного использования в будущем. БД находится на специальном компьютере в электронном виде, постоянно редактируется и изменяется работниками библиотеки. С базой данных библиотеки могут работать как руководители (администрация) библиотеки, так и работники библиотеки, заинтересованные в получении необходимой информации.

При работе с системой библиотекарь должен иметь возможность решать следующие задачи:

* принимать новые книги и регистрировать их в библиотеке;
* относить книги к одной или к нескольким областям знаний;
* проводить каталогизацию книг, то есть назначение новых инвентарных номеров вновь принятым книгам;
* вести учет выданных книг читателям, при этом предполагается два режима работы: выдача книг читателю и прием от него возвращаемых им книг обратно в библиотеку. При выдаче книг фиксируется, когда и какая книга была выдана данному читателю и на какой срок выдается данная книга. При приеме книги, возвращаемой читателем, проверяется соответствие возвращаемого инвентарного номера книги выданному инвентарному номеру, название книги, и она ставится на свое старое место в библиотеке. Администрация библиотеки должна иметь возможность получать сведения о должниках - читателях библиотеки, которые не вернули вовремя взятые книги.

В библиотеке ведется картотека читателей. О каждом читателе заносятся следующие сведения:

* ФИО;
* номер паспорта;
* адрес.

Каждому читателю присваивается номер читательского билета. Кроме того, учитываются сведения о наличии у читателя штрафов.

В случае выдачи экземпляра книги в библиотеке остается вкладыш, в котором указана дата выдачи, дата предполагаемого возврата и номер читательского билета.

При возврате книги во вкладыше отмечается срок возврата. При просрочке возврата книги, читатель получает предупреждение. Когда предупреждений у читателя накапливается больше определенного предела, он лишается правом пользования библиотекой на определенный срок.

## **Определение Visual Studio Code**

Visual Studio Code (VS Code) — это свободный и открытый исходный код редактор текста, разработанный компанией Microsoft. Он предназначен для разработки программного обеспечения и поддерживает множество языков программирования, включая JavaScript, TypeScript, Python, C++, Java, и многие другие. Visual Studio Code обладает широким набором функций, включая подсветку синтаксиса, автоматическое дополнение кода, интеграцию с системами управления версиями, встроенный терминал, отладчик и поддержку расширений, что делает его мощным и гибким инструментом для разработки программного обеспечения. Он доступен для различных операционных систем, включая Windows, macOS и Linux.

Visual Studio Code основан на языке программирования Electron и реализуется через веб-редактор Monaco.

VS Code позволяет легко писать, форматировать и редактировать код на разных языках. С его помощью можно быстро создать проект и структуру файлов в нем, он подсвечивает синтаксис кода и помогает автоматически править ошибки. В нем есть возможности для отладки и запуска кода на некоторых языках. Это делает процесс поиска и исправления ошибок в коде более эффективным и удобным.

Редактор легко расширяется, поэтому к перечисленным функциям можно добавить новые — достаточно просто скачать нужное дополнение из официального каталога. Дополнения тоже распространяются бесплатно.

Visual Studio Code также имеет встроенный терминал, что позволяет разработчикам выполнять командные строки без необходимости переключаться между окнами. Это особенно удобно для выполнения задач сборки, запуска тестов и управления зависимостями.

На рисунке 2 изображено основное меню редактора.

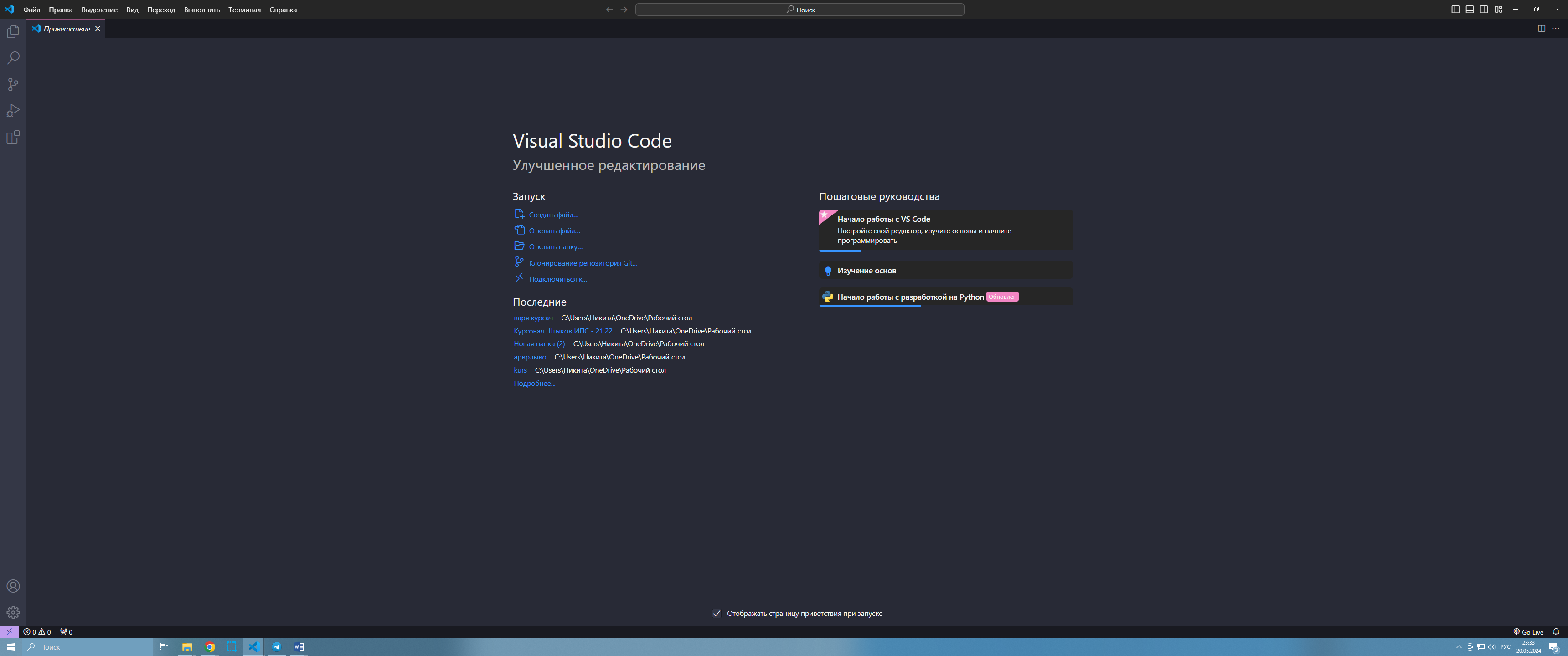


Рисунок 2. Главное меню редактора Visual Studio Code

## **Функции и особенности приложения Visual Studio Code**

Visual Studio Code — это приложение, которое умеет многое. Оно поддерживает:

1. Написание и редактирование кода приложения. Это основное предназначение бесплатного продукта от MS. Пользователи смогут не только написать, но и отредактировать, а также сохранить код. Редактор оснащен функциями корректировки. Он автоматически расставляет в коде отступы для повышения его читаемости. Предусматривает автодополнение, что помогает ускорить разработку программы, особенно на первых порах. Visual Studio Code – это продукт, в котором имеется целая система помощи редактирования исходного кода. Она называется IntelliSense.
2. Автоматическое редактирование. Полезная функция, позволяющая автоматизировать исправление и корректировку исходного кода приложения. Реализовывается через специальные плагины и расширения. С их помощью в Visual Studio Code будут исправлены некоторые незначительные ошибки: неправильные отступы, именование, несоответствие стиля и так далее. Для корректировки сложных неполадок не расширения не подойдут – система не поймет, как все исправить. Но для исправления небольших недочетов у VS Code специальные возможности есть. И работают они очень хорошо.
3. Быструю навигацию. У Visual Studio Code есть строка поиска с поддержкой регулярных выражений. А еще – автодополнение. Также редактор позволяет выделить код и закомментировать его при помощи простого сочетания клавиш.
4. Горячие клавиши. Используются для быстрого применения функций без предварительного входа в меню редактора. Можно воспользоваться встроенными сочетаниями Visual Studio Code или настроить их самостоятельно.
5. Контроль версий. Visual Studio Code позволяет сразу из редактора провести откат, добавить комментарий, отправить коммит в удаленный репозиторий. В консоль для этого заходить не придется, как и набирать те или иные команды – все осуществляется через встроенный визуальный интерфейс.
6. Установку дополнений. Это – ключевая особенность Visual Studio Code. Из редактора можно перейти в каталог дополнение и расширений, подобрать необходимый плагин и установить его в несколько кликов. Пример – поддержка различных языков, автоматическое исправление, конфигураторы. Данная опция дает возможность настройки редактора под нужды каждого конкретного разработчика.
7. Отладчик. Он используется для языка JavaScript и базирующихся на нем технологий. Пример – TypeScript. Для остальных языков разработки встроенных отладчиков нет – они устанавливаются в виде плагинов. Интерактивная [отладка](https://otus.ru/journal/otladka-prilozheniya-chto-dolzhen-pomnit-razrabotchik/) пошагово помогает выполнять код, а также на каждом этапе отслеживать изменение данных.

Visual Studio Code позволяет запускать код. По умолчанию спектр поддерживаемых языков для этого сильно ограничен. За счет дополнений можно обеспечить возможность запуска кода на любой задействованном ЯП. В интерфейсе появятся специальные кнопки для сборки и запуска приложения. Изначально в редакторе VS предусматривается только кладка «Запуск и отладка».

К преимуществам Visual Studio Code относят следующие моменты:

1. Простое освоение. Редактор оснащен интуитивно понятным интерфейсом, а также возможностью настройки. Единственная проблема для новичков – это горячие клавиши, которые организованы необычным образом.
2. Небольшой вес.
3. Кроссплатформенность.
4. Наличие online версии. В случае ее использования для написания программного кода не придется вообще ничего устанавливать. Visual Studio Online работает через браузер.
5. Поддержка разных языков. Опция активируется за счет установки плагинов.
6. Гибкость.

Данная технология позволяет работать со строками (lines), а также приводить исходный код к единому стилю в несколько кликов. Ей легко пользоваться, но недостатки у VS Code тоже есть.

К минусам Visual Studio Code относят:

1. Ограниченную функциональность, особенно в сравнении с полноценной IDE от Майкрософта.
2. Медленную работу, особенно на старых устройствах. Связано это с тем, что редактор написан на Election.

Visual Studio Code – это простой редактор кода, который подойдет не только новичкам, но и опытным разработчикам в ситуациях, когда не требуются возможности IDE.

### **ГЛАВА 2. Практическая часть создания проекта**

Данная глава представляет собой информацию по приложению Visual Studio Code и используемых языков программирования, их особенностей, технологий и причину выбора данных языков и приложения.

## **Обоснование выбора программы для разработки информационной системы**

Как уже говорилось выше, Visual Studio Code (VS Code) является достаточно универсальной программой, обладающей большим набором функций и инструментов.

VS Code поддерживает множество языков программирования и предоставляет мощные средства для разработки, такие как подсветка синтаксиса, автодополнение кода (IntelliSense), встроенный терминал и отладчик. Благодаря этим функциям разработчики могут легко писать, тестировать и отлаживать код, что значительно ускоряет процесс разработки.

В целом, использование Visual Studio Code обеспечивает комфортное и продуктивное рабочее окружение для разработчиков, упрощая процесс написания кода и обеспечивая интеграцию с другими инструментами разработки.

## **Почему Visual Studio Code. Анализ приложений конкурентов**

Visual Studio Code (VS Code) предоставляет удобный и функциональный редактор кода, который помогает разработчикам писать код быстро и эффективно. Он имеет все необходимые инструменты, такие как подсветка синтаксиса, автодополнение, переход к определению, что облегчает работу.

Однако существуют и другие редакторы кода, такие как Sublime Text, Atom и IntelliJ IDEA. Они также популярны у разработчиков и имеют свои преимущества. Например, Sublime Text известен своей скоростью работы и большим количеством плагинов. Atom, с другой стороны, предлагает высокую степень настраиваемости и гибкость. IntelliJ IDEA известен своими мощными функциями для Java-разработки.

Тем не менее, многие разработчики предпочитают Visual Studio Code из-за его удобного интерфейса, богатых возможностей и активного сообщества пользователей и разработчиков, которое постоянно расширяет функциональность редактора через создание новых расширений и интеграцию с другими инструментами разработки.

Давайте рассмотрим преимущества и недостатки Visual Studio Code и аналогичных приложений.

Visual Studio Code:

Преимущества:

1. Бесплатный и открытый исходный код.
2. Широкие возможности настройки и расширений.
3. Интеграция с системами контроля версий и другими инструментами разработки.
4. Хорошая производительность, особенно при работе с большими проектами.

Недостатки:

1. Иногда требует больше ресурсов системы в сравнении с другими редакторами.

Sublime Text:

Преимущества:

1. Очень быстрый и легкий.
2. Большое количество плагинов и поддержка множества языков программирования.
3. Простой и интуитивно понятный интерфейс.

Недостатки:

1. Проприетарное программное обеспечение, требует покупки лицензии для полной версии.
2. Меньше настраиваемый по сравнению с VS Code.

Atom:

Преимущества:

1. Бесплатный и открытый исходный код.
2. Высокая настраиваемость и гибкость благодаря большому количеству плагинов и тем.
3. Интеграция с GitHub и другими сервисами.

Недостатки:

1. Может работать медленнее на старых компьютерах из-за высокого потребления ресурсов.

IntelliJ IDEA:

Преимущества:

1. Мощная интегрированная среда разработки, особенно для языка Java.
2. Расширенные инструменты для анализа кода, рефакторинга и отладки.
3. Высокая производительность и эффективность при работе с большими проектами.

Недостатки:

1. Тяжелее и потребляет больше ресурсов, чем более легкие редакторы кода, такие как VS Code.

В итоге выбор между этими редакторами кода зависит от индивидуальных предпочтений и требований проекта. VS Code хорош для тех, кто ценит бесплатность, гибкость и большое сообщество пользователей и разработчиков. Sublime Text и Atom подходят для тех, кто ищет быстрый и легкий редактор с большим количеством настраиваемых возможностей. IntelliJ IDEA прекрасно подходит для разработки на Java и других языках, где требуется мощная интегрированная среда разработки.

Таким образом, хотя существует много конкурирующих редакторов кода, Visual Studio Code остается одним из наиболее популярных и предпочтительных выборов среди разработчиков благодаря своей удобной среде разработки и обширным возможностям.

## **Определение языков программирования, используемых в проекте**

Python - высокоуровневый интерпретируемый язык программирования, который стал популярным средством разработки благодаря своей простоте, гибкости и мощным возможностям. Язык был разработан нидерландским программистом Гвидо ван Россумом в качестве проекта по написанию языка для обучения программированию. Официально был опубликован в феврале 1991 года. Название получил в честь британской комик-группы, любимой автором благодаря комедийному шоу «Летающий цирк Монти Пайтона».

Преимущества Python:

* Простота в изучении и использовании. Python имеет простой и понятный синтаксис, который делает его идеальным выбором для начинающих программистов. Он структурирован и легко читаем, что упрощает разработку и поддержку кода.
* Множество библиотек и фреймворков. Python имеет огромное сообщество пользователей, которое создало богатую экосистему библиотек и фреймворков для различных целей, включая научные вычисления, веб-разработку, машинное обучение, анализ данных и многое другое. Например, библиотека NumPy для работы с числовыми данными или фреймворк Django для веб-разработки.
* Переносимость. Python доступен для различных платформ, включая Windows, macOS и Linux. Это означает, что вы можете писать код на Python и запускать его на разных операционных системах без изменений.
* Широкое применение. Python используется во многих областях, таких как веб-разработка, научные исследования, автоматизация задач, разработка игр, машинное обучение и искусственный интеллект. Это делает его полезным инструментом для различных проектов и задач.
* Активное сообщество. Python имеет огромное и активное сообщество пользователей и разработчиков, которое постоянно развивает и улучшает язык, создавая новые библиотеки, фреймворки и инструменты.

Недостатки Python:

* Производительность. По сравнению с некоторыми компилируемыми языками, такими как C++ или Java, Python может быть менее производительным из-за своей интерпретируемой природы. Однако, в большинстве случаев это не является критической проблемой.
* Глобаль блокировка интерпретатора. В Python есть Global Interpreter Lock (GIL), который ограничивает одновременное выполнение потоков в многопоточных приложениях. Это может снижать эффективность многопоточных программ.
* Использование памяти. Python может потреблять больше памяти по сравнению с некоторыми другими языками из-за динамической типизации и автоматического управления памятью. Это может быть проблемой для приложений, требующих эффективного использования ресурсов.

В целом, Python — это мощный и универсальный язык программирования, который обладает множеством преимуществ и подходит для широкого спектра задач. Он идеально подходит для начинающих программистов, а также для опытных разработчиков, которые хотят создавать быстрые и эффективные приложения.

## **Определение выбора метода для разработки**

Методом для разработки проекта стоит выбрать встроенные инструменты и пользовательский интерфейс приложения Visual Studio Code, так как это оптимально подходит для создания информационной системы.

На рисунке 3 изображён интерфейс приложения Visual Studio Code:

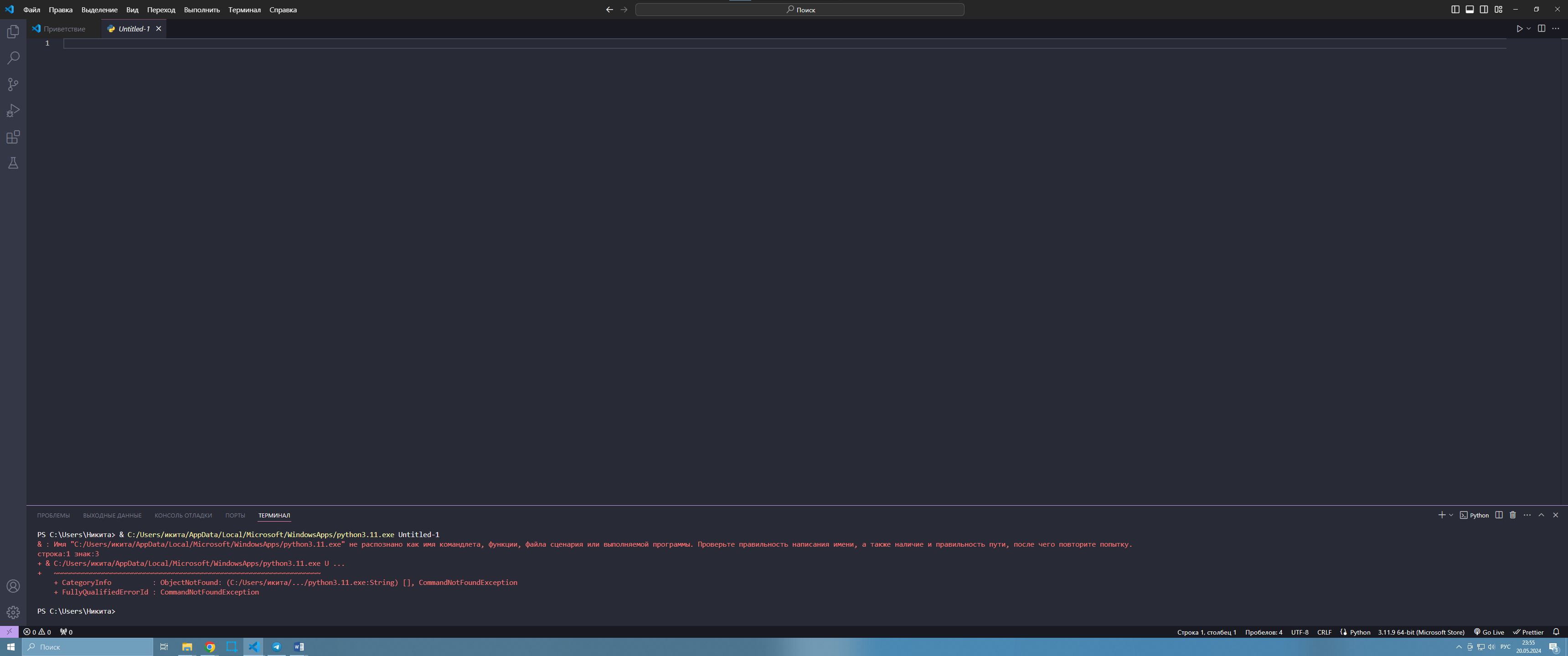


Рисунок 3. Интерфейс Visual Studio Code

На нем можем увидеть:

1. Рабочая область.

Это основная рабочая зона. Текстовый редактор, в котором пользователь создаёт свой код.

1. Терминал.

Встроенная в приложение командная строка, одновременно с этим являющийся элементом отладчика, отображая выводимый кодом текст, обнаруженные в ходе записи/работы ошибки и предупреждения, предусмотренные как стандартными исключениями и правила языка программирования, так и прописанные вручную пользователем. Так же способен получать от пользователя команды и строки, используемые в командной строке или запрашиваемые программой в ходе работы.

1. Файловый менеджер.

Встроенный файловый менеджер с стандартными для этого функциями: открытие папок, файлов, навигация по структурной иерархии папок.

1. Боковое меню.

В боковое меню входят вкладки: проводника, поиска, системы управлении версии (встроенный Git), запуска и отладки программы, расширения. При помощи установленных расширений пополняется новыми вкладками, такие как тестирование.

1. Меню учётных записей.

Позволяет пользователю связать программу с его учётной записью Microsoft или Github, что позволяет автоматически проводить резервное копирование и синхронизацию файлов в облако для работы на разных устройствах.

1. Меню настроек.

Открывает обширное меню, в котором можно настроить:  
- Палитру команд, доступных в основном меню.  
- Профили.  
- Параметры программы и рабочей области.  
- Расширения.  
- Сочетания клавиш.  
- Пользовательские фрагменты кода.  
- Задачи пользователя.  
- Темы.  
- Параметры резервного копирования и синхронизации.  
- Проверить наличие обновлений.

1. Строка состояния.

Отображает текущее состояние программы: наличие ошибок и предупреждений, используемые порты. Позволяет использовать быстрый переход в нужную строку/столбец, настроить размер отступа, кодировку, последовательность конца строки, языковой режим (языка программирования), текущую версию используемого языка программирования.

1. Основное меню.

Здесь можно создать, загрузить, сохранить, закрыть проекты, сбросить настройки, вносить правки, копировать и вставлять куски кода/строк, выделить строки, настроить внешний вид программы, перейти к нужному участку кода, запустить программе с/без режима отладки, создать терминал и перейти к встроенной справке о программе.

## **Поэтапная разработка информационной системы**

В данной сегменте будет подробно рассказано, как устроена программа.

На рисунке 4 представлен импорт необходимых библиотек для написания информационной системы.

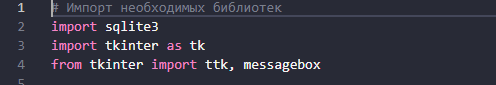


Рисунок 4. Импорт необходимых для написания ИС библиотек

Происходит импорт необходимых библиотек для работы с базой данных и графическим интерфейсом.

На рисунке 5 представлена функция для создания таблиц в базе данных.



Рисунок 5. Создание таблиц в БД

Создание функции для создания таблиц в базе данных, написание SQL кода для создания разметки таблиц, которые в будущем будут заполняться вводимыми пользователем данными.

На рисунке 6 представлен код, который выполняет сохранение изменений и закрывает соединение с БД.

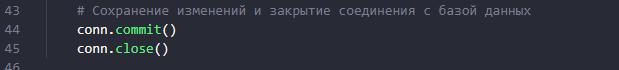


Рисунок 6. Сохранение изменений и закрытие соединения с базой данных

На рисунке 7 представлена функция для обновления информации о книгах.

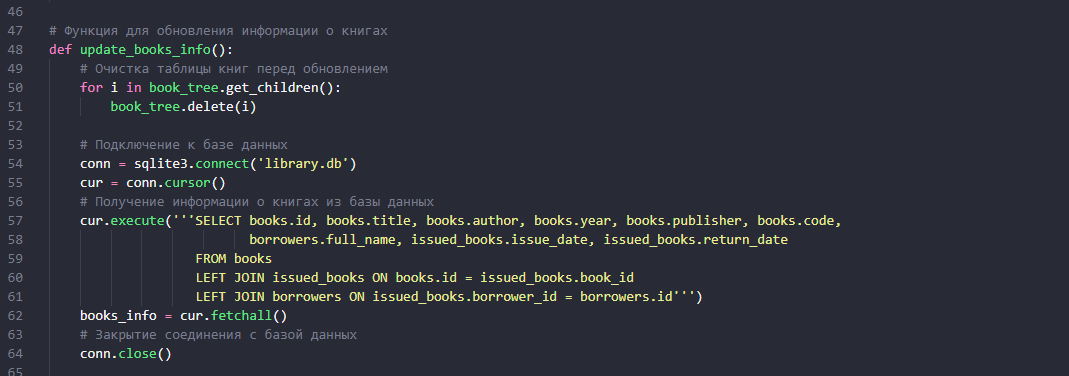


Рисунок 7. Создание функции для обновления информации о книгах

Функция, которая обновляет отображаемую информацию о книгах, отражать последние изменения в базе данных, такие как выдача новых книг или возврат существующих.

Сначала функция удаляет все строки из таблицы книг. Это гарантирует, что перед загрузкой новой информации старая информация будет удалена, чтобы избежать дублирования. Затем функция устанавливает соединение с базой данных SQLite, предположительно с именем "library.db".

Функция выполняет SQL-запрос для извлечения информации о книгах из базы данных. Запрос объединяет таблицы "books", "issued\_books" и "borrowers" по ключам и получает информацию о книгах, включая их заголовок, автора, год издания, издателя, код, а также информацию о заимствованиях и возвратах (если книга выдана).

После получения информации функция закрывает соединение с базой данных, чтобы избежать утечки ресурсов или конфликтов доступа к базе данных.

На рисунке 8 изображена функция для обновления информации о читателях.

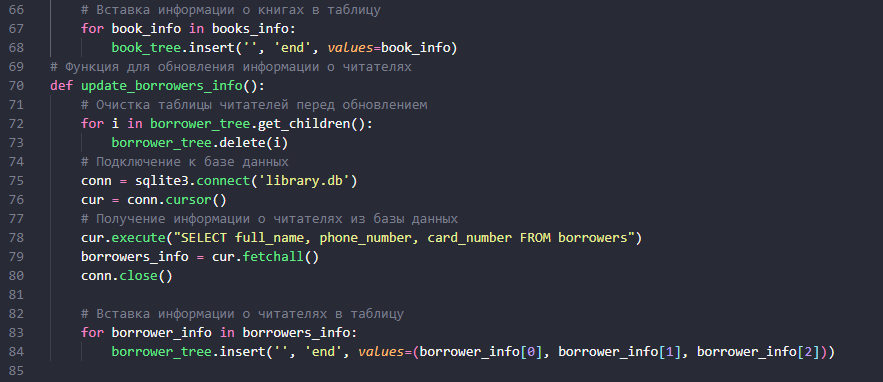


Рисунок 8. Создание функции для вставки информации о книгах в таблицу

Эта функция обновляет информацию о читателях таблице.

Подобно предыдущей функции, она начинается с удаления всех строк из таблицы читателей, чтобы избежать дублирования информации. Устанавливается соединение с базой данных SQLite с именем "library.db". SQL-запрос выполняется для извлечения информации о читателях из базы данных. Он выбирает их полные имена, номера телефонов и номера читательских билетов. После извлечения информации о читателях функция закрывает соединение с базой данных. Затем информация о каждом читателе добавляется в таблицу. Каждый элемент информации о читателе (полное имя, номер телефона, номер билета) добавляется в строку таблицы.

На рисунке 9 изображена функция для добавления книги в БД.

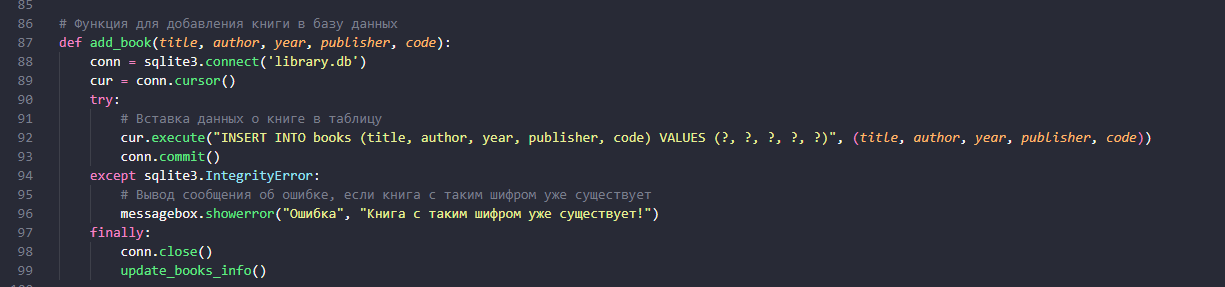


Рисунок 9. Создание функции для добавления книги в базу данных

Эта функция предназначена для добавления информации о книге в базу данных. Функция начинается с установления соединения с базой данных SQLite с именем "library.db". Затем выполняется SQL-запрос для вставки данных о книге в таблицу "books". Значения для вставки передаются в виде параметров запроса, чтобы избежать SQL-инъекций. В случае возникновения исключения sqlite3.IntegrityError, обрабатывается ситуация, когда книга с таким же шифром уже существует в базе данных. В таком случае выводится сообщение об ошибке с помощью messagebox.showerror(). После выполнения запроса и обновления информации о книгах, соединение с базой данных закрывается в блоке finally, чтобы избежать утечек ресурсов. После успешного добавления книги вызывается функция update\_books\_info(), которая обновляет информацию о книгах в интерфейсе или таблице.

На рисунке 10 изображена функция для добавления читателя в БД.

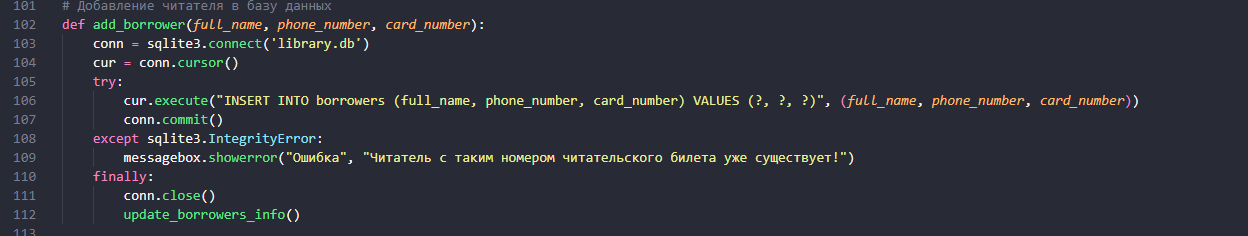


Рисунок 10. Создание функции для добавления читателей в базу данных

Выполняется аналогичная функция, что и на рисунке 9, только вместо книг выполняется добавление читателей в БД.

На рисунке 11 изображена функция для добавления возможности выдачи и возврата книг читателю.

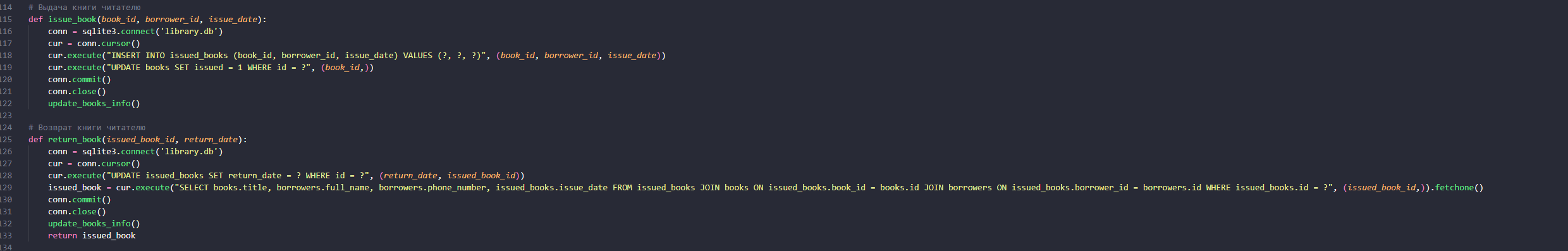


Рисунок 11. Создание функции для выдачи и возврата книг читателю

Эти две функции предназначены для управления процессом выдачи и возврата книг в библиотеке.

Функция принимает три параметра: book\_id (идентификатор книги), borrower\_id (идентификатор читателя) и issue\_date (дата выдачи).

Сначала устанавливается соединение с базой данных.

Выполняется SQL-запрос для добавления записи о выдаче книги в таблицу issued\_books. Затем обновляется статус книги, устанавливая issued в 1 (что, указывает на то, что книга выдана). После выполнения запросов соединение с базой данных закрывается, и вызывается функция update\_books\_info() для обновления информации о книгах.

Возврат книги читателю (return\_book). Эта функция принимает два параметра: issued\_book\_id (идентификатор выданной книги) и return\_date (дата возврата). Соединение с базой данных устанавливается, и выполняется SQL-запрос для обновления записи о выданной книге с указанием даты возврата. Затем получается информация о книге, которая была возвращена, для отображения в интерфейсе. Соединение с базой данных закрывается, и вызывается функция update\_books\_info(), чтобы обновить информацию о книгах.

Наконец, возвращается кортеж с информацией о книге, читателе и дате выдачи, которая, используется для отображения подтверждения возврата книги.

На рисунке 12 изображена функция для возможности удалять книг или читателей из БД.

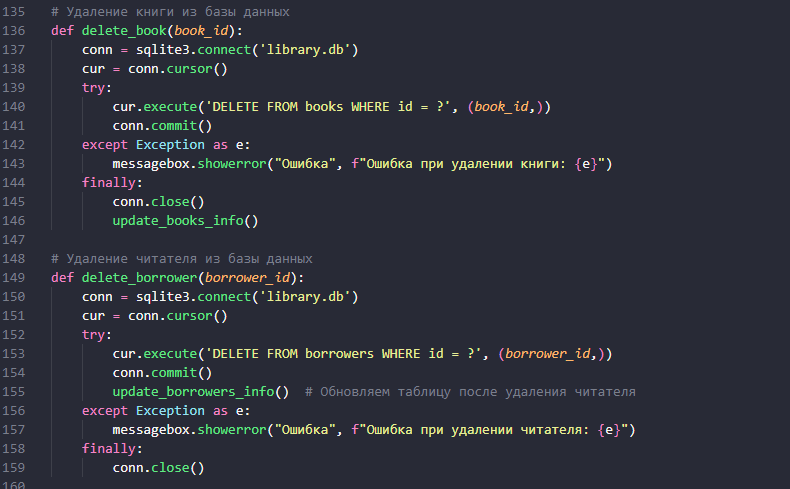


Рисунок 12. Создание функций для удаления книг/читателей

Эти функции предназначены для удаления записей о книгах и читателях из базы данных.

Удаление книги из базы данных (delete\_book): функция принимает book\_id в качестве параметра, который является идентификатором книги, которую нужно удалить. Сначала устанавливается соединение с базой данных. Выполняется SQL-запрос для удаления записи о книге из таблицы books на основе предоставленного book\_id.

Если при выполнении запроса возникает исключение, отображается сообщение об ошибке с деталями ошибки.

Независимо от того, успешно выполнен запрос или нет, соединение с базой данных закрывается, а затем вызывается функция update\_books\_info(), чтобы обновить отображаемую информацию о книгах.

Удаление читателя из базы данных (delete\_borrower): функция принимает borrower\_id в качестве параметра, который, является идентификатором читателя, которого нужно удалить. Подобно предыдущей функции, сначала устанавливается соединение с базой данных. Выполняется SQL-запрос для удаления записи о читателе из таблицы borrowers на основе предоставленного borrower\_id. После выполнения запроса информация о читателях обновляется вызовом функции update\_borrowers\_info().

В случае возникновения исключения отображается сообщение об ошибке.

Соединение с базой данных закрывается.

На рисунке 13 изображена функции для обновления доступных книг при выборе книги для выдачи и для обновления списка доступных читателей при выборе читателя для выдачи книги.



Рисунок 13. Создание функции для обновления списка доступных книг при выборе книги для выдачи и функции для обновления списка доступных читателей при выборе читателя для выдачи книги

Эти функции предназначены для обновления списков доступных книг и читателей при выборе книги для выдачи или читателя для выдачи книги соответственно.

Функция update\_book\_options: устанавливается соединение с базой данных SQLite. Выполняется SQL-запрос для получения списка книг, которые не выданы (issued = 0). Полученные книги сохраняются в переменной books. Закрывается соединение с базой данных. Создается список book\_options, содержащий только названия книг из полученных записей. Сбрасывается значение переменной book\_var. Удаляются все существующие элементы из выпадающего меню book\_menu. Для каждой книги в book\_options добавляется команда в выпадающее меню book\_menu, чтобы установить значение book\_var в выбранную книгу.

Функция update\_borrower\_options: аналогично функции update\_book\_options, устанавливается соединение с базой данных SQLite. Выполняется SQL-запрос для получения списка читателей. Полученные читатели сохраняются в переменной borrowers. Закрывается соединение с базой данных. Создается список borrower\_options, содержащий только имена читателей из полученных записей. Сбрасывается значение переменной borrower\_var (вероятно, объекта StringVar, связанного с выпадающим меню выбора читателя). Удаляются все существующие элементы из выпадающего меню borrower\_menu. Для каждого читателя в borrower\_options добавляется команда в выпадающее меню borrower\_menu, чтобы установить значение borrower\_var в выбранного читателя.

Обе функции выполняют обновление выпадающих меню выбора книги и читателя, чтобы отображать последние данные из базы данных.

На рисунке 14 изображена функция для обновления информации о читателях.

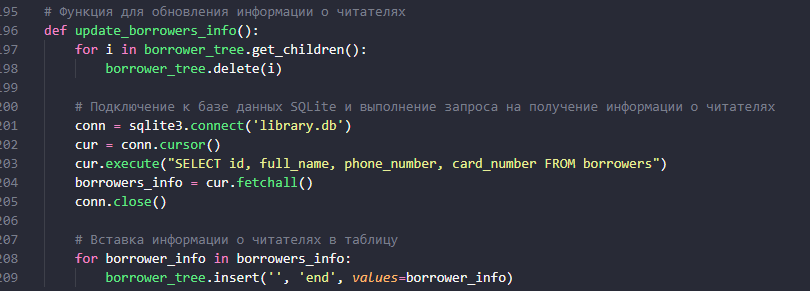


Рисунок 14. Создание функции для обновления информации о читателях

Эта функция обновляет информацию о читателях в таблице. Сначала функция удаляет все строки из таблицы читателей (происходит очистка таблицы). Это гарантирует, что перед загрузкой новой информации старая информация будет удалена, чтобы избежать дублирования. Затем функция устанавливает соединение с базой данных SQLite с именем "library.db". После этого выполняется SQL-запрос для извлечения информации о читателях из базы данных. После получения информации о читателях они вставляются в таблицу или интерфейс. Каждая запись о читателе (содержащая идентификатор, полное имя, номер телефона и номер билета) добавляется как новая строка в таблицу. Эта функция гарантирует, что информация о читателях в интерфейсе или таблице обновляется с учетом последних изменений в базе данных.

На рисунке 15 изображена функция для открытия окна добавления книги.

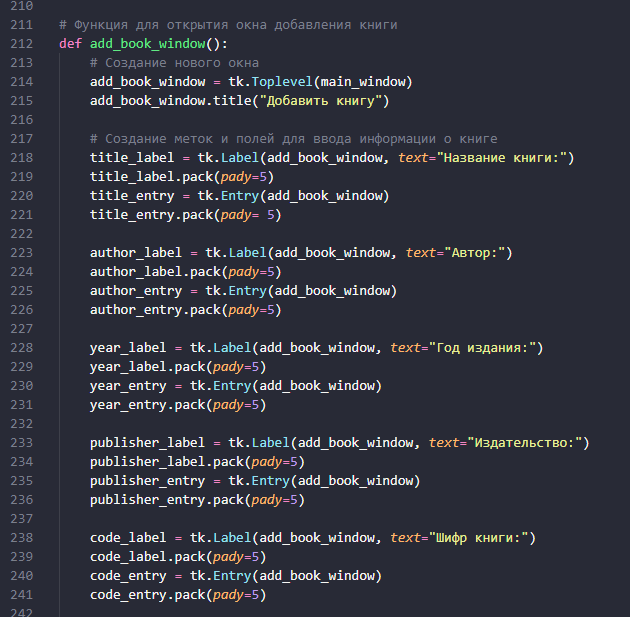


Рисунок 15. Создание функции для открытия окна добавления книги

Эта функция создает окно для добавления новой книги в библиотеку. Функция создает новое дочернее окно (Toplevel) от главного окна (main\_window) и устанавливает его заголовок.

Для каждого атрибута книги (название, автор, год издания, издательство, шифр) создается метка (Label) и поле для ввода (Entry). Метки и поля упаковываются методом pack() для отображения в созданном окне.

Эта функция готовит интерфейс для пользователя, чтобы тот мог ввести информацию о новой книге, которую он хочет добавить в библиотеку.

На рисунке 16 создание функции для добавления книги в БД.

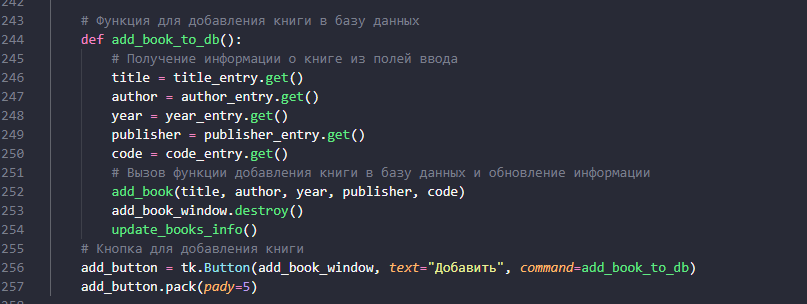


Рисунок 16. Создание функции для добавления книги в базу данных

Эта функция добавляет новую книгу в базу данных библиотеки, используя информацию, введенную пользователем в поля для ввода в окне добавления книги. Функция получает информацию о книге, введенную пользователем в поля для ввода, такие как название, автор, год издания, издательство и шифр. Затем функция вызывает функцию add\_book, передавая ей полученную информацию о книге. После добавления книги вызывается функция update\_books\_info для обновления информации о книгах в интерфейсе. После добавления книги и обновления информации в интерфейсе вызывается метод destroy() для закрытия окна добавления книги.

Создается кнопка "Добавить", которая при нажатии вызывает функцию add\_book\_to\_db.

Функция обеспечивает взаимодействие пользователя с интерфейсом для добавления новых книг в базу данных библиотеки.

На рисунке 17 изображена функция для открытия окна добавления читателя.

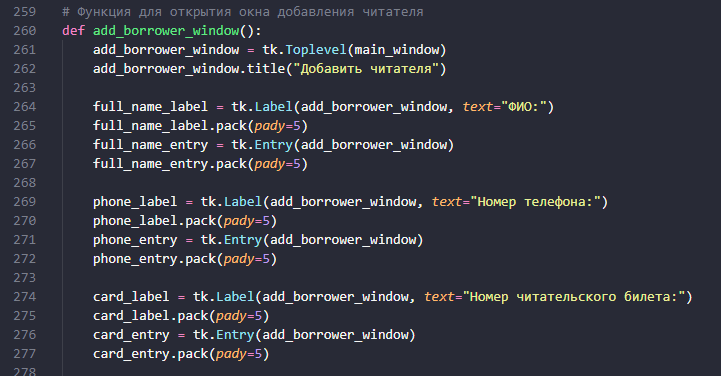


Рисунок 17. Создание функции для открытия окна добавления читателя

Эта функция создает окно для добавления нового читателя в библиотеку.

Функция создает новое дочернее окно (Toplevel) от главного окна (main\_window) и устанавливает его заголовок. Для каждого атрибута читателя (ФИО, номер телефона, номер читательского билета) создается метка (Label) и поле для ввода (Entry). Метки и поля упаковываются методом pack() для отображения в созданном окне. Функция готовит интерфейс для пользователя, чтобы тот мог ввести информацию о новом читателе, которого он хочет добавить в библиотеку.

На рисунке 18 изображена функция для добавления читателя в БД.

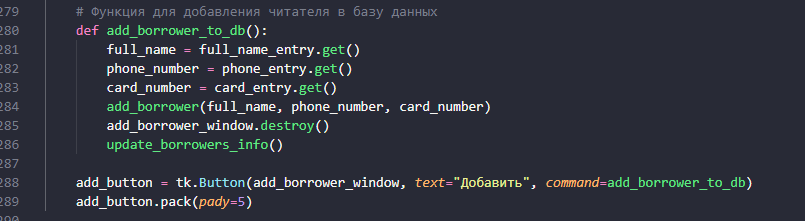


Рисунок 18. Создание функции для добавления читателя в базу данных

Эта функция добавляет нового читателя в базу данных библиотеки, используя информацию, введенную пользователем в поля для ввода в окне добавления читателя. Функция получает информацию о читателе, введенную пользователем в поля для ввода, такие как ФИО, номер телефона и номер читательского билета. Затем функция вызывает функцию add\_borrower, передавая ей полученную информацию о читателе. После добавления читателя вызывается функция update\_borrowers\_info для обновления информации о читателях в интерфейсе. После добавления читателя и обновления информации в интерфейсе вызывается метод destroy() для закрытия окна добавления читателя. Создается кнопка "Добавить", которая при нажатии вызывает функцию add\_borrower\_to\_db. Эта функция обеспечивает взаимодействие пользователя с интерфейсом для добавления новых читателей в базу данных библиотеки.

На рисунке 19 изображена функция для возможности открытия окна выдачи книги.

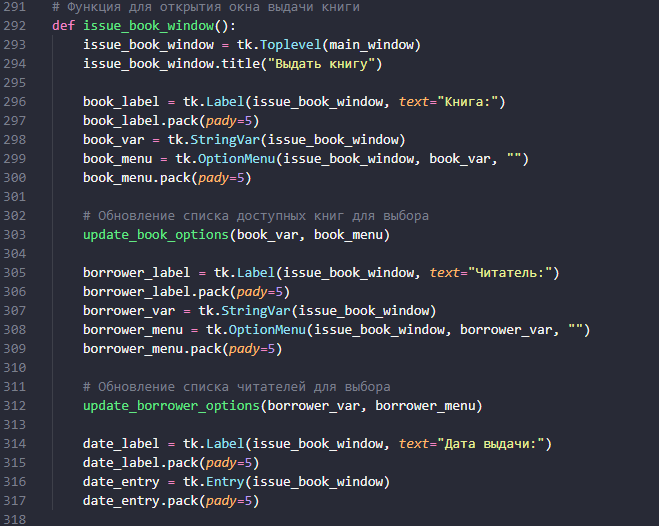


Рисунок 19. Создание функции для открытия окна выдачи книги

Эта функция создает окно для выдачи книги читателю. Функция создает новое дочернее окно (Toplevel) от главного окна (main\_window) и устанавливает его заголовок. Для каждого атрибута (книга, читатель, дата выдачи) создаются метки (Label) и соответствующие виджеты для выбора или ввода данных (OptionMenu для выбора книги и читателя, Entry для ввода даты). Для обновления списков доступных книг и читателей вызываются соответствующие функции update\_book\_options и update\_borrower\_options, передавая им переменные, связанные с выпадающими меню для выбора книги и читателя. Функция готовит интерфейс для пользователя, чтобы тот мог выбрать книгу, читателя и ввести дату выдачи для операции выдачи книги.

На рисунке 20 изображена функция для обработки выдачи выбранной книги читателю.

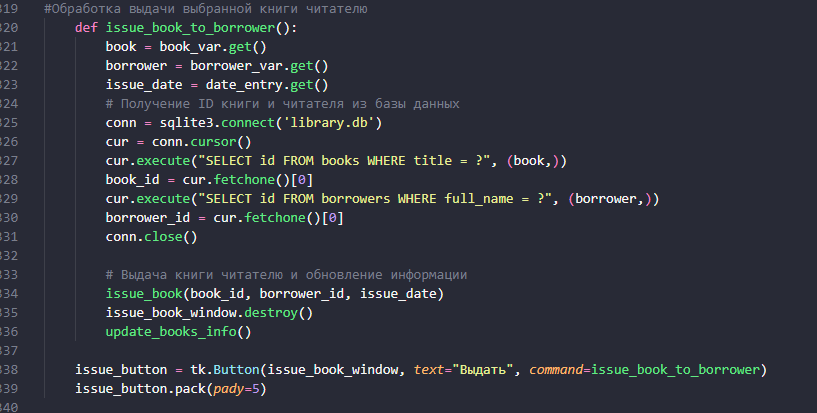


Рисунок 20. Создание функции для обработки выдачи выбранной книги читателю

Эта часть кода обрабатывает выдачу выбранной книги читателю после заполнения всех соответствующих полей в окне выдачи книги. Функция получает информацию о выбранной книге, выбранном читателе и введенной дате выдачи из соответствующих полей ввода. Для того чтобы выполнить выдачу, необходимо получить идентификаторы книги и читателя из базы данных. Для этого выполняются SQL-запросы к базе данных для получения идентификаторов книги и читателя на основе их названия (или имени читателя). После получения идентификаторов книги и читателя вызывается функция issue\_book, которая выполняет соответствующие SQL-запросы для выдачи книги читателю в базе данных. После этого окно выдачи книги закрывается методом destroy(), и обновляется информация о книгах в интерфейсе вызовом функции update\_books\_info(). Создается кнопка "Выдать", которая при нажатии вызывает функцию issue\_book\_to\_borrower.

Эта часть кода обеспечивает взаимодействие пользователя с интерфейсом для выполнения операции выдачи книги читателю.

На рисунке 21 изображена функция для возможности открытия овна возврата книги.

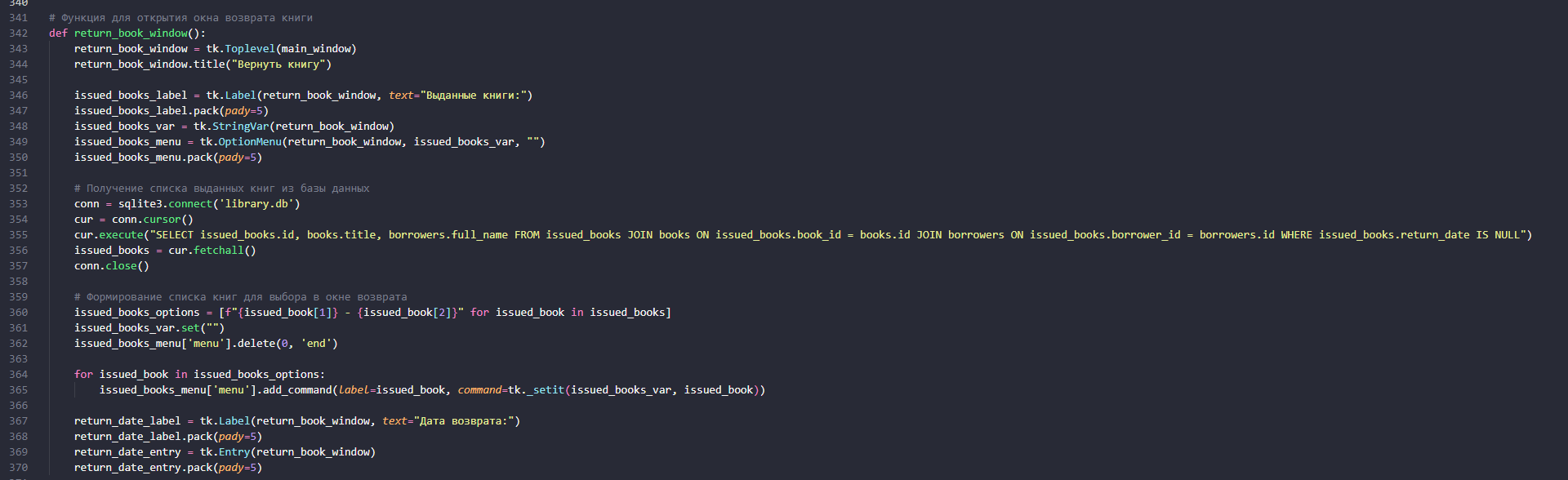


Рисунок 21. Создание функции для открытия окна возврата книги

Эта функция открывает окно для возврата книги в библиотеку. Функция создает новое дочернее окно (Toplevel) от главного окна (main\_window) и устанавливает его заголовок. Создается метка и выпадающее меню (OptionMenu) для выбора книги, которую читатель хочет вернуть. Метка указывает на список выданных, но еще не возвращенных книг. Выполняется SQL-запрос к базе данных для получения списка книг, которые были выданы, но еще не возвращены. Полученные книги сохраняются в переменной issued\_books. Для каждой выданной книги создается строка, содержащая название книги и имя читателя. Затем эти строки добавляются в выпадающее меню. Создается поле для ввода даты возврата. Это поле предоставляет пользователю возможность ввести дату, когда книга была возвращена в библиотеку.

Функция готовит интерфейс для пользователя, чтобы тот мог выбрать книгу, которую он хочет вернуть, и ввести дату возврата.

На рисунке 22 изображена функция для обработки возврата выбранной книги читателю.

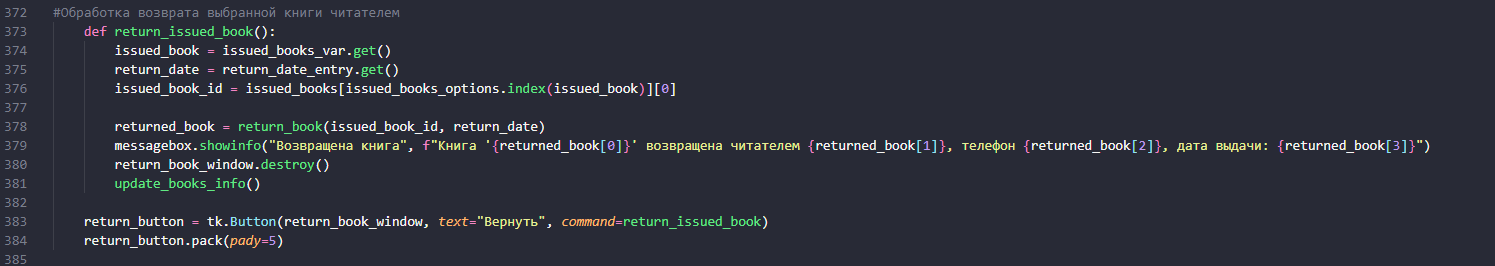


Рисунок 22. Создание функции для обработки возврата выбранной книги читателю

Этот код обрабатывает возврат выбранной книги читателем после заполнения всех соответствующих полей в окне возврата книги. Функция получает информацию о выбранной книге (которую читатель хочет вернуть) и введенной дате возврата из соответствующих полей ввода. Используя выбранную книгу, функция находит соответствующий ей идентификатор в списке выданных книг (issued\_books). Идентификатор выданной книги необходим для выполнения операции возврата. Функция вызывает функцию return\_book, передавая ей идентификатор выданной книги и введенную дату возврата. После успешного выполнения операции возврата, окно возврата книги закрывается методом destroy(), и обновляется информация о книгах в интерфейсе вызовом функции update\_books\_info(). После успешного возврата книги выводится информационное окно с деталями о книге, которая была возвращена, включая ее название, имя читателя, телефон и дату выдачи. Создается кнопка "Вернуть", которая при нажатии вызывает функцию return\_issued\_book.

Этот код обеспечивает взаимодействие пользователя с интерфейсом для выполнения операции возврата книги в библиотеку.

На рисунке 23 изображен код Python, который объявляет функцию создания таблицы для отображения книг.

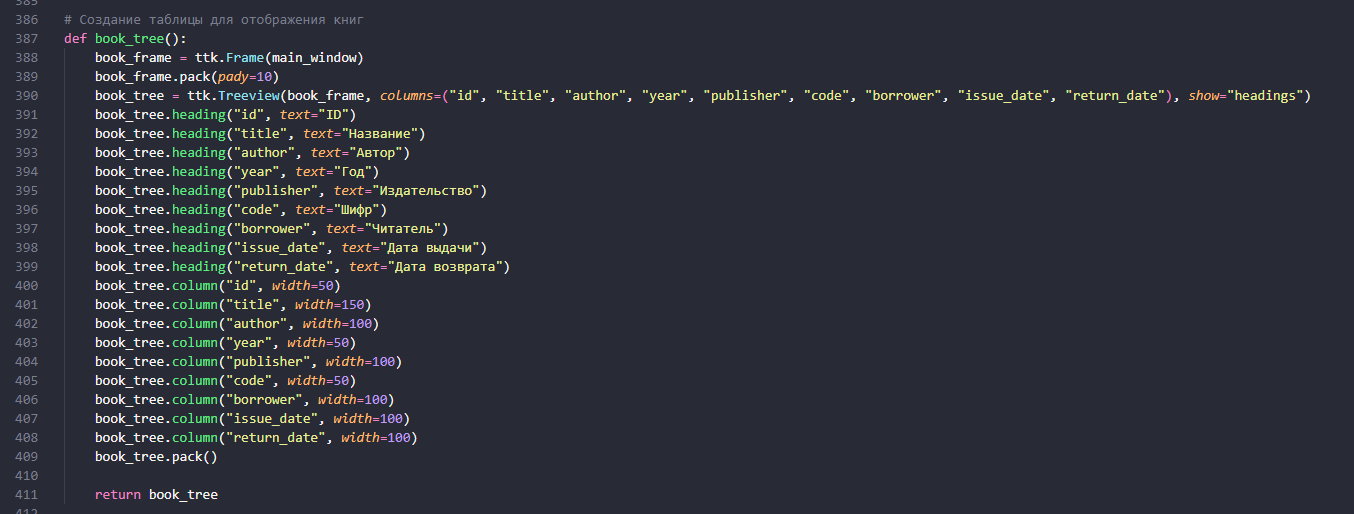


Рисунок 23. Создание таблицы для отображения книг

Этот код создает таблицу для отображения информации о книгах в графическом интерфейсе. Создается фрейм (book\_frame) для размещения таблицы, и этот фрейм упаковывается на главное окно. Затем создается объект Treeview (book\_tree), который представляет собой таблицу для отображения данных. Он имеет несколько колонок, соответствующих различным атрибутам книги. Для каждой колонки таблицы задаются заголовки с помощью метода heading. Также для каждой колонки устанавливается ширина с помощью метода column. Таблица упаковывается в фрейм с помощью метода pack(). В конце функция возвращает объект book\_tree, который представляет собой таблицу с настроенными заголовками и колонками.

Этот код создает инструмент для отображения информации о книгах в интерфейсе библиотеки.

На рисунке 24 изображена функция создания таблицы для отображения читателей.



Рисунок 24. Создание таблицы для отображения читателей

Этот код создает таблицу для отображения информации о читателях в графическом интерфейсе. Создается фрейм (borrower\_frame) для размещения таблицы, и этот фрейм упаковывается на главное окно. Затем создается объект Treeview (borrower\_tree), который представляет собой таблицу для отображения данных. Он имеет несколько колонок, соответствующих различным атрибутам читателя. Для каждой колонки таблицы задаются заголовки с помощью метода heading. Также для каждой колонки устанавливается ширина с помощью метода column. Таблица упаковывается в фрейм с помощью метода pack(). В конце функция возвращает объект borrower\_tree, который представляет собой таблицу с настроенными заголовками и колонками.

Этот код создает инструмент для отображения информации о читателях в интерфейсе библиотеки.

На рисунке 25 изображена основная часть программы.

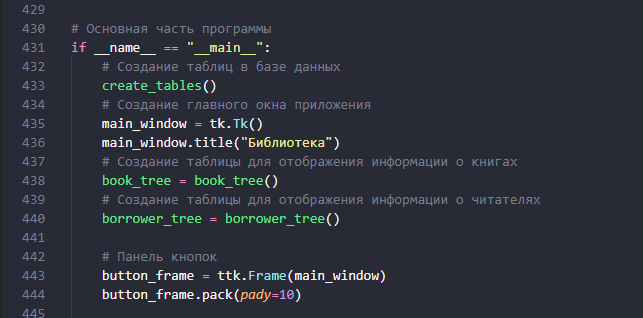


Рисунок 25. Основная часть программы

Эта часть кода является основной частью программы. Условие if \_\_name\_\_ == "\_\_main\_\_" гарантирует, что этот блок кода выполнится только если файл был запущен как скрипт, а не импортирован как модуль. Вызывается функция create\_tables(), которая создает необходимые таблицы в базе данных, если они еще не существуют. Создается основное окно приложения с помощью tk.Tk(), устанавливается его заголовок. Вызываются функции book\_tree() и borrower\_tree(), которые создают и настраивают таблицы для отображения информации о книгах и читателях.

Этот блок кода представляет основную структуру приложения, включая создание графического интерфейса и подготовку базы данных для работы программы.

На рисунке 26 представлен код Pyhton, который создает кнопки для различных действий.

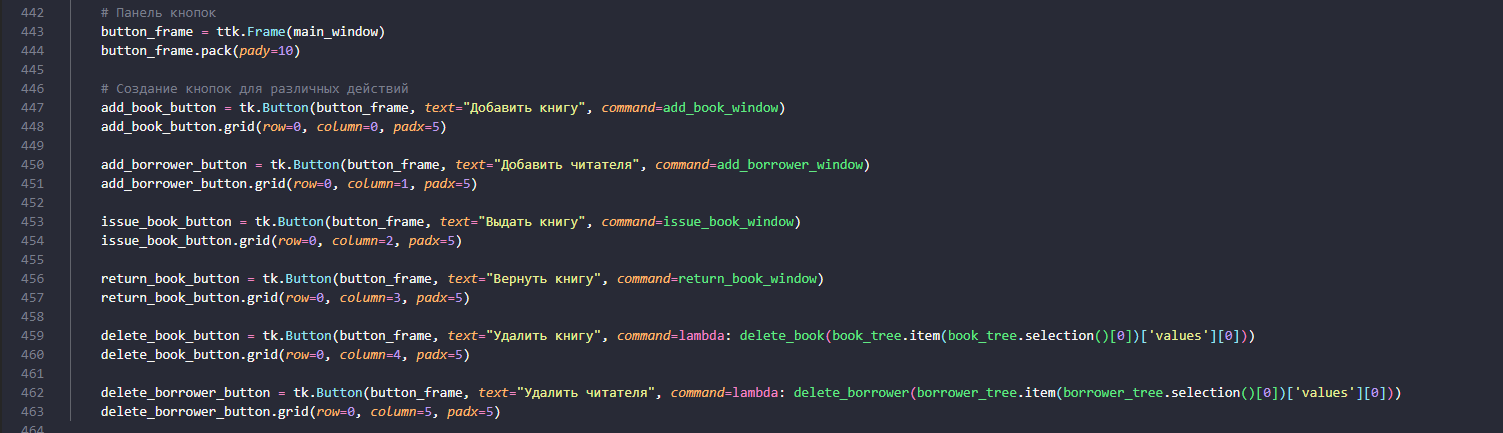


Рисунок 26. Создание кнопок

Этот код создает панель кнопок для выполнения различных действий в программе. Создается фрейм (button\_frame), который будет содержать кнопки, и этот фрейм упаковывается на главное окно.

Создается кнопка "Добавить книгу", при нажатии на которую открывается окно для добавления новой книги.

Создается кнопка "Добавить читателя", при нажатии на которую открывается окно для добавления нового читателя.

Создается кнопка "Выдать книгу", при нажатии на которую открывается окно для выдачи книги читателю.

Создается кнопка "Вернуть книгу", при нажатии на которую открывается окно для возврата книги в библиотеку.

Создается кнопка "Удалить книгу", при нажатии на которую удаляется выбранная книга из базы данных и обновляется информация.

Создается кнопка "Удалить читателя", при нажатии на которую удаляется выбранный читатель из базы данных и обновляется информация.

Каждая кнопка связана с соответствующей функцией или окном, которые выполняют определенные действия в программе.

На рисунке 27 изображена функция для отображения данных.

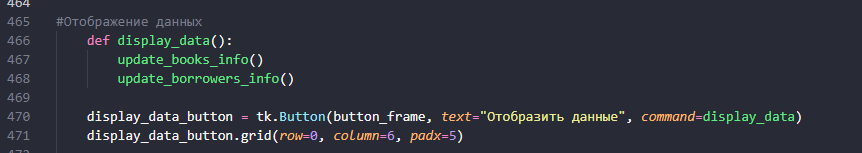


Рисунок 27. Создание функции для отображения данных

Функция display\_data() предназначена для обновления и отображения данных о книгах и читателях в таблицах. Кнопка "Отобразить данные" связана с этой функцией и позволяет пользователю обновить информацию на экране в любое время.

На рисунке 28 изображен код Python, который запускает основной цикл программы.

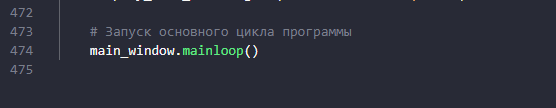


Рисунок 28. Запуск основного цикла программы

Этот вызов main\_window.mainloop() запускает основной цикл программы Tkinter, который ожидает событий от пользователя (например, нажатие кнопок, перемещение мыши) и обрабатывает их соответственно. Он продолжает выполнение программы до тех пор, пока пользователь не закроет главное окно, после чего программа завершается.

## **Итоговая версия десктоп-приложения**

На рисунке 29 изображено десктоп-приложение для учета выданных книг в библиотеке.

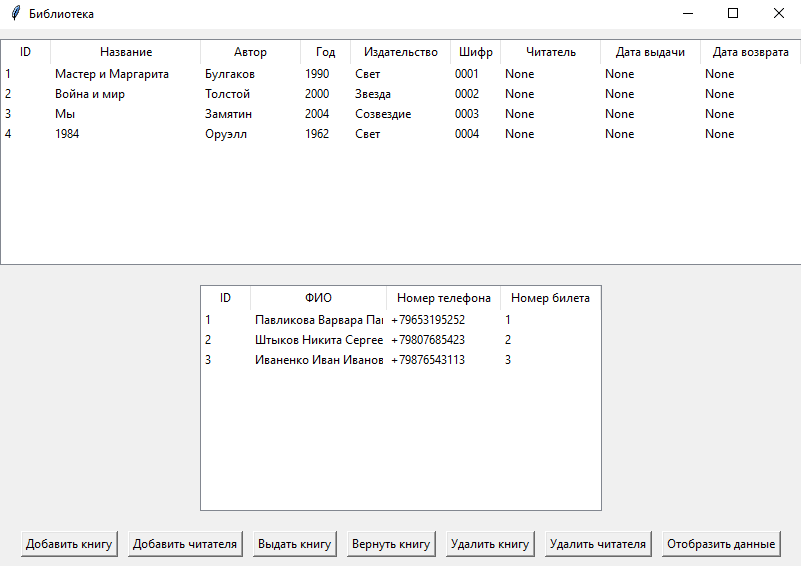


Рисунок 29. Информационная система для учета выданных книг в библиотеке

### **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы были осуществлены исследование и разработка информационной системы для учета выданных книг в библиотеке с целью оптимизации и упрощения процесса управления библиотечным фондом. Результаты работы представляют собой значимый вклад в область библиотечного дела и информационных технологий.

Первоначально была проведена детальная аналитика предметной области библиотечного дела и процесса учета книг, что позволило определить основные требования пользователей и функциональные возможности информационной системы. Затем были выбраны соответствующие технологии и программные решения для реализации системы, включая язык программирования Python, среду разработки Visual Studio Code и базу данных.

На основе этих данных была разработана и реализована программа для создания информационной системы, которая эффективно учитывает выданные книги в библиотеке. Эта система представляет собой инструмент, способствующий упрощению и оптимизации работы библиотечного учреждения, увеличивая эффективность обслуживания пользователей.

Важным этапом работы является проведенное тестирование разработанной системы, которое позволило убедиться в ее надежности и функциональности.

Практическая значимость данной работы состоит в том, что разработанная информационная система представляет собой важный инструмент для библиотек, способствующий повышению уровня сервиса и эффективности работы учреждения. Кроме того, результаты данной работы могут быть использованы в качестве дополнительного элемента в портфолио для демонстрации профессиональных навыков в программировании.

Таким образом, выполненная курсовая работа успешно решает поставленные задачи, достигает цели и вносит свой вклад в развитие сферы информационных технологий и библиотечного дела.

### **ИСПОЛЬЗУЕМАЯ** **ЛИТЕРАТУРА**

**Нормативные документы:**

Конституция Российской Федерации (с учетом поправок, внесен­ных за­конами Российской Федерации «О поправках к Конституции Россий­ской Федерации» от 30 декабря 2008 г. № 6-ФКЗ и от 30 декабря 2008г. № 7-ФКЗ) // Российская газета № 4831 от 21 января 2009 г.

Федеральный закон Российской Федерации от 7 февраля 2011 № 3-ФЗ «О полиции»// Россий­ская газета от 8 февраля 2011 № 5401.

**Основные источники:**

Бен, Форта SQL за 10 минут / Форта Бен. - М.: Диалектика / Вильямс, 2015. - 673 c.

Дунаев, В. В. Базы данных. Язык SQL для студента / В.В. Дунаев. - М.: БХВ-Петербург, 2016. - 288 c.

Эйри, Джоунс Функции SQL. Справочник программиста / Джоунс Эйри. - М.: Диалектика / Вильямс, 2014. - 556 c

Билл Любанович. Простой Python. Современный стиль программирования. / Билл Любанович. – 2020г. – 592с.

Кольцов Д.М., Дубовик О.В. Справочник Python. Кратко, быстро, под рукой. / Кольцов Д.М., Дубовик О.В. – Наука и техника. – 2021г. – 290с.

Эл Свейгарт. Большая книга проектов Python. / Эл Свейгарт. – Питер. – 2022г. – 432с.

Поль Дюбуа. MySQL. Сборник рецептов. / Поль Дюбуа. – Символ-Плюс. – 2005г. – 1056с.

Шаббир Чаллавала, Джадип Лакхатария, Чинтан Мехта, Кандарп Патель. MySQL 8 для больших данных. / Шаббир Чаллавала, Джадип Лакхатария, Чинтан Мехта, Кандарп Патель. – ДМК Пресс. – 2018г. – 228с.

Мигель Гринберг. Разработка веб-приложений с использованием Flask на языке Python. / Мигель Гринберг. – ДМК Пресс. – 2016г. – 274с.

**Дополнительные источники**:

Маркин, А. В. Построение запросов и программирование на SQL. Учебное пособие / А.В. Маркин. - М.: Диалог-Мифи, 2014. - 384 c.

**Словари и энциклопедии**

Социальная философия: словарь / под общ. ред. В. Е. Кемерова, Т. Х. Керимова. – М. : Академический Проект, 2011. – 588 с.

**Электронные ресурсы:**

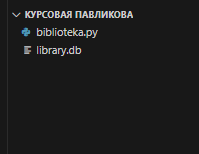
<https://flask.palletsprojects.com/en/3.0.x/>

<https://timeweb.cloud/tutorials/python/rabota-s-bazami-dannyh-sql-v-python>

<https://www.ihc.ru/articles/instrukciya-po-nastrojke-vpsvds-servera-s-nulya.html>

### **ПРИЛОЖЕНИЕ**

Приложение 1. Репозиторий проекта



Приложение 2. Исходный код

# Импорт необходимых библиотек

import sqlite3

import tkinter as tk

from tkinter import ttk, messagebox

# Функция для создания таблиц в базе данных

def create\_tables():

    # Подключение к базе данных

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

    cur = conn.cursor()

    # Создание таблицы книг

    cur.execute('''CREATE TABLE IF NOT EXISTS books (

                    id INTEGER PRIMARY KEY,

                    title TEXT,

                    author TEXT,

                    year INTEGER,

                    publisher TEXT,

                    code TEXT UNIQUE,

                    issued INTEGER DEFAULT 0

                )''')

    # Создание таблицы читателей

    cur.execute('''CREATE TABLE IF NOT EXISTS borrowers (

                    id INTEGER PRIMARY KEY,

                    full\_name TEXT,

                    phone\_number TEXT,

                    card\_number TEXT UNIQUE

                )''')

    # Создание таблицы выданных книг

    cur.execute('''CREATE TABLE IF NOT EXISTS issued\_books (

                    id INTEGER PRIMARY KEY,

                    book\_id INTEGER,

                    borrower\_id INTEGER,

                    issue\_date DATE,

                    return\_date DATE,

                    FOREIGN KEY (book\_id) REFERENCES books(id),

                    FOREIGN KEY (borrower\_id) REFERENCES borrowers(id)

                )''')

    # Сохранение изменений и закрытие соединения с базой данных

    conn.commit()

    conn.close()

# Функция для обновления информации о книгах

def update\_books\_info():

    # Очистка таблицы книг перед обновлением

    for i in book\_tree.get\_children():

        book\_tree.delete(i)

    # Подключение к базе данных

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

    cur = conn.cursor()

    # Получение информации о книгах из базы данных

    cur.execute('''SELECT books.id, books.title, books.author, books.year, books.publisher, books.code,

                          borrowers.full\_name, issued\_books.issue\_date, issued\_books.return\_date

                   FROM books

                   LEFT JOIN issued\_books ON books.id = issued\_books.book\_id

                   LEFT JOIN borrowers ON issued\_books.borrower\_id = borrowers.id''')

    books\_info = cur.fetchall()

    # Закрытие соединения с базой данных

    conn.close()

    # Вставка информации о книгах в таблицу

    for book\_info in books\_info:

        book\_tree.insert('', 'end', values=book\_info)

# Функция для обновления информации о читателях

def update\_borrowers\_info():

    # Очистка таблицы читателей перед обновлением

    for i in borrower\_tree.get\_children():

        borrower\_tree.delete(i)

    # Подключение к базе данных

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

    cur = conn.cursor()

    # Получение информации о читателях из базы данных

    cur.execute("SELECT full\_name, phone\_number, card\_number FROM borrowers")

    borrowers\_info = cur.fetchall()

    conn.close()

    # Вставка информации о читателях в таблицу

    for borrower\_info in borrowers\_info:

        borrower\_tree.insert('', 'end', values=(borrower\_info[0], borrower\_info[1], borrower\_info[2]))

# Функция для добавления книги в базу данных

def add\_book(title, author, year, publisher, code):

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

    cur = conn.cursor()

    try:

        # Вставка данных о книге в таблицу

        cur.execute("INSERT INTO books (title, author, year, publisher, code) VALUES (?, ?, ?, ?, ?)", (title, author, year, publisher, code))

        conn.commit()

    except sqlite3.IntegrityError:

        # Вывод сообщения об ошибке, если книга с таким шифром уже существует

        messagebox.showerror("Ошибка", "Книга с таким шифром уже существует!")

    finally:

        conn.close()

        update\_books\_info()

# Добавление читателя в базу данных

def add\_borrower(full\_name, phone\_number, card\_number):

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

    cur = conn.cursor()

    try:

        cur.execute("INSERT INTO borrowers (full\_name, phone\_number, card\_number) VALUES (?, ?, ?)", (full\_name, phone\_number, card\_number))

        conn.commit()

    except sqlite3.IntegrityError:

        messagebox.showerror("Ошибка", "Читатель с таким номером читательского билета уже существует!")

    finally:

        conn.close()

        update\_borrowers\_info()

# Выдача книги читателю

def issue\_book(book\_id, borrower\_id, issue\_date):

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

    cur = conn.cursor()

    cur.execute("INSERT INTO issued\_books (book\_id, borrower\_id, issue\_date) VALUES (?, ?, ?)", (book\_id, borrower\_id, issue\_date))

    cur.execute("UPDATE books SET issued = 1 WHERE id = ?", (book\_id,))

    conn.commit()

    conn.close()

    update\_books\_info()

# Возврат книги читателю

def return\_book(issued\_book\_id, return\_date):

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

    cur = conn.cursor()

    cur.execute("UPDATE issued\_books SET return\_date = ? WHERE id = ?", (return\_date, issued\_book\_id))

    issued\_book = cur.execute("SELECT books.title, borrowers.full\_name, borrowers.phone\_number, issued\_books.issue\_date FROM issued\_books JOIN books ON issued\_books.book\_id = books.id JOIN borrowers ON issued\_books.borrower\_id = borrowers.id WHERE issued\_books.id = ?", (issued\_book\_id,)).fetchone()

    conn.commit()

    conn.close()

    update\_books\_info()

    return issued\_book

# Удаление книги из базы данных

def delete\_book(book\_id):

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

    cur = conn.cursor()

    try:

        cur.execute('DELETE FROM books WHERE id = ?', (book\_id,))

        conn.commit()

    except Exception as e:

        messagebox.showerror("Ошибка", f"Ошибка при удалении книги: {e}")

    finally:

        conn.close()

        update\_books\_info()

# Удаление читателя из базы данных

def delete\_borrower(borrower\_id):

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

    cur = conn.cursor()

    try:

        cur.execute('DELETE FROM borrowers WHERE id = ?', (borrower\_id,))

        conn.commit()

        update\_borrowers\_info()  # Обновляем таблицу после удаления читателя

    except Exception as e:

        messagebox.showerror("Ошибка", f"Ошибка при удалении читателя: {e}")

    finally:

        conn.close()

# Функция для обновления списка доступных книг при выборе книги для выдачи

def update\_book\_options(book\_var, book\_menu):

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

    cur = conn.cursor()

    # Получение списка доступных книг (не выданных)

    cur.execute("SELECT id, title FROM books WHERE issued = 0")

    books = cur.fetchall()

    conn.close()

    book\_options = [book[1] for book in books]

    book\_var.set("")

    book\_menu['menu'].delete(0, 'end')

    # Добавление опций выбора книги в выпадающее меню

    for book in book\_options:

        book\_menu['menu'].add\_command(label=book, command=tk.\_setit(book\_var, book))

# Функция для обновления списка доступных читателей при выборе читателя для выдачи книги

def update\_borrower\_options(borrower\_var, borrower\_menu):

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

    cur = conn.cursor()

    # Получение списка читателей

    cur.execute("SELECT id, full\_name FROM borrowers")

    borrowers = cur.fetchall()

    conn.close()

    borrower\_options = [borrower[1] for borrower in borrowers]

    borrower\_var.set("")

    borrower\_menu['menu'].delete(0, 'end')

    # Добавление опций выбора читателя в выпадающее меню

    for borrower in borrower\_options:

        borrower\_menu['menu'].add\_command(label=borrower, command=tk.\_setit(borrower\_var, borrower))

# Функция для обновления информации о читателях

def update\_borrowers\_info():

    for i in borrower\_tree.get\_children():

        borrower\_tree.delete(i)

    # Подключение к базе данных SQLite и выполнение запроса на получение информации о читателях

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

    cur = conn.cursor()

    cur.execute("SELECT id, full\_name, phone\_number, card\_number FROM borrowers")

    borrowers\_info = cur.fetchall()

    conn.close()

    # Вставка информации о читателях в таблицу

    for borrower\_info in borrowers\_info:

        borrower\_tree.insert('', 'end', values=borrower\_info)

# Функция для открытия окна добавления книги

def add\_book\_window():

    # Создание нового окна

    add\_book\_window = tk.Toplevel(main\_window)

    add\_book\_window.title("Добавить книгу")

    # Создание меток и полей для ввода информации о книге

    title\_label = tk.Label(add\_book\_window, text="Название книги:")

    title\_label.pack(pady=5)

    title\_entry = tk.Entry(add\_book\_window)

    title\_entry.pack(pady= 5)

    author\_label = tk.Label(add\_book\_window, text="Автор:")

    author\_label.pack(pady=5)

    author\_entry = tk.Entry(add\_book\_window)

    author\_entry.pack(pady=5)

    year\_label = tk.Label(add\_book\_window, text="Год издания:")

    year\_label.pack(pady=5)

    year\_entry = tk.Entry(add\_book\_window)

    year\_entry.pack(pady=5)

    publisher\_label = tk.Label(add\_book\_window, text="Издательство:")

    publisher\_label.pack(pady=5)

    publisher\_entry = tk.Entry(add\_book\_window)

    publisher\_entry.pack(pady=5)

    code\_label = tk.Label(add\_book\_window, text="Шифр книги:")

    code\_label.pack(pady=5)

    code\_entry = tk.Entry(add\_book\_window)

    code\_entry.pack(pady=5)

    # Функция для добавления книги в базу данных

    def add\_book\_to\_db():

        # Получение информации о книге из полей ввода

        title = title\_entry.get()

        author = author\_entry.get()

        year = year\_entry.get()

        publisher = publisher\_entry.get()

        code = code\_entry.get()

        # Вызов функции добавления книги в базу данных и обновление информации

        add\_book(title, author, year, publisher, code)

        add\_book\_window.destroy()

        update\_books\_info()

    # Кнопка для добавления книги

    add\_button = tk.Button(add\_book\_window, text="Добавить", command=add\_book\_to\_db)

    add\_button.pack(pady=5)

# Функция для открытия окна добавления читателя

def add\_borrower\_window():

    add\_borrower\_window = tk.Toplevel(main\_window)

    add\_borrower\_window.title("Добавить читателя")

    full\_name\_label = tk.Label(add\_borrower\_window, text="ФИО:")

    full\_name\_label.pack(pady=5)

    full\_name\_entry = tk.Entry(add\_borrower\_window)

    full\_name\_entry.pack(pady=5)

    phone\_label = tk.Label(add\_borrower\_window, text="Номер телефона:")

    phone\_label.pack(pady=5)

    phone\_entry = tk.Entry(add\_borrower\_window)

    phone\_entry.pack(pady=5)

    card\_label = tk.Label(add\_borrower\_window, text="Номер читательского билета:")

    card\_label.pack(pady=5)

    card\_entry = tk.Entry(add\_borrower\_window)

    card\_entry.pack(pady=5)

    # Функция для добавления читателя в базу данных

    def add\_borrower\_to\_db():

        full\_name = full\_name\_entry.get()

        phone\_number = phone\_entry.get()

        card\_number = card\_entry.get()

        add\_borrower(full\_name, phone\_number, card\_number)

        add\_borrower\_window.destroy()

        update\_borrowers\_info()

    add\_button = tk.Button(add\_borrower\_window, text="Добавить", command=add\_borrower\_to\_db)

    add\_button.pack(pady=5)

# Функция для открытия окна выдачи книги

def issue\_book\_window():

    issue\_book\_window = tk.Toplevel(main\_window)

    issue\_book\_window.title("Выдать книгу")

    book\_label = tk.Label(issue\_book\_window, text="Книга:")

    book\_label.pack(pady=5)

    book\_var = tk.StringVar(issue\_book\_window)

    book\_menu = tk.OptionMenu(issue\_book\_window, book\_var, "")

    book\_menu.pack(pady=5)

    # Обновление списка доступных книг для выбора

    update\_book\_options(book\_var, book\_menu)

    borrower\_label = tk.Label(issue\_book\_window, text="Читатель:")

    borrower\_label.pack(pady=5)

    borrower\_var = tk.StringVar(issue\_book\_window)

    borrower\_menu = tk.OptionMenu(issue\_book\_window, borrower\_var, "")

    borrower\_menu.pack(pady=5)

    # Обновление списка читателей для выбора

    update\_borrower\_options(borrower\_var, borrower\_menu)

    date\_label = tk.Label(issue\_book\_window, text="Дата выдачи:")

    date\_label.pack(pady=5)

    date\_entry = tk.Entry(issue\_book\_window)

    date\_entry.pack(pady=5)

#Обработка выдачи выбранной книги читателю

    def issue\_book\_to\_borrower():

        book = book\_var.get()

        borrower = borrower\_var.get()

        issue\_date = date\_entry.get()

        # Получение ID книги и читателя из базы данных

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

        cur = conn.cursor()

        cur.execute("SELECT id FROM books WHERE title = ?", (book,))

        book\_id = cur.fetchone()[0]

        cur.execute("SELECT id FROM borrowers WHERE full\_name = ?", (borrower,))

        borrower\_id = cur.fetchone()[0]

        conn.close()

        # Выдача книги читателю и обновление информации

        issue\_book(book\_id, borrower\_id, issue\_date)

        issue\_book\_window.destroy()

        update\_books\_info()

    issue\_button = tk.Button(issue\_book\_window, text="Выдать", command=issue\_book\_to\_borrower)

    issue\_button.pack(pady=5)

# Функция для открытия окна возврата книги

def return\_book\_window():

    return\_book\_window = tk.Toplevel(main\_window)

    return\_book\_window.title("Вернуть книгу")

    issued\_books\_label = tk.Label(return\_book\_window, text="Выданные книги:")

    issued\_books\_label.pack(pady=5)

    issued\_books\_var = tk.StringVar(return\_book\_window)

    issued\_books\_menu = tk.OptionMenu(return\_book\_window, issued\_books\_var, "")

    issued\_books\_menu.pack(pady=5)

    # Получение списка выданных книг из базы данных

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

    cur = conn.cursor()

    cur.execute("SELECT issued\_books.id, books.title, borrowers.full\_name FROM issued\_books JOIN books ON issued\_books.book\_id = books.id JOIN borrowers ON issued\_books.borrower\_id = borrowers.id WHERE issued\_books.return\_date IS NULL")

    issued\_books = cur.fetchall()

    conn.close()

    # Формирование списка книг для выбора в окне возврата

    issued\_books\_options = [f"{issued\_book[1]} - {issued\_book[2]}" for issued\_book in issued\_books]

    issued\_books\_var.set("")

    issued\_books\_menu['menu'].delete(0, 'end')

    for issued\_book in issued\_books\_options:

        issued\_books\_menu['menu'].add\_command(label=issued\_book, command=tk.\_setit(issued\_books\_var, issued\_book))

    return\_date\_label = tk.Label(return\_book\_window, text="Дата возврата:")

    return\_date\_label.pack(pady=5)

    return\_date\_entry = tk.Entry(return\_book\_window)

    return\_date\_entry.pack(pady=5)

#Обработка возврата выбранной книги читателем

    def return\_issued\_book():

        issued\_book = issued\_books\_var.get()

        return\_date = return\_date\_entry.get()

        issued\_book\_id = issued\_books[issued\_books\_options.index(issued\_book)][0]

        returned\_book = return\_book(issued\_book\_id, return\_date)

        messagebox.showinfo("Возвращена книга", f"Книга '{returned\_book[0]}' возвращена читателем {returned\_book[1]}, телефон {returned\_book[2]}, дата выдачи: {returned\_book[3]}")

        return\_book\_window.destroy()

        update\_books\_info()

    return\_button = tk.Button(return\_book\_window, text="Вернуть", command=return\_issued\_book)

    return\_button.pack(pady=5)

# Создание таблицы для отображения книг

def book\_tree():

    book\_frame = ttk.Frame(main\_window)

    book\_frame.pack(pady=10)

    book\_tree = ttk.Treeview(book\_frame, columns=("id", "title", "author", "year", "publisher", "code", "borrower", "issue\_date", "return\_date"), show="headings")

    book\_tree.heading("id", text="ID")

    book\_tree.heading("title", text="Название")

    book\_tree.heading("author", text="Автор")

    book\_tree.heading("year", text="Год")

    book\_tree.heading("publisher", text="Издательство")

    book\_tree.heading("code", text="Шифр")

    book\_tree.heading("borrower", text="Читатель")

    book\_tree.heading("issue\_date", text="Дата выдачи")

    book\_tree.heading("return\_date", text="Дата возврата")

    book\_tree.column("id", width=50)

    book\_tree.column("title", width=150)

    book\_tree.column("author", width=100)

    book\_tree.column("year", width=50)

    book\_tree.column("publisher", width=100)

    book\_tree.column("code", width=50)

    book\_tree.column("borrower", width=100)

    book\_tree.column("issue\_date", width=100)

    book\_tree.column("return\_date", width=100)

    book\_tree.pack()

    return book\_tree

# Создание таблицы для отображения читателей

def borrower\_tree():

    borrower\_frame = ttk.Frame(main\_window)

    borrower\_frame.pack(pady=10)

    borrower\_tree = ttk.Treeview(borrower\_frame, columns=("id", "full\_name", "phone\_number", "card\_number"), show="headings")

    borrower\_tree.heading("id", text="ID")

    borrower\_tree.heading("full\_name", text="ФИО")

    borrower\_tree.heading("phone\_number", text="Номер телефона")

    borrower\_tree.heading("card\_number", text="Номер билета")

    borrower\_tree.column("id", width=50)

    borrower\_tree.column("full\_name", width=150)

    borrower\_tree.column("phone\_number", width=100)

    borrower\_tree.column("card\_number", width=100)

    borrower\_tree.pack()

    return borrower\_tree

# Основная часть программы

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

    # Создание таблиц в базе данных

    create\_tables()

    # Создание главного окна приложения

    main\_window = tk.Tk()

    main\_window.title("Библиотека")

    # Создание таблицы для отображения информации о книгах

    book\_tree = book\_tree()

    # Создание таблицы для отображения информации о читателях

    borrower\_tree = borrower\_tree()

    # Панель кнопок

    button\_frame = ttk.Frame(main\_window)

    button\_frame.pack(pady=10)

    # Создание кнопок для различных действий

    add\_book\_button = tk.Button(button\_frame, text="Добавить книгу", command=add\_book\_window)

    add\_book\_button.grid(row=0, column=0, padx=5)

    add\_borrower\_button = tk.Button(button\_frame, text="Добавить читателя", command=add\_borrower\_window)

    add\_borrower\_button.grid(row=0, column=1, padx=5)

    issue\_book\_button = tk.Button(button\_frame, text="Выдать книгу", command=issue\_book\_window)

    issue\_book\_button.grid(row=0, column=2, padx=5)

    return\_book\_button = tk.Button(button\_frame, text="Вернуть книгу", command=return\_book\_window)

    return\_book\_button.grid(row=0, column=3, padx=5)

    delete\_book\_button = tk.Button(button\_frame, text="Удалить книгу", command=lambda: delete\_book(book\_tree.item(book\_tree.selection()[0])['values'][0]))

    delete\_book\_button.grid(row=0, column=4, padx=5)

    delete\_borrower\_button = tk.Button(button\_frame, text="Удалить читателя", command=lambda: delete\_borrower(borrower\_tree.item(borrower\_tree.selection()[0])['values'][0]))

    delete\_borrower\_button.grid(row=0, column=5, padx=5)

#Отображение данных

    def display\_data():

        update\_books\_info()

        update\_borrowers\_info()

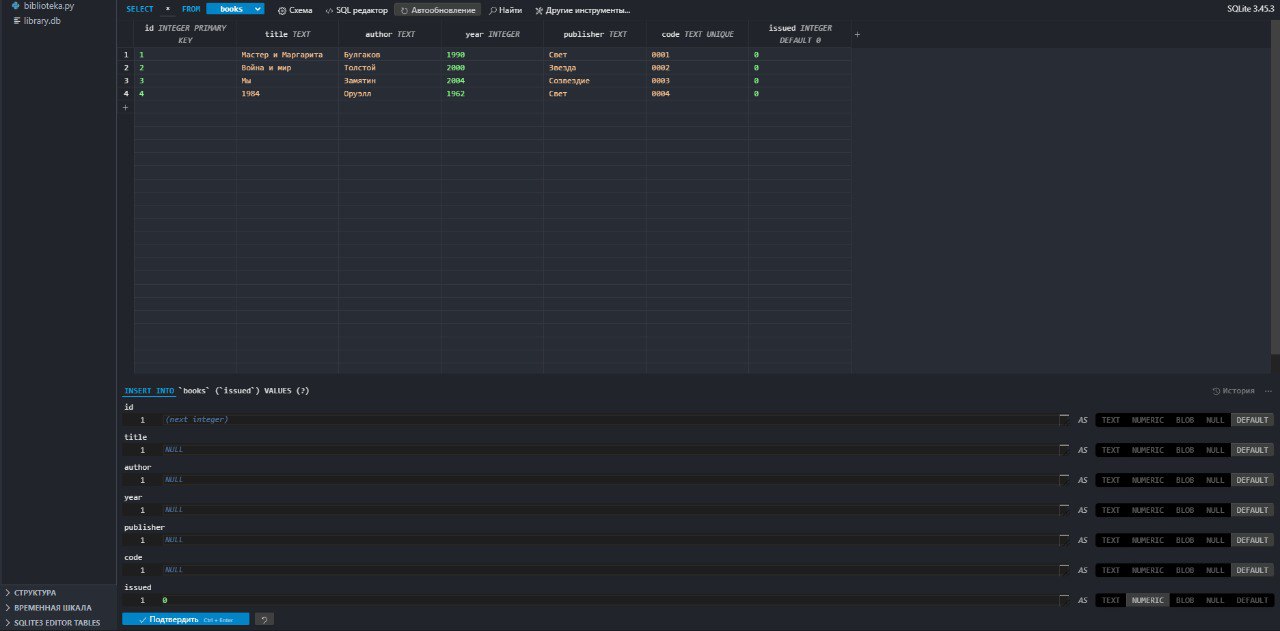
    display\_data\_button = tk.Button(button\_frame, text="Отобразить данные", command=display\_data)

    display\_data\_button.grid(row=0, column=6, padx=5)

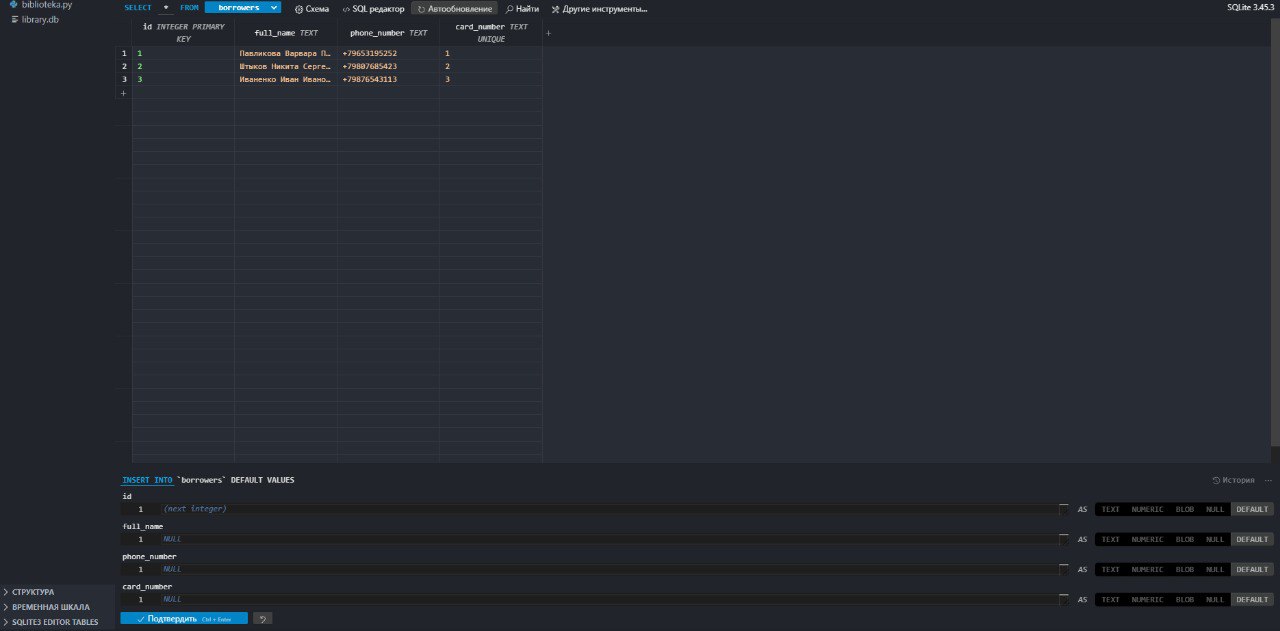
    # Запуск основного цикла программы

    main\_window.mainloop()

Приложение 3. База данных (книги)



Приложение 4. База данных (читатели)



Приложение 5. База данных (выданные книги)

