

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

филиал «РКТ» МАИ в г. Химки Московской области

Специальность 09.02.07 Информационные системы и программирование

Группа ИСП-41-19

Квалификация Программист

КУРСОВОЙ ПРОЕКТ (РАБОТА)

По МДК 02.01 Технология разработки программного обеспечения

(указывается код и название дисциплины, МДК, ПМ)

На тему: Разработка информационной подсистемы управления заказами клиентов книжного

магазина

Автор курсового проекта (работы) Макарова Дарья Андреевна /

(Фамилия Имя Отчество) (подпись)

Руководитель Жилина Татьяна Анатольевна /

(Фамилия Имя Отчество) (подпись)



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (МАИ)

филиал «РКТ» МАИ в г. Химки Московской области

Специальность 09.02.07 Информационные системы и программирование

Группа ИСП-41-19

Квалификация Программист

З А Д А Н И Е

на курсовой проект (работу)

По МДК 02.01 Технология разработки программного обеспечения

(указывается код и название дисциплины, МДК, ПМ)

Студенту Макаровой Дарье Андреевне /

(Фамилия Имя Отчество) (подпись)

Руководитель Жилина Татьяна Анатольевна /

(Фамилия Имя Отчество) (подпись)

Преподаватель, филиал «РКТ» МАИ в г. Химки Московской области

(Ученая степень, ученое звание, должность и место работы)

1. Наименование темы: Разработка информационной подсистемы управления заказами

клиентов книжного магазина

2. Срок сдачи студентом законченной работы 22.02.2023

3. Техническое задание и исходные данные к работе

Разработать информационную подсистему управления заказами клиентов книжного магазина, и реализовать её функционал: создание заказа, редактирование данных таблиц в базе данных, создание отчётов, а также реализовать разграничение доступа. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Перечень подлежащих разработке разделов и этапы выполнения работы

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование раздела или этапа | Трудоёмкость в % от  полной трудоёмкости  курсовой проекта(работы) | Срок выполнения |
| 1 | Анализ инструментальных средств | 5% | 29.11.2022 |
| 2 | Подготовка к разработке | 10% | 06.12.2022 |
| 3 | Разработка приложения | 50% | 13.12.2022 |
| 4 | Тестирование функционала программы | 5% | 10.01.2023 |
| 5 | Создание исполняемого файла и инсталятора | 5% | 17.01.2023 |
| 6 | Подготовка расчетно-пояснительной записки | 20% | 24.01.2023 |
| 7 | Защита курсовой работы | 5% | 13.03.2023 |

5. Перечень иллюстративно-графических материалов:

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование | Количество листов |
| 1 | Иллюстрации к пояснительной записке | 17 |
| 2 | Презентация курсового проекта | 16 |
|  |  |  |
|  |  |  |

6. Исходные материалы и пособия

1 Изучаем Python, 3 е издание / Лутц М; [пер. с англ. А. Киселева]. — СПб.: Символ Плюс, 2009.

— 848 с., ил.

2 Технология разработки программных продуктов. Практикум: учеб. пособие для студ. учреждений сред. проф. образования / А. В.Рудаков, Г. Н. Федорова. — 4-е изд., стер. — М.: Издательский центр «Академия»; 2014. — 192 с. ISBN 978-5-4468-0465-8

3 Python 3 и PyQt 5. Разработка приложений. — 2-е изд., перераб. и доп. / Н. А. Прохоренок, В.А.

Дронов. — СПб.: БХВ-Петербург, 2018. — 832 с.: ил. — (Профессиональное программирование)

7. Дата выдачи задания 10.11.2022

Руководитель

(подпись)

Задание принял к исполнению

(подпись)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc129117789)

[1 Анализ инструментальных средств 6](#_Toc129117790)

[2 Подготовка к разработке 12](#_Toc129117791)

[2.1 Диаграмма переходов состояний 12](#_Toc129117792)

[2.2 Функциональная диаграмма 14](#_Toc129117793)

[2.3 Диаграмма «Сущность-Связь» 15](#_Toc129117794)

[2.4 Структурная схема 16](#_Toc129117795)

[2.5 Функциональная схема 17](#_Toc129117796)

[2.6 Диаграммы вариантов использования 18](#_Toc129117797)

[2.7 Диаграммы последовательности 20](#_Toc129117798)

[3 Разработка приложения 21](#_Toc129117799)

[3.1 Разработка базы данных 21](#_Toc129117800)

[3.2 Разработка шаблонов и макетов 25](#_Toc129117801)

[3.3 Разработка интерфейсов 28](#_Toc129117802)

[3.4 Организация связей с БД 31](#_Toc129117803)

[3.5 Функционал программы 33](#_Toc129117804)

[4 Тестирование функционала программы 38](#_Toc129117805)

[5 Создание исполняемого файла и инсталлятора 42](#_Toc129117806)

[ЗАКЛЮЧЕНИЕ 45](#_Toc129117807)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 46](#_Toc129117808)

[ПРИЛОЖЕНИЕ А. Запросы на создание таблиц и добавление в них данных 47](#_Toc129117809)

[ПРИЛОЖЕНИЕ Б. Иллюстрации форм 49](#_Toc129117810)

[ПРИЛОЖЕНИЕ В. Исходный код форм в формате .py 54](#_Toc129117811)

[ПРИЛОЖЕНИЕ Г. Исходный код программы 81](#_Toc129117812)

[ПРИЛОЖЕНИЕ Д. Исходный код тестов 119](#_Toc129117813)

[ПРИЛОЖЕНИЕ Е. Скрипт для создания инсталлятора приложения 122](#_Toc129117814)

# **ВВЕДЕНИЕ**

Организация управления книжного магазина в общем и системы заказов клиентов в частности, вручную — является довольно длительным и сложным процессом. На каждую операцию требуется свой специалист определенной категории, совершение каждой операции занимает продолжительное время, а количество ошибок, возникающих в процессе работы, оформления заказа, довольно велико, зачастую ввиду человеческого фактора.

Также, порой, необходимую книгу трудно найти, условно говоря, рядом с домом. Перспектива искать книгу, обходя абсолютно каждый книжный магазин, довольно сомнительная.

В связи с этим разработка информационной подсистемы управления заказами, как нельзя кстати подходит для решения описанных трудностей.

Разрабатываемой информационной системой будет являться программное обеспечение, работающее по четкому и слаженному алгоритму, способное обрабатывать заказы клиентов книжного магазина мгновенно, показывая какая книга есть в наличии и существенно ускоряя процесс работы системы всего книжного магазина, устраняя затраты и нивелируя ошибки персонала книжного магазина, имея на такой случай специальную систему защиты.

Главная цель курсового проекта — разработка подсистемы управления заказами клиентов книжного магазина.

При разработке подсистемы должны быть выполнены следующие задачи:

* Разработка базы данных, в которой хранятся все сведения о книгах и о доставке;
* Реализация модуля оформления заказа;
* Проведение проверки на наличие той или иной книги на складе.

1. **Анализ инструментальных средств**

На сегодняшний день имеется масса различных инструментов для разработки приложений, для работы с базой данных. Чтобы определиться с конечным вариантом используемого инструментария, необходимо провести сравнительный анализ — отобрать 2-3 программы для каждой задачи, сравнить их функционал, выбрать наиболее подходящую, удобную для разработчика. На сегодняшний день есть много различных СУБД. Их выбор обширен.

Для сравнительного анализа были выбраны СУБД Oracle Database, MySQL, PostgreSQL, так как они наиболее популярны и часто используемые в разработке и администрировании баз данных. Оценивались по следующим критериям: особенности, стоимость и максимальный размер базы данных. Результат сравнительного анализа средств СУБД представлен в таблице 1.

Таблица 1 — Cравнительный анализ средств СУБД

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | Oracle Database | MySQL | PostgreSQL |
| Особенности | Обрабатывает большие данные.  Поддерживает SQL, к нему можно получить доступ из реляционных БД Oracle.  Oracle NoSQL Database с Java/C API для чтения и записи данных. | Удобный доступ к базам данных.  Корректное распределенное хранение данных на сервере.  Быстрый поиск нужной информации в базе с помощью языка SQL.  Создание, редактирование и удаление записей, которые есть в базе.  Множественный доступ к базе с разных устройств — | Поддержка табличных пространств, а также хранимых процедур, объединений, представлений и триггеров.  Восстановление на момент времени (PITR).  Асинхронная репликация |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | Oracle Database | MySQL | PostgreSQL |
|  |  | например, из браузеров нескольких пользователей. |  |
| Стоимость | Стандартная версия $17500 | Бесплатная | Бесплатная |
| Максимальный размер базы данных | 11ГБ | Неограниченно | Неограниченно |

В результате проведенного анализа СУБД, исходя из особенностей каждого из представленных вариантов, для использования в данной работе будет использоваться СУБД MySQL, поскольку функционал данной СУБД обширен, интерфейс интуитивно понятен.

Следующий этап анализа инструментальных средства — выбор языка программирования и среды разработки самого приложения.

Для разработки приложения было выбрано три языка программирования — C#, Java и Python. Выбор конкретно данных языков обусловлен тем, что они наиболее востребованы на рынке профессий в данный момент, охватывают больше сфер разработки — от написания простых программ до создания сложных проектов с разработкой графического интерфейса и работы с базами данных. Для оценивания были выбраны следующие критерии: простота в изучении, процесс разработки программ, количество библиотек и производительность. Результат сравнительного анализа языков программирования представлен в таблице 2.

Таблица 2 — Анализ языков программирования

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | C# | Java | Python |
| Простота в изучении | Как C-типичный язык, язык довольно сложен для изучения | С-типичные конструкции языка усложняют процесс осваивания языка | Упрощенные С-типичные конструкции и простой синтаксис обеспечивают более низкий порог вхождения в язык, способствуют более скорому изучению языка |
| Разработка программ | Написание и выполнение кода возможно только в специализированной среде разработки с использованием необходимого компилятора | Разработка ведется только в специализированной IDE, либо в IDE, где реализована поддержка всех языков программирования, включая Java | Написание кода возможно, как в IDE, так и в блокноте. При установке интерпретатора Python, выполнение программы возможно даже в командной строке |
| Библиотеки | Количество библиотек очень мало | Малое количество библиотек | Большое количество библиотек |
| Производительность | Высокая | Высокая | Низкая |

По итогу анализа в качестве используемого языка программирования был выбран Python [1], поскольку он довольно прост в сравнении с другими языками программирования, а его возможности работы с различными сторонними библиотеками очень сильно упрощают и ускоряют процесс разработки и позволяют наполнить программу обширным функционалом.

Заключительный этап — выбор среды разработки.

Так как в качестве используемого языка программирования был выбран Python, необходимо выбирать и анализировать среды разработки, поддерживающие данный язык программирования, либо специализирующиеся только на данном языке программирования.

Интегрированная среда разработки (IDE) — комплекс программных средств, используемый программистами для разработки программного обеспечения.

Среда разработки включает в себя:

* Текстовый редактор;
* Транслятор (компилятор и/или интерпретатор);
* Средства автоматизации сборки;
* Отладчик.

На данный момент существует множество различных сред разработки. Выбор ограничивается лишь желанием разработчика.

Для сравнительного анализа были выбраны среды разработки Microsoft Visual Studio, Eclipse, PyCharm, так как являются наиболее популярными и часто используемыми. Проанализировав данные среды разработки, были определены преимущества и недостатки каждой IDE. Результат сравнительного анализа сред разработки представлен в таблице 3.

Таблица 3 — Сравнительный анализ сред разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | Microsoft Visual Studio | Eclipse | PyCharm |
| Описание | Премиум IDE, стоимость которой зависит от редакции и типа подписки. Она позволяет создавать самые разные проекты, начиная с мобильных и веб-приложений и заканчивая видеоиграми. Microsoft Visual Studio включает в | Бесплатная среда разработки, которая хорошо подойдет как новичкам, так и опытным разработчикам. Помимо инструментов отладки и поддержки Git/CVS, Eclipse поставляется с Java и инструментом для создания плагинов. | Кроссплатформенная среда разработки для языка программирования Python. Для разработки приложения используется язык программирования Python и среда разработки PyCharm. Данная IDE является одной из самых удобных и широко |

Продолжение таблицы 3

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | Microsoft Visual Studio | Eclipse | PyCharm |
|  | себя множество инструментов для тестирования совместимости. | Изначально Eclipse использовалась только для Java, но сейчас, благодаря плагинам и расширениям, ее функции значительно расширились. Именно из-за возможности расширить Eclipse своими модулями эта платформа и завоевала свою популярность среди разработчиков. | используемых сред разработки. |
| Преимущества | Огромная коллекция всевозможных расширений, которая постоянно пополняется.  Технология автодополнения IntelliSense.  Возможность кастомизировать рабочую панель.  Поддержка разделенного экрана (split screen). | Возможность программировать на множестве языков.  Значительная гибкость среды за счет модульности.  Возможность интеграции JUnit.  Удаленная отладка (при использовании JVM. | Специализированный под python инструмент, который умеет практически все - отладка, работа с бд, гит, автодополнения, плагины и так далее.  Есть версии для всех основных ОС (Windows/Linux/MacOS).  Есть полноценная бесплатная версия |

Продолжение таблицы 3

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | Microsoft Visual Studio | Eclipse | PyCharm |
|  |  |  | Широко используется в большинстве компаний по разработке программных продуктов. |
| Недостатки | Тяжеловесность IDE. | Новичкам может быть сложно разобраться в многообразии возможностей. | Потребление ресурсов (место на диске, оперативная память, ЦПУ). Система сложная и большая, на старом железе будет тормозить. |

В результате выбор пал на среду разработки PyCharm, так как данная IDE специализируется только на языке Python, а значит, не имеет ничего лишнего, а также обладает всеми возможностями для работы — установка различных пакетов, в том числе и для работы с базами данных и разработки полноценного приложения Windows.

1. **Подготовка к разработке**

Для создания базы данных необходимо определить основные сущности, атрибуты сущностей, а также реализовать связи между сущностями.

Сущности определяются в процессе анализа предметной области. Вычленяются ключевые элементы, определяются их свойства, устанавливаются взаимосвязи между ними.

Предметная область — Книжный магазин. Сущности и атрибуты:

* Сущность «Жанр». Атрибуты: код жанра, жанр;
* Сущность «Издательство». Атрибуты: код издательства, название издательства;
* Сущность «Автор». Атрибуты: код автора, фамилия, имя, отчество, дата рождения;
* Сущность «Книга». Атрибуты: код книги, название книги, год издания, количество страниц, стоимость, жанр, автор, издательство;
* Сущность «Клиент». Атрибуты: код клиента, фамилия, имя, отчество, дата рождения, номер телефона, почта;
* Сущность «Сотрудник». Атрибуты: код сотрудника, фамилия, имя, отчество, дата рождения, номер телефона, почта;
* Сущность «Заказ». Атрибуты: код заказа, количество книг, цена, код сотрудника, код клиента, код книги;
* Сущность «История входа». Атрибуты: код входа, дата входа, дата выхода, блокировка, сотрудник.
  1. **Диаграмма переходов состояний**

Диаграмма переходов состояний — это схема переходов и состояния, специальная техника для перехода ТЗ из одного статуса в другой. С ее помощью пользователь в наглядной форме может просматривать переход продукта из одной стадии в другую.

С помощью STD-диаграмм можно моделировать последующее функционирование системы на основе ее предыдущего и текущего функционирования. Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены [2].

Процесс реализации данной диаграммы состоит из нескольких этапов:

* Для начала необходимо определить основные состояния, взаимодействия (условия перехода), выполняемые действия, возможные переходы
* Начальным состоянием является поступление запроса на создание заказа. На схеме начальное состояние обозначается черным кружком. От начального состояния следует совершение действия на получение строки запроса. Схематично действие обозначается стрелочкой, ведущей к промежуточному состоянию;
* Промежуточным состоянием является проверка строки заказа. Исполнение процесса зависит от того, какое из трех заданных условий выполняется. Если проверены не все строки, то данный процесс повторяется до тех пор, пока все строки не будут проверены (на схеме действие обозначается круговой стрелкой исходящей и ведущей в одно место — промежуточное состояние);
* Если все строки проверены, и некоторые позиции отсутствуют на складе, от промежуточного состояния проверки строки действие переходит к состоянию ожидания. Если в процессе ожидания книга была получена, но некоторые позиции также отсутствуют на складе, данный процесс будет совершаться до тех пор, пока все позиции не будут доступны на складе. Когда все позиции доступны, программа переходит в промежуточное состояние формирования заказа книги. После совершения выдачи процесс окончательно завершается;
* В случае, если все строки проверены и все позиции доступны, процесс ожидания минуется и напрямую выполняется процесс формирования выдачи заказанной книги. После этого процесс также успешно завершается.

Результат реализации диаграммы переходов состояний представлен на рисунке 1.



Рисунок 1 — Диаграмма переходов состояний

* 1. **Функциональная диаграмма**

Функциональная диаграмма – диаграмма, отражающая взаимные связи функций, разрабатываемого программного обеспечения. Они создаются на ранних стадиях проектирования, для того чтобы помочь проектировщику выявить основные функции и по возможности обнаружить и устранить существующие ошибки.

Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, представленные в виде блоков. Детали каждого из основных блоков показаны в виде блоков на других диаграммах

Процесс реализации данной диаграммы состоит из нескольких этапов:

* Определяются функции системы и управляемая информация;
* Функции представляются блоками. Обрабатываемая информацию поступает в блок, то есть добавляется стрелка в левую часть блока, в верхнюю и нижнюю;
* Результат работы блока, то есть функции, выдается с правой стороны блока, с исходящей стрелкой, идущей в другой блок с левой стороны;
* Новый блок является дочерним для предыдущего — родительского.

В него также может поступать информация с внешней стороны (сверху или снизу), если в качестве информации выступает результат работы родительского блока, то стрелка идет в левую часть дочернего блока.

Результат реализации функциональной диаграммы представлен на рисунке 2.

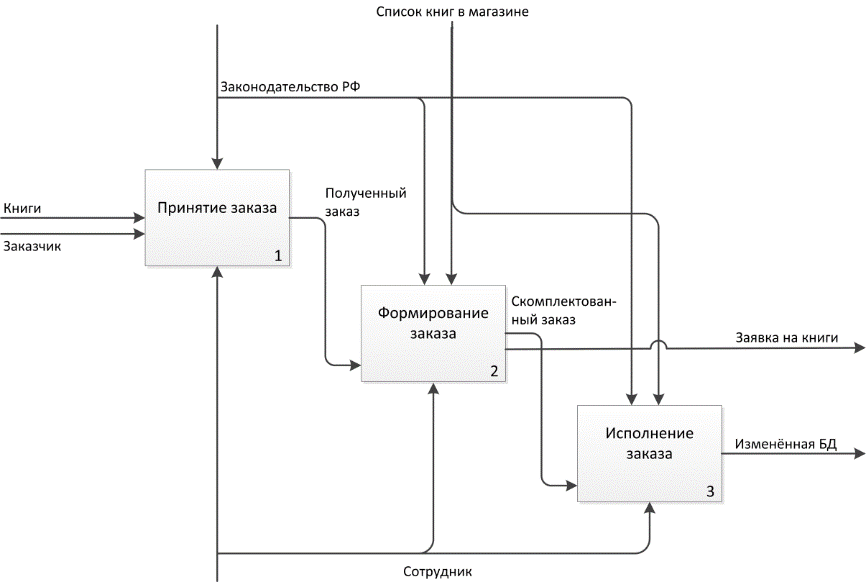


Рисунок 2 — Функциональная диаграмма

* 1. **Диаграмма «Сущность-Связь»**

Данная диаграмма — (ER-модель данных) обеспечивает стандартный способ определения данных и отношений между ними. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области.

Диаграмма «Сущность-Связь» — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса.

Пример разработки диаграммы «Сущность-Связь» описан ниже:

* Выделяются основные сущности и связи;
* Перечисляются объекты, которые могут быть полезны при проектировании данных организации, которым соответствуют сущности;
* Между сущностями назначаются связи в зависимости от их назначения;
* В зависимости от сущностей указываются типы установленных связей;

Результат реализации диаграммы представлен на рисунке 3.

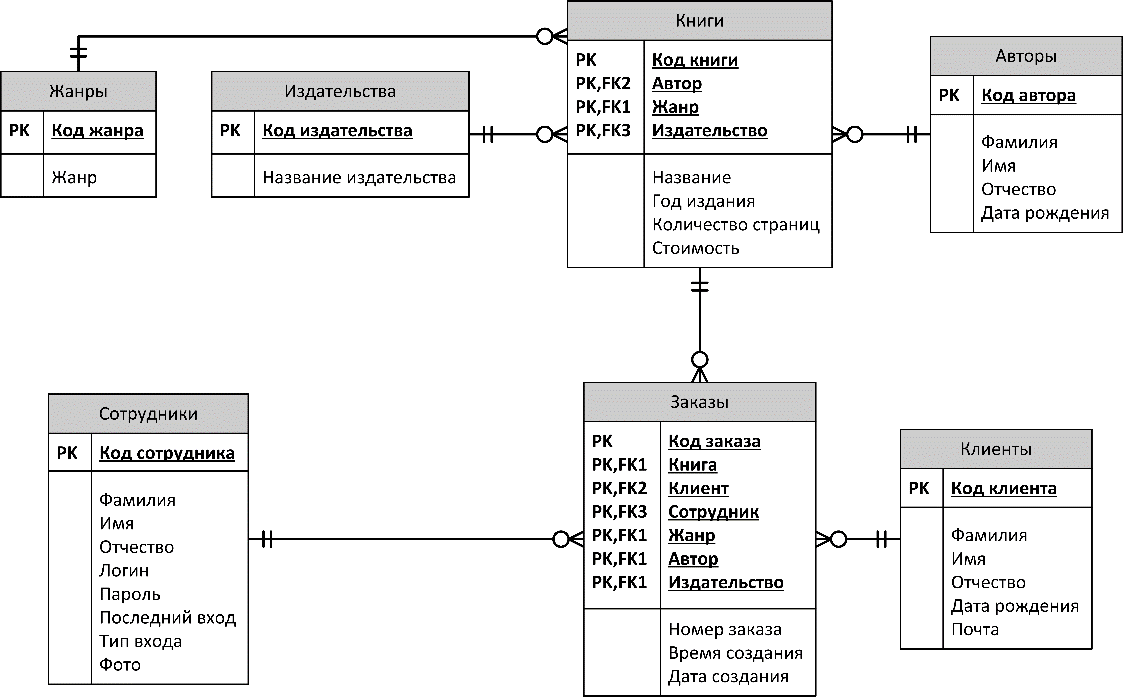


Рисунок 3 — Диаграмма «Сущность-Связь»

* 1. **Структурная схема**

Структурной называют схему, отражающую состав и взаимодействие по управлению частей разрабатываемого программного обеспечения. Структурная схема программной системы, как правило, показывает наличие подсистем или других структурных компонентов. В отличие от программного комплекса отдельные части (подсистемы) программной системы интенсивно обмениваются данными между собой и, возможно, с основной программой.

Структурные схемы пакетов программ не информативны, поскольку организация программ в пакеты не предусматривает передачи управления между ними. Поэтому структурные схемы разрабатывают для каждой программы пакета, а список программ пакета определяют, анализируя функции, указанные в техническом задании.

Самый простой вид программного обеспечения - программа, которая в качестве структурных компонентов может включать только подпрограммы и библиотеки ресурсов. Разработку структурной схемы программы обычно выполняют методом пошаговой детализации.

Структурными компонентами программной системы или программного комплекса могут служить программы, подсистемы, базы данных, библиотеки ресурсов и т. п.

Для АИС структурная схема состоит из следующих компонентов:

* Непосредственно, сама информационная система, подразделяющаяся на две подсистемы:

1. Подсистема обработки первичных документов. Данная подсистема отвечает за оформление заказов книг;
2. Подсистема ведения справочников. Задачей данной подсистемы является ведение учета выданных книг и сведений о сотрудниках.

Визуально структурная схема выглядит как граф, где главной вершиной является сама информационная система, а дочерними вершинами являются подсистема обработки первичных документов и подсистема ведения справочников, отвечающие, соответственно, за оформление заказов и за хранение сведений о книгах и сотрудниках.

Результат реализации структурной схемы представлен на рисунке 4.

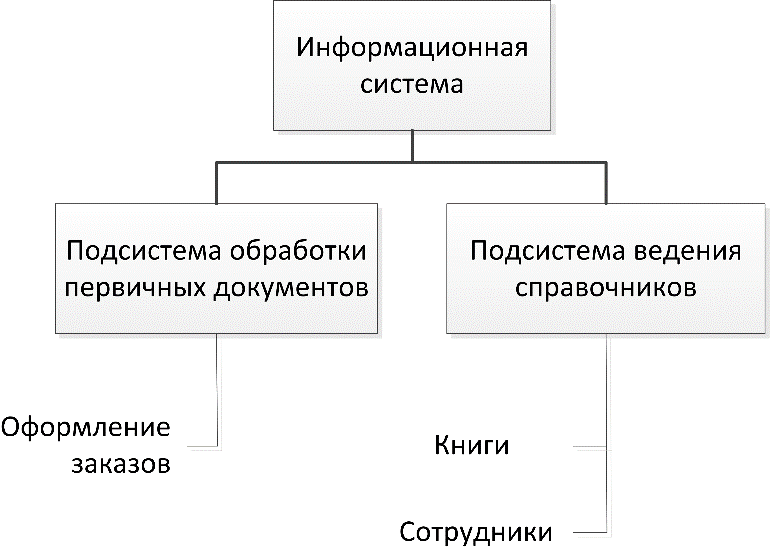


Рисунок 4 — Структурная схема

* 1. **Функциональная схема**

Функциональная схема — это схема взаимодействия компонентов программного обеспечения с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств. Для изображения функциональных схем используют специальные обозначения, установленные стандартом.

Функциональные схемы, более информативны, чем структурные. Все компоненты структурных и функциональных схем должны быть описаны. При структурном подходе особенно тщательно необходимо прорабатывать спецификации межпрограммных интерфейсов, так как от качества их описания зависит количество самых дорогостоящих ошибок. К самым дорогим относятся ошибки, обнаруживаемые при комплексном тестировании, так как для их устранения могут потребоваться серьезные изменения уже отлаженных текстов.

Для АИС функциональная схема реализуется по следующему алгоритму:

* Начало работы, то есть инициализация всех процессов. Схематично обозначается прямоугольником с закругленными краями;
* Следующий этап — ввод запроса на создание заказа. Действие схематично изображается параллелограммом;
* В зависимости от выбранного варианта действия, программа выполняет следующие функции:
  1. Происходит проверка на наличие книг на складе. Схематично изображается блоком с ограниченными полями по краям. Проверка наличия книг на складе (схематично изображается ромбов) может выдать два результата — «Да» или «Нет». В случае «Да», происходит оформление заказа на получение книги. В случае «Нет» происходит отмена заказа и завершение рабочего процесса программы;
  2. Процесс оформления заказа на получение книги. Данные о заказе сохраняются в хранилище данных (схематично обозначается цилиндр). Следом за сохранением происходит вывод сведений о заказе. Параллельно с этим программа завершает данный рабочий цикл;

Функциональная схема представлена на рисунке 5.

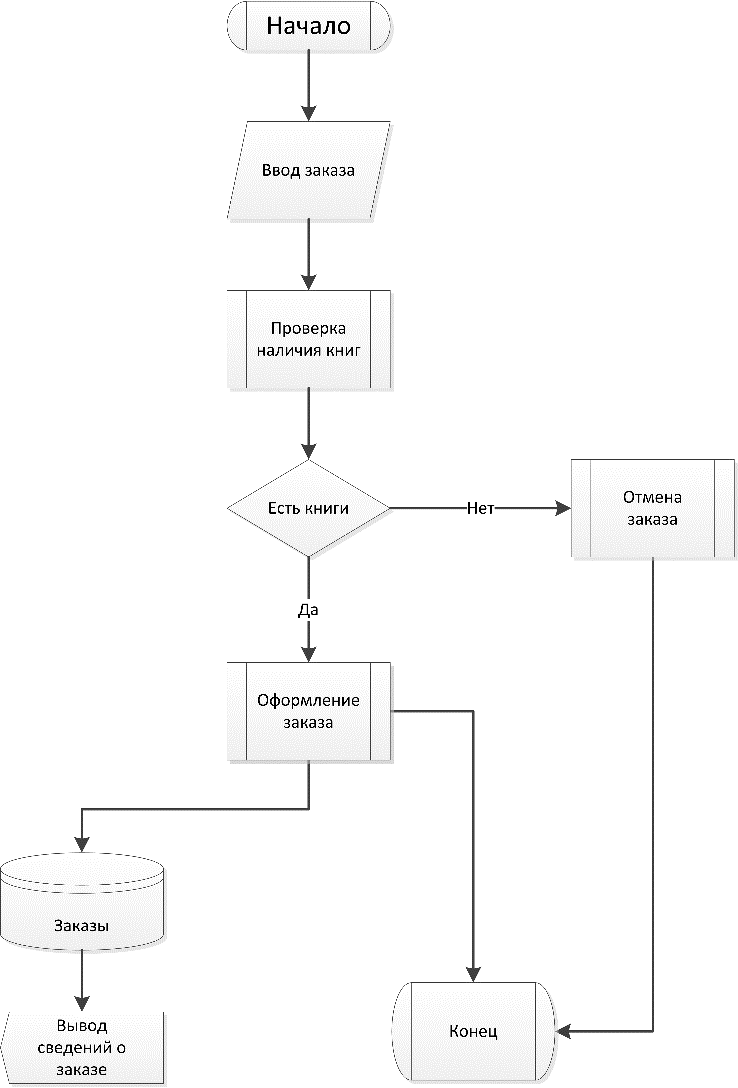


Рисунок 5 — Функциональная схема

* 1. **Диаграммы вариантов использования**

Диаграмма вариантов использования — это графическое изображение возможных взаимодействий пользователя с системой. Позволяет наглядно представить ожидаемое поведение программной системы.

Главное назначение диаграммы вариантов использования заключается в формализации функциональных требований к системе и возможности согласования полученной модели с заказчиком на ранней стадии проектирования.

Для разработки диаграммы вариантов использования соблюдать некоторую последовательность действий:

* Определить группы пользователей — «Клиент», «Сотрудник», «Администратор»;
* Определить отношения ассоциации между пользователями:

1. «Администратор» — «Изменения сведения в БД» и «Авторизация»;
2. «Сотрудник» — «Заказ книги» и «Авторизация»;
3. «Клиент» — «Заказ книги».

* «Заказ книги» включает в себя вариант использования «Определение наличия книги».
* Пользователи схематично обозначаются «фигурой человека»
* Варианты использования и отношения обозначаются овалом и линиями, указывающими взаимодействие пользователя с вариантом использования.

Реализация диаграммы вариантов использования представлена на рисунке 6.

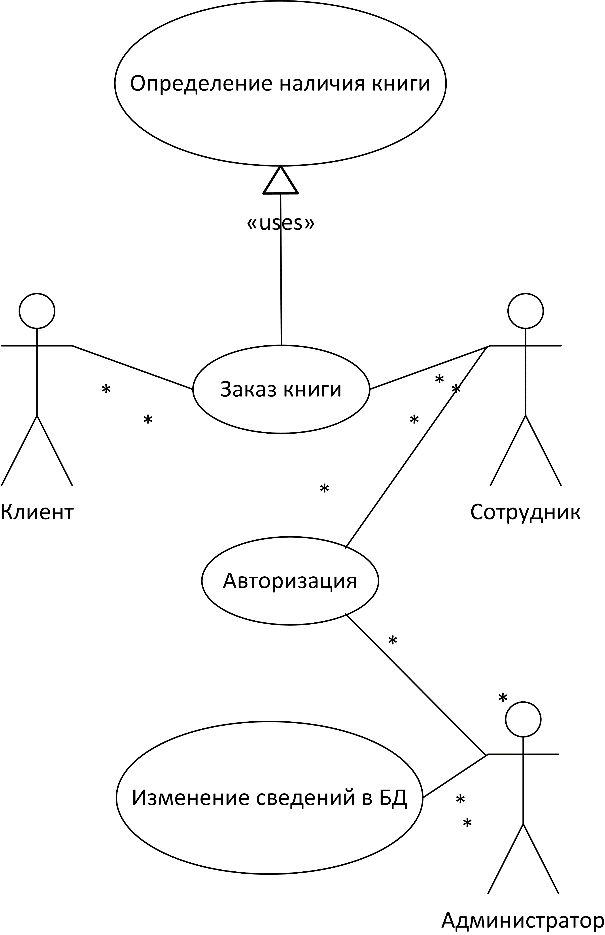


Рисунок 6 — Диаграмма вариантов использования

* 1. **Диаграммы последовательности**

Диаграмма последовательности системы — графическая модель, которая для определенного сценария варианта использования показывает динамику взаимодействия объектов во времени.

Для построения диаграммы последовательности АИС необходимо:

* Идентифицировать каждое действующее лицо — «Сотрудник», «Клиент», и система — «Книжный магазин», изобразить для каждого объекта линию жизни;
* Из описания варианта использования определить множество системных событий и их последовательность для каждого действующего лица, объекта;
* Изобразить системные события в виде линий со стрелкой на конце между линиями жизни действующих лиц и системы, а также указать имена событий и списки передаваемых значений.

Для лица «Клиент» системные события связаны с запросом на заказ книги, получением информации о книге и информации о заказе. Линии событий связаны с линией жизни лица «Сотрудник» и «Книжный магазин».

Для лица «Сотрудник» определены системные события — оформление заказа, проверка наличия и подтверждение наличия книги в магазине. Последние два события связывают лицо «Кладовщик» с объектом «Книжный магазин».

Для объекта «Книжный магазин» системным событием является принятие заказа в случае наличия книги в магазине. Данным событием связан с лицом «Клиент».

Диаграмма последовательности для автоматизированной системы представлена на рисунке 7.

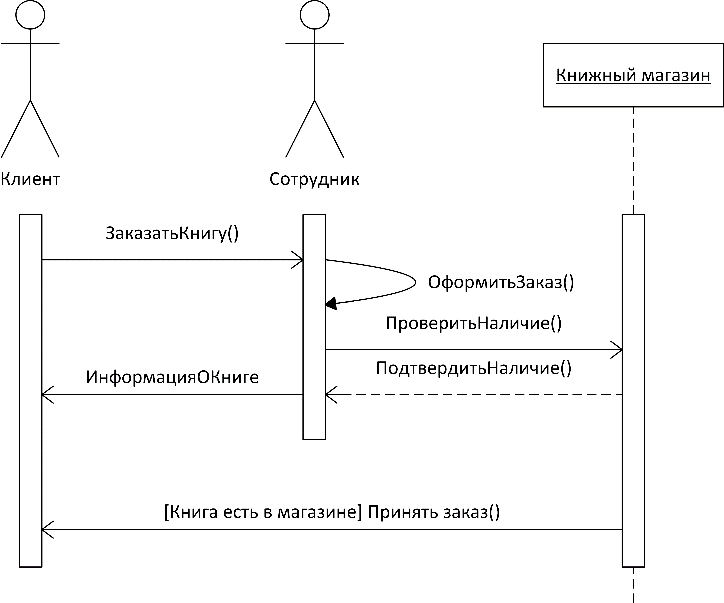


Рисунок 7 — Диаграмма последовательности

1. **Разработка приложения**
   1. **Разработка базы данных**

Для разработки базы данных, используя СУБД MySQL, необходимо провести следующие операции:

Настройка СУБД проводилась следующим образом:

1. Скачать СУБД MySQL по следующей ссылке: <https://dev.mysql.com/downloads/installer/>
2. Откроется следующее окно выбора варианта установки MySQL:

* Developer – это вариант по умолчанию, он предполагает установку всего того, что требуется разработчику, это и MySQL Server, и MySQL Workbench и другие инструменты для работы с MySQL;
* Server Only – этот вариант предполагает установку только сервера MySQL;
* Client Only – этот вариант предполагает установку только клиентской части для работы с MySQL Server, то есть серверная часть устанавливаться не будет;
* Full – установка всех включённых в дистрибутив компонентов;
* Custom – выборочная установка, в данном случае Вы можете установить только то, что нужно Вам.

Для данной работы больше всего подходит вариант установки «Developer», так как содержит в себе все необходимые элементы для разработки и администрирования базы данных.

После выбора данного варианта начинается установка MySQL.

1. В изменившемся окне выводится список всех устанавливаемых расширений для MySQL. Необходимо нажать кнопку «Execute». Результатом успешной установки свидетельствует наличие зеленых галочек рядом с названием пакета и надписей «Complete»., что и представлено на рисунке 8.

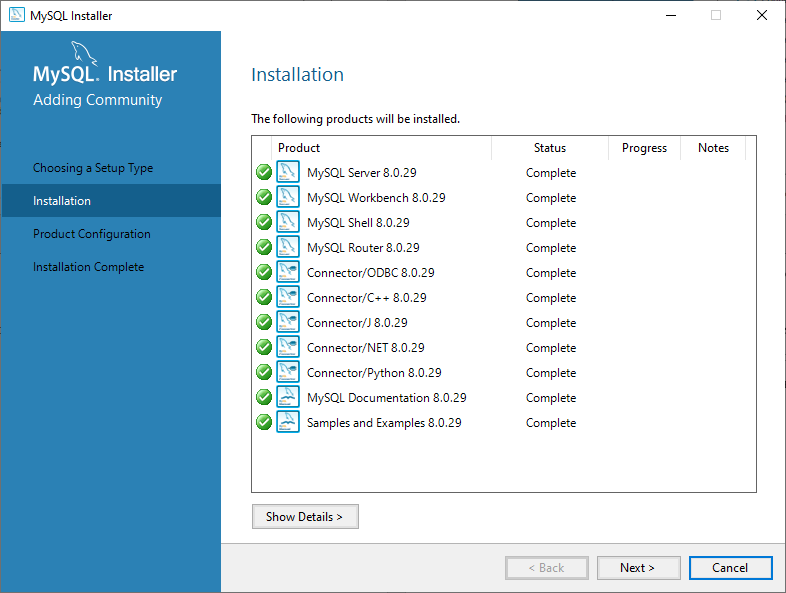


Рисунок 8 — Установка пакетов завершена

На данном этапе процесс установки СУБД переходит в настройку работы СУБД.

В открывшемся окне производится настройка соединения, назначается тип соединения, порт.

По нажатию кнопки «Next» открывается меню выбора режима аутентификации в базе посредством СУБД. Для данной работы необходимо выбрать рекомендуемый вариант «Use Strong Password Encryption for Authentication». Далее по нажатию кнопки «Next» открывается меню создания профиля для работы с сервером и БД. Для добавления нового профиля, необходимо нажать на кнопку «Add User», где указывается имя профиля и пароль, назначается его роль. Меню создания профиля показано на рисунке 9.

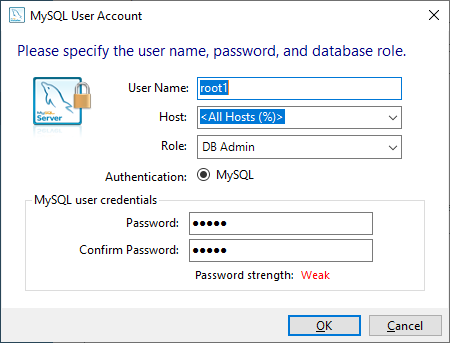


Рисунок 9 — Продолжение настройки профиля

Требуется ввести имя пользователя, указать хост, определить роль, ввести дважды пароль. По умолчанию имя пользователя определено как “root1”, пароль — “admin”. Указывается аутентификация для MySQL.

Следующий шаг — реализация запросов на создание таблиц и заполнение их данными. Для начала необходимо создать саму базу, используя SQL-запрос:

“CREATE DATABASE IF NOT EXISTS bookshop”

Результатом будет появившаяся схема, или же база данных, представленная на рисунке 10.

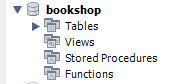


Рисунок 10 — Схема базы данных “bookshop”

Далее создаются таблицы [3]. Ниже представлены SQL-запросы на создание таблиц:

1. Создание таблицы «Автор»:

CREATE TABLE `author` (

`Код автора` int NOT NULL AUTO\_INCREMENT,

`Фамилия` varchar(45) DEFAULT NULL,

`Имя` varchar(45) DEFAULT NULL,

`Отчество` varchar(45) DEFAULT NULL,

`Дата рождения` datetime DEFAULT NULL,

PRIMARY KEY (`Код автора`)

);

1. Создание таблицы «Книга»:

CREATE TABLE `books` (

`Код книги` int NOT NULL AUTO\_INCREMENT,

`Наименование книги` varchar(45) DEFAULT NULL,

`Стоимость` varchar(45) DEFAULT NULL,

`Жанр` int DEFAULT NULL,

`Автор` int DEFAULT NULL,

`Издательство` int DEFAULT NULL,

CONSTRAINT `books\_FK` FOREIGN KEY (`Жанр`) REFERENCES `genre` (`Код жанра`),

CONSTRAINT `books\_FK\_1` FOREIGN KEY (`Автор`) REFERENCES `author` (`Код автора`),

CONSTRAINT `books\_FK\_2` FOREIGN KEY (`Издательство`) REFERENCES `publishing\_house` (`Код издательства`)

);

Остальные запросы на создание таблиц по структуре идентичны представленным выше и приведены в приложении А. Результат выполнения запросов на создание таблиц представлен на рисунке 11:

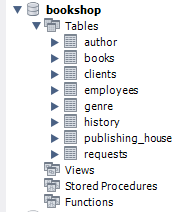


Рисунок 11 — Создана база данных и таблицы

Благодаря функционалу MySQL Workbench у разработчиков есть возможность добавлять данные несколькими способами:

* Написание SQL-запросов в поле ввода текста;
* Ручное добавление данных в ячейки таблиц базы данных;
* Импорт данных в БД из стороннего файла.

В качестве способа добавления данных в БД был выбран способ импорта данных из стороннего файла, поскольку, по сравнению с другими способами, не требует слишком много времени и интуитивно понятен. Для этого необходимо открыть меню со всеми таблицами, и выбрать ту таблицу, в которую будет производиться добавление данных. В качестве примера, выбрана таблица «Клиент». Необходимо нажать по названию таблицы и выбрать функцию «Table data import wizard». Далее в открывшемся окне указывается путь к csv файлу, предварительно созданному при использовании Microsoft Excel. Затем необходимо указывать, создавать ли новую таблицу, или же выбрать уже существующую. На рисунке 12 представлен результат импорта данных.



Рисунок 12 — Результат добавления данных

Важнейшей частью данной работы является разработка Windows-приложения для взаимодействия с базой данных.

Главная цель разработки приложения — обеспечить вывод всех требуемых данных из базы данных, реализовать возможность добавления новых данных: новых картин, новых художников, новых посетителей. Реализовать форму авторизации как средство разделения уровня доступа к базе данных.

* 1. **Разработка шаблонов и макетов**

Для разработки приложения необходимо изначально спроектировать макеты форм интерфейса приложения:

* Макет формы главного окна приложения;
* Макет формы авторизации;
* Макет вывода данных;
* Макет ввода данных;
* Макет формы добавления нового клиента.

На рисунке 13 представлен разработанный макет формы авторизации. В верхней части макета указан заголовок. Ниже расположены поля для ввода логина и пароля, а также кнопка для скрытия пароля и обновления капчи, окно для вывода капчи, кнопка входа в систему.



Рисунок 13 — Макет формы авторизации

Затем проектируется макет формы главного окна приложения. Результат представлен на рисунке 14. В верхней части макета указан заголовок, во вторичном окне выводятся данные, ниже расположены кнопки перехода по формам.



Рисунок 14 — Макет формы главного окна приложения

Следующий макет — макет формы окна вывода данных. Представлен на рисунке 15. В верхней части макета указан заголовок, ниже выводятся данные, в нижней части расположены кнопки добавления, удаления, сохранения данных.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 15 — Макет формы окна вывода данных

Далее следует проектирование макета окна вывода данных без кнопок. Представлен на рисунке 16. В верхней части представлен заголовок. Ниже расположено поле для вывода данных из базы данных.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16 — Макет формы окна вывода данных

Следующий макет — макет формы окна добавления данных. Представлен на рисунке 17. В верхней части формы расположен заголовок, ниже — поля для ввода данных и кнопки для выполнения различных функций.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 17 — Макет формы окна добавления данных

Несколько форм окон добавления и вывода данных могут быть реализованы по одному макету, посколько существенных различий не имеют.

* 1. **Разработка интерфейсов**

Для создания форм интерфейса используется инструмент QtDesigner [4] из пакета pyqt5-tools. Данный инструмент необходим при разработке форм интерфейса Windows-приложения. Он позволяет создавать окна для вывода любой необходимой информации, создавать кнопки и назначать им определенные функции, обеспечивать возможность ввода данных с клавиатуры, представлять данные в табличном виде, а также работать с графическими элементами.

Для реализации всех форм используются следующие объекты QtDesigner:

* QPushButton — кнопка, совершающая назначенное действие;
* QMainWindow — класс главного окна;
* QWidget — класс виджета, всплывающего окна;
* QTableWidget — для вывода данных в табличном виде;
* QListWidget — для вывода данных в виде списка.

Алгоритм создания формы в QtDesigner на примере формы авторизации:

1. Так как форма авторизации является диалоговым окном, необходимо выбрать «Dialog without Buttoms» при выборе типа формы.
2. Далее из списка элементов WidgetBox необходимо переместить все используемые в форме элементы:

* QLineEdit — для ввода данных с клавиатуры;
* QLabel — для установления текста в конкретной области формы;
* QStackedWidget — для перехода по формам (страницам) программы;
* QGraphicsView — для отображения капчи
* QPushButton — для кнопок, выполяющих определенные действия
* QHBoxLayout – для группировки виджетов горизонтально
* QVBoxLayout– для группировки виджетов вертикально

1. Чтобы сгруппировать все объекты используются элементы QHBoxLayout и QVBoxLayout.
2. Чтобы изменить название виджета, выбирается нужный виджет по нажатию левой кнопки мыши, в свойствах объекта (property editor) внести изменения в поле «objectName».
3. Для изменения минимального или максимального размера элемента необходимо по нажатию левой кнопки мыши выбрать нужный виджет, в свойствах (property editor) изменить параметры «minimumSize» (width, height) или «maximumSize» (width, height). Для установления фиксированного размера элемента в параметрах «minimumSize» и «maximumSize» для wight и height указываются одинаковые значения
4. Чтобы установить изображение в QLabel, необходимо изменить поле «pixmap», указав путь к файлу изображения.
5. Для установки иконка окна изменяется поле «windowIcon», с указанием пути до файла изображения.

Форма авторизации представлена на рисунке 18.

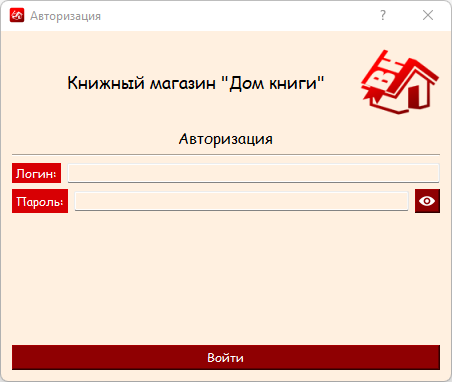


Рисунок 18 — Форма авторизации

В зависимости от должности сотрудника, в программе реализовано разделение уровня доступа к содержимому базы данных. На рисунке 19 показана форма окна информации о пользователе.

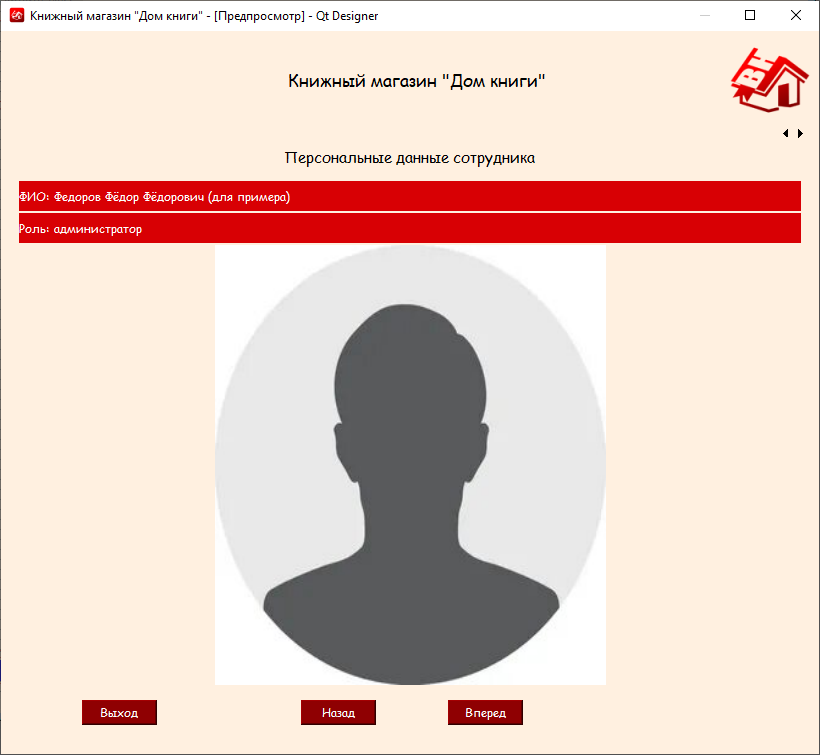


Рисунок 19 — Форма окна информации о пользователе

В зависимости от должности сотрудник видит только ту информацию, которая ему предоставлена. Роль администратора позволяет просматривать следующие формы:

* Просмотр истории входа сотрудников;
* Просмотр жанров. В этой же форме реализована возможность добавить новый жанр или же удалить выбранный жанр и сохранить все изменения в базу данных;
* Просмотр книг. Также реализована возможность просмотра всех книг, добавления новых и удаления имеющихся, сохранения изменений;
* Просмотр авторов. Возможно добавить нового автора, удалить существующего, сохранить изменения;
* Просмотр издательств. Также есть возможность добавить издательство для книги.

Для старшего смены и продавца реализованы следующие формы:

* Создание и сохранение заказа. Указывается номер заказа, дата заказа, время заказа, клиент и книга. Также реализована возможность создания штрих-кода;
* Добавление нового клиента. Открывается окно создания нового клиента. Указывается фамилия, имя, отчество посетителя, дата рождения, Email;
* Просмотр всех клиентов. Выводится список всех клиентов.

Формы окон приложения представлены в приложении Б. Листинг форм в формате .py представлен в приложении В. Листинг графического модуля программы представлен в приложении В.

* 1. **Организация связей с БД**

Реализовать модуль для взаимодействия с базой данных необходимо для полноценной работы программы.

Сам модуль реализован на языке Python с использованием библиотеки “mysql” для связи с созданной БД в СУБД MySQL. В модуле обеспечивается подключение к локальному серверу, на котором развернута база данных, с помощью следующего кода:

“self.conn = mysql.connector.connect(host='localhost', port=3306, user='root', database=`bookshop`)”

Где host — название соединения/сервера, port — номер порта соединения, user — имя пользователя, созданного при первой настройке СУБД MySQL (по-умолчанию — root), password — пароль пользователя, database — название базы данных.

Взаимодействие модулей графического интерфейса и базы данных осуществлено через модуль “facade”, основанном на одноимённом паттерне проектирования [5]. Листинг данного модуля представлен в приложении В.

При авторизации в системе выполняется запрос на получение данных о пользователе по введенному логину. Делается это для того, чтобы реализовать разделенный доступ к базе данных в целях безопасности. Одновременно с этим выполняется запрос на сохранение в историю входа времени и даты последней авторизации.

В окне просмотра истории входа, в результате выполнения запроса, выводятся «ID», «Время входа» и «Логин сотрудника» для всех записей об авторизации:

* Получение информации о сотруднике по его логину;
* Получение списка логинов сотрудников;
* Добавление записи истории входа;
* Получение истории входа.

Так как в программе присутствует разделение уровня доступа к базе данных, должности администратора, старшего смены и продавца имеют различные полномочия.

Для пользователя «Администратор» реализованы следующие запросы:

В окне просмотра книг реализованы запросы:

* Удаление книги;
* Сохранение книги;
* Изменение информации о книге;
* Вывод на экран всех книг.

В окне добавления книги реализован запрос:

* Добавление книги.

В окне просмотра всех авторов реализованы следующие запросы:

* Вывод на экран всех авторов;
* Изменение автора;
* Удаление автора;
* Сохранение автора.

В окне добавления нового автора реализован запрос:

* Добавление автора.

В окне просмотра жанров книги реализуются следующие запросы:

* Добавление жанра книги;
* Изменение жанра книги;
* Получение списка жанров книги;
* Удаление жанра книги;
* Вывод на экран списка всех жанров книги.

В окне просмотра издательств реализуются следующие запросы:

* Изменение издательства;
* Добавление издательства;
* Удаление издательства;
* Получение списка издательств.

Для пользователя «Старший смены» и «Продавец» реализованы следующие запросы:

В окне создания заказа реализованы следующие запросы:

* Добавления книги к заказу;
* Получение списка книг;
* Получение кода клиента;
* Получение списка клиентов;
* Получение списка жанров;
* Получение id клиента;
* Сохранение заказа;
* Генерация штрих-кода.

В окне добавления клиента реализован следующий запрос:

* Добавление клиента.

В окне просмотра всех клиентов реализован следующий запрос:

* Получение списка клиентов.

Листинг модуля базы данных представлен в приложении В.

* 1. **Функционал программы**

Для реализации данной системы был использован один из самых используемых подходов к программированию — объектно-ориентированное (ООП).

Суть данного подхода заключается в том, что каждый модуль программы является отдельным классом, функционал модуля — это метод класса, а свойства модуля — атрибуты данного класса.

При разработке программы было реализовано несколько модулей — классов.

Для модуля, обеспечивающего работу с базой данных, создан класс Database. Он используется библиотеку языка Python под названием «mysql» для обеспечения работы с СУБД MySQL и налаживания связи с самой базой данных. Ниже перечислены методы данного класса:

* \_\_init\_\_(self) — подключение к базе данных;
* insert\_book(self, name, year, lists, cost, genre, author, ph) — добавление новой книги;
* insert\_request(self, number, date, time, client, book) — добавление нового заказа;
* update\_book(self, id, name, year, lists, cost, genre, author, ph) — обновление книги;
* delete\_book(self, id) — удаление книги;
* select\_clients(self) — получение списка клиентов;
* select\_employees(self) — получение списка сотрудников;
* select\_books(self) — получение списка книг;
* get\_info(self, login) — получение информации о сотруднике по его логину для авторизации;
* get\_logins(self) — получение списка логинов сотрудников;
* get\_code\_client(self, surname, name, lastname) — получение кода клиента;
* get\_clients(self) — получение списка ФИО клиентов;
* insert\_client(self, surname, name, lastname, dateOfBirth, email) — добавление нового клиента;
* get\_books(self) — получение cписка наименований книг;
* get\_book\_id(self, name) — получение кода книги;
* get\_client\_id(self, surname, name, lastname) — получение кода клиента;
* insert\_time\_entry(self, login, time, success) — добавление времени входа сотрудника (для истории входа);
* insert\_time\_exit(self, login, time, block) — добавление времени выхода сотрудника (для истории входа);
* select\_history(self) — получение истории входа сотрудников;
* get\_date\_book(self) — получение данных для создания отчётов.

Следующий модуль — модуль графического интерфейса. С использованием библиотеки PyQt5 для языка Python было реализовано несколько классов данного модуля.

В процессе проектирования было установлено, что модуль графического интерфейса будет иметь несколько разделов для каждой функции. В связи с этим и проводилась разработка нескольких классов для каждого окна интерфейса

Для основного окна интерфейса программы разработан класс MainWindow(QMainWindow). Ниже перечислены методы данного класса:

* \_\_init\_\_(self) — отвечает за подключение к кнопкам, объявление переменных, заполнение таблиц, получение списка страниц StackedWidget;
* exit(self, block) — отвечает за выход из программы;
* page\_all\_clients(self) — отвечает за переход к странице с таблицей клиентов. Обновление таблицы клиентов;
* updateTableClient(self) — отвечает за обновление таблицы клиентов;
* updateTableBook(self) — отвечает за обновление таблицы книг;
* updateTableHistory(self) — отвечает за обновление таблицы истории входа;
* new\_book(self) — добавление новой книги;
* delete\_book(self) — удаление книги;
* get\_from\_table\_book(self) — получение данных из таблицы, чтобы потом записать их в БД;
* save\_book(self) — сохранение данных о книгах в базу данных. Обновление таблицы в интерфейсе;
* build\_combobox\_clients(self) — добавление списка клиентов в ComboBox;
* build\_combobox\_books(self) — добавление списка книг в ComboBox;
* add\_new\_request(self) — оформление нового заказа и его показ в ListWidget;
* add\_book\_to\_request(self) — добавление книги в заказ;
* save\_request(self) — отвечает за сохранение заказа в базу данных;
* generateCode(self) — отвечает за создание штрих-кода по номеру, дате и времени заказа. Создаётся в форматах .png и .pdf;
* mes\_box(self, text) — открывает messagebox с переданным текстом. Вызывается при успешном создании кода;
* otchot(self) — выборка данных для создания отчётов;
* count\_order(self) — функция создания отчёта "Количество заказов по дням за период времени";
* count\_book(self) — функция создания отчёта "Количество проданных книг по дням за период времени";
* count\_order\_book(self) — функция создания отчёта "Количество заказов по дням за период времени по каждой книге";
* order\_pdf(self, type, data) — создание отчёта в формате pdf;
* next\_page(self) — отвечает за переход к следующей странице;
* back\_page(self) — отвечает за переход к предыдущей странице;
* open\_auth(self) — создает и показывает диалоговое окно авторизации. Вызывается в \_\_init\_\_ и в функции exit;
* open\_new\_client(self) — создает и показывает диалоговое окно создания нового клиента.

Для окна авторизации разработан класс DialogaAuth(QDialog). Ниже перечислены методы данного класса:

* \_\_init\_\_(self, parent=None) — отвечает за подключением к кнопкам, объявление переменных, создание сцены для «graphicsView»;
* vis\_pas(self) — вызывается при нажатии на кнопку «btn\_hide\_password». Скрывает и показывает пароль (в соответствии с переменной self.vis\_p);
* visible\_captcha(self, visible=True) — вызывается в \_\_init\_\_ (с параметром False) и при второй неуспешной попытки входа неправильный ввод пароля или логина) с параметом True. При False скрывает поле ввода, кнопку обновления и сцену для отрисовки капчи При True - показывает поле ввода, кнопку обновления и сцену для отрисовки капчи;
* captcha\_generation(self) — вызывается при второй неуспешной попытке входа и при нажатии на кнопку «btn\_new\_captcha». Выводит капчу в «graphicsView» и возвращает значение капчи в переменной self.now\_captcha;
* mes\_box(self, text) — открывает messagebox с переданным текстом. Вызывается при неверном вводе пользователем логина, пароля, капчи;
* enter(self) — вызывается при нажатии на кнопку btn\_enter. Обрабатывает все случаи ввода данных (капчи, логина, пароля) и считает неуспешные попытки входа. Проверяет есть ли у пользователя блокировка и до скольки она длится. При успешном входе передает в фасад время и логин успешного входа (для записи в бд), записывает индексы доступных страничек «Stacked Widget» (у разных сотрудников могут быть разные странички).

Для окна добавления нового клиента был разработан класс DialogNewClient(QDialog). Ниже перечислены его методы:

* \_\_init\_\_(self, parent=None) — отвечает за подключением к кнопке "Добавить";
* add(self) — отвечает за добавление клиента в базу данных;
* mes\_box(self, text) — открывает messagebox с переданным текстом. Вызывается, если какое-либо поле не заполнено.

Для взаимодействия модуля графического интерфейса с модулем базы данных был разработан смежный модуль — класс Facade, обеспечивающий передачу данных одного модуля другому. Метода данного класса представлены ниже:

* \_\_init\_\_(self) — создание объекта базы данных;
* get\_logins(self) — получение списка логинов сотрудников;
* get\_code\_client(self, surname, name, lastname) — получение кода клиента;
* get\_clients(self) — получение списка клиентов;
* get\_books(self) — получение списка книг;
* get\_book\_id(self, name) — получение кода книги;
* get\_id\_client(self, surname, name, lastname) — получение кода клиента;
* get\_for\_authorization(self, login) — получение информации о сотруднике;
* insert\_book(self, name, year, lists, cost, genre, author, ph) — добавление книги;
* delete\_book(self, id) — удаление книги;
* update\_book(self, id, name, year, lists, cost, genre, author, ph) — обновление книги;
* create\_request(self, number, date, time, client, book) — создание заказа;
* read\_clients(self) — получение списка клиентов;
* read\_history(self) — получение списка истории входа;
* insert\_client(self, surname, name, lastname, dateOfBirth, email) — добавление нового клиента;
* read\_books(self) — получение списка книг;
* insert\_time\_entry(self, login, time, success) — добавление времени входа сотрудника;
* insert\_time\_exit(self, login, time, block) — добавление времени выхода сотрудника;
* get\_date\_book(self) — получение даты создания заказа.

Для сборки всех модулей в единую систему был разработан класс Builder на основе одноименного паттерна «строителя». Это порождающий паттерн проектирования, который позволяет создавать сложные объекты пошагово. Методы описаны ниже:

* \_\_init\_\_(self) — инициализация приложения, главного окна интерфейса и окна авторизации;
* auth(self) — авторизация в приложении и вызов главного окна.

1. **Тестирование функционала программы**

Тестированием называется выполнение программы в целях обнаружения ошибок. Отладкой называется локализация и исправление ошибок. Тесты разрабатывают в определенном порядке. При этом по внешней спецификации различают тесты:

* для каждого класса входных данных;
* для граничных и особых значений входных данных.

Контролируется, все ли классы выходных данных при этом проверены, и добавляются при необходимости нужные тесты.

Для тестирования функционала информационной системы «Книжный магазин», с использованием библиотеки unittest, были реализованы два класса — TestPush(TestCase), для тестирования всего функционала добавления данных, и TestDeleteAndSave(TestCase) — для тестирования функций удаления и сохранения данных.

Для класса TestPush() реализованы следующие модули — тесты конкретных функций:

* setUp() — инициализация класса, подключение ко всем модулям и класса для тестирования их функций;
* test\_push\_client() — тестирование функции добавления клиента. Имитируется ввод данных о клиенте (заданными значениями «test\_test») и нажатие кнопки «btn\_add» — «Добавить»;
* test\_push\_genre() — тестирование функции добавления жанра. Имитируется ввод данных о жанре (тестовые данные — «test\_genre» для наименования) и нажатие кнопки «btn\_add» — «Добавить»;
* test\_push\_author() — тестирование функции добавления автора. Имитируется ввод данных о поставке (заданными значениями «test\_author») и нажатие кнопки «btn\_add» — Добавить;
* test\_push\_ph() — тестирование функции добавления издательства. Имитируется ввод данных о издательстве — его наименования (тестовое значение для наименования — «test\_ph») и нажатие кнопки «btn\_add» — «Добавить»;
* test\_push\_book() — тестирование функции добавления книги. Имитируется ввод данных о книге (тестовое значение для наименования — «test\_book») и нажатие кнопки «btn\_add» — «Добавить».

Для класса TestDeleteAndSave() реализованы следующие модули — тесты конкретных функций:

* setUp() — инициализация класса, подключение ко всем модулям и класса для тестирования их функций;
* test\_delete\_genre() — тестирование функции удаления жанра. Имитируется выбор строки в таблице, затем нажатие кнопки «btn\_delete\_genre» — «Удалить»;
* test\_delete\_author() — тестирование функции удаления автора. Имитируется выбор строки в таблице, затем нажатие кнопки «btn\_delete\_author» — «Удалить»;
* test\_delete\_ph() — тестирование функции удаления издательства. Имитируется выбор строки в таблице, затем нажатие кнопки «btn\_delete\_ph» — «Удалить»;
* test\_delete\_book() — тестирование функции удаления книги. Имитируется выбор строки в таблице, затем нажатие кнопки «btn\_delete\_book» — «Удалить».

Результаты проведения тестирования функционала представлены в таблице 4.

Таблица 4 — Результаты тестирования функционала

|  |  |  |
| --- | --- | --- |
| Название теста | Входные данные | Результат |
| test\_push\_client | В интерфейс приложения введены данные о клиенте (фамилия, имя, отчество, дата рождения) — “test\_test”.  Левый клик мыши по кнопке btn\_add\_client — «Добавить». | Добавление записи о клиенте в базу данных. |
| test\_push\_genre | В интерфейс приложения введено название жанра — “test\_genre”.  Левый клик мыши по кнопке btn\_add\_genre — «Добавить». | Добавление записи о жанре в базу данных. |
| test\_push\_author | В интерфейс приложения введены данные о клиенте (фамилия, имя, отчество, дата рождения) — “test\_test”. | Добавление записи об авторе в базу данных. |

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| Название теста | Входные данные | Результат |
|  | Левый клик мыши по кнопке btn\_add\_author — «Добавить». |  |
| test\_push\_ph | В интерфейс приложения введено название издательства — “test\_ph”.  Левый клик мыши по кнопке btn\_add\_ph — «Добавить». | Добавление записи об издательстве в базу данных. |
| test\_push\_book | В интерфейс приложения введено название книги — “test\_book”.  Левый клик мыши по кнопке btn\_add\_book — «Добавить». | Добавление записи о книге в базу данных. |
| test\_delete\_genre | Левый клик мыши по последней записи.  Левый клик мыши по кнопке btn\_delete\_genre — «Удалить».  Левый клик мыши по кнопке btn\_save\_genre — «Сохранить». | Удаление записи о жанре из базы данных. |
| test\_delete\_author | Левый клик мыши по последней записи.  Левый клик мыши по кнопке btn\_delete\_author — «Удалить».  Левый клик мыши по кнопке btn\_save\_author — «Сохранить». | Удаление записи об авторе из базы данных. |

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| Название теста | Входные данные | Результат |
| test\_delete\_ph | Левый клик мыши по последней записи.  Левый клик мыши по кнопке btn\_delete\_ph — «Удалить».  Левый клик мыши по кнопке btn\_save\_ph — «Сохранить». | Удаление записи об издательстве из базы данных. |
| test\_delete\_book | Левый клик мыши по последней записи.  Левый клик мыши по кнопке btn\_delete\_book — «Удалить».  Левый клик мыши по кнопке btn\_save\_bool — «Сохранить». | Удаление записи о книге из базы данных. |

По результатам тестирования функционала программы ошибки не обнаружены.

1. **Создание исполняемого файла и инсталлятора**

Создание инсталляции программного продукта позволяет автоматизировать процесс установки программного продукта на компьютеры пользователей, предоставляя им при этом возможность выбора различных сценариев установки и обеспечивая корректность его дальнейшей работы. Кроме того, у разработчиков программного продукта появляется хорошая возможность запретить несанкционированные установки.

Первоначальным этапом в создании инсталлятора программы является компиляция кода программы в исполняемый файл с расширением “.exe”. Для этого в языке программирования Python есть специальный модуль — PyInstaller.

Процесс создания файла “.exe” описан ниже:

1. Открыть командную строку и перейти в директорию с проектом, введя команду — «cd \*путь до папки с проектом\*»;
2. Находясь в папке с проектом, ввести команду — «pyinstaller -m “gui.py”». Начнется процесс компиляции скрипта Python в исполняемый файл формата “.exe”;
3. Pyinstaller анализирует файл и создает:

* Файл gui.spec в том же каталоге, что и gui.py;
* Папку сборки в том же каталоге, что и gui.py;
* Папку dist в том же каталоге, что и gui.py;
* Файлы журналов в папке сборки;
* Само приложение можно будет найти вложенным в папку dist.

Результат представлен на рисунке 20.

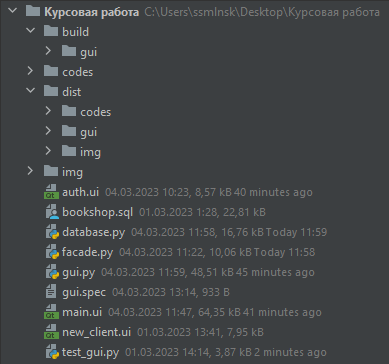


Рисунок 20 — Результат компиляции скрипта “gui.py”

После того, как исполняемый файл создан, необходимо создать инсталлятор самой программы. Для создания инсталлятора используется программа Inno Setup Compiler.

Создание инсталлятора с помощью Inno Setup происходит следующим образом:

1. При запуске программы закрыть окно “Welcome”, а затем в меню “File” выбрать опцию “New”;
2. Откроется мастер создания инсталлятора Inno Setup и нужно нажать кнопку “Next”;
3. Следующее окно — меню настроек приложения, где вводится название приложения, его версия, издатель и его сайт. После ввода информации произвести нажатие кнопки «Next»;
4. В следующем окне настройки остаются без изменений;
5. По нажатию кнопки «Next» открывается окно указания пути к исполняемому файлу, а также к различным ресурсам для добавления их к инсталлятору проекта:

* В поле “Application main executable file” указывается путь к проекту;
* В поле “Other application files” указываются дополнительные файлы необходимые для работы программы.

1. В последующих двух окнах необходимо снять галочку с полей “Associate a file type to the main executable” и “Create a shortcut to the main executable in the Start Menu Programs folder”. По нажатию на кнопку “Next” перейти к следующему окну;
2. В окне программной документации указываются пути до лицензионных файлов и инструкций к установке;
3. В остальных окнах осуществляется выбор режима установки. Как правило, следует выбирать вариант “Administrative install mode”. Происходит выбор языка установки, а также указывается папка, в которую сохраняется созданный инсталлятор. Указывается наименование инсталлятора, его иконка и пароль (при необходимости);
4. Последним этапом является выбор — использовать ли директивы “#define”. Необходимо снять галочку с поля “Yes, use #define compiler directives” и нажать кнопки “Next” и“Finish”.

По завершении установки в указанной выше папке появится только что созданный инсталлятор программы. Результат представлен на рисунке 21.

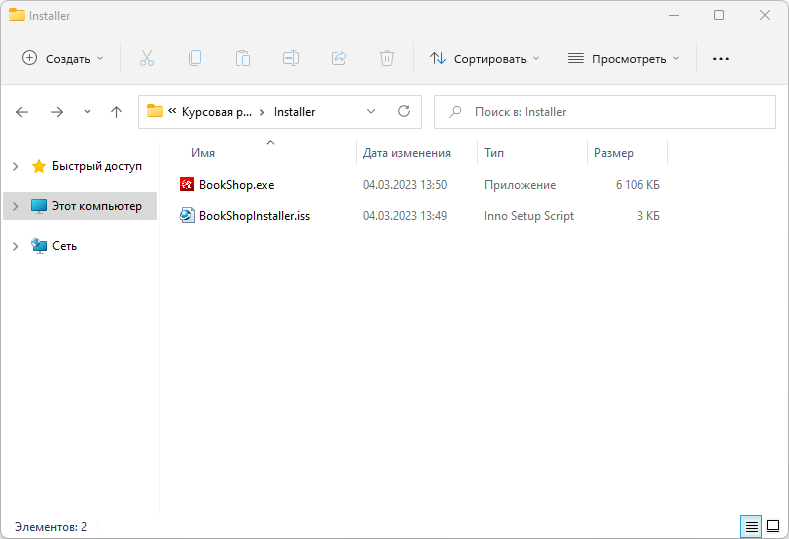


Рисунок 21 — Созданный инсталлятор

**ЗАКЛЮЧЕНИЕ**

В результате проведенной работы была достигнута цель — разработка информационной подсистемы управления заказами книжного магазина, а также были выполнены поставленные задачи:

1. Изучение предметной области, реализация концептуальной, логической и физической модели базы данных;
2. Настройка СУБД для работы с базой данных, размещение базы на локальном сервере. Разработка и настройка базы данных;
3. Разработаны следующие модули для работы приложения:
   * Модуль для взаимодействия с базой данных. Реализованы операции добавления, удаления, сохранения записей, а также получения данных из таблиц;
   * Модуль графического интерфейса – для полного взаимодействия с функционалом программы. Реализована возможность просмотра персональных данных сотрудника, создание заказа и отчётов, редактирование данных в таблицах, просмотр таблиц;
   * Модуль, основанный на шаблоне “Facade”, позволяющий собрать функции всех модулей воедино, для более удобного проектирования;
   * Было разработано и проведено тестирование программы. Целью тестирования была проверка работоспособности каждого модуля программы на корректность;
4. Написание отчета по проведенной работе с подробным описанием всех этапов.

По итогу выполнения курсовой работы получены навыки анализа предметной области, проектирования базы данных на основе реализованных схем, разработки модуля графического интерфейса и модуля для работы с базой данных, используя библиотеки PyQT5 и MySQL, а также работы с IDE PyCharm в процессе разработки функционала приложения. Все разработанные модули работают корректно, что можно считать успешным результатом проведенной работы.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Изучаем Python, 3‑е издание / Лутц М; [пер. с англ. А. Киселева]. — СПб.: Символ‑Плюс, 2009. — 848 с., ил.
2. Технология разработки программных продуктов. Практикум: учеб. пособие для студ. учреждений сред. проф. образования / А. В.Рудаков, Г. Н. Федорова. — 4-е изд., стер. — М. : Издательский центр «Академия»; 2014. — 192 с. ISBN 978-5-4468-0465-8
3. Основы проектирования баз данных / Г. Н. Федорова. - Москва : Академия, 2017. - 218, [1] с. : ил.
4. Python 3 и PyQt 5. Разработка приложений. — 2-е изд., перераб. и доп. / Н. А. Прохоренок, В. А. Дронов. — СПб.: БХВ-Петербург, 2018. — 832 с.: ил. — (Профессиональное программирование)
5. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. П75 Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001 — 368 с.: ил. (Серия «Библиотека программиста»)

# **ПРИЛОЖЕНИЕ А Запросы на создание таблиц и добавление в них данных**

Создание таблицы «Клиент»

CREATE TABLE `clients` (

`Код клиента` int NOT NULL AUTO\_INCREMENT,

`Фамилия` varchar(50) NOT NULL,

`Имя` varchar(50) NOT NULL,

`Отчество` varchar(50) NULL,

`Дата рождения` varchar(50) NOT NULL,

`e-mail` varchar(50) NOT NULL,

PRIMARY KEY (`Код клиента`)

);

Создание таблицы «Сотрудник»

CREATE TABLE `employees` (

`Код сотрудника` int NOT NULL AUTO\_INCREMENT,

`Фамилия` varchar(50) NOT NULL,

`Имя` varchar(100) NOT NULL,

`Отчество` varchar(100) NULL,

`Должность` varchar(50) NOT NULL,

`Логин` varchar(50) NOT NULL,

`Пароль` varchar(50) NOT NULL,

`Последний вход` varchar(50) NOT NULL,

`Тип входа` varchar(50) NOT NULL,

`Фото` varchar(255) NOT NULL,

PRIMARY KEY (`Код сотрудника`)

);

Создание таблицы «Жанр»

CREATE TABLE `genre` (

`Код жанра` int NOT NULL AUTO\_INCREMENT,

`Жанр` varchar(45) NOT NULL,

PRIMARY KEY (`Код жанра`)

);

Создание таблицы «История входа»

CREATE TABLE `history` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Дата входа` varchar(50) NULL,

`Дата выхода` varchar(50) NULL,

`Блокировка` varchar(50) NOT NULL,

`Логин сотрудника` int NOT NULL,

PRIMARY KEY (`ID`),

KEY `history\_FK` (`Логин сотрудника`),

CONSTRAINT `history\_FK` FOREIGN KEY (`Логин сотрудника`) REFERENCES `employees` (`Код сотрудника`)

);

Создание таблицы «Издательство»

CREATE TABLE `publishing\_house` (

`Код издательства` int NOT NULL AUTO\_INCREMENT,

`Название издательства` varchar(100) NOT NULL,

PRIMARY KEY (`Код издательства`)

);

Создание таблицы «Заказ»

CREATE TABLE `requests` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Код заказа` varchar(50) NOT NULL,

`Дата создания` varchar(50) NOT NULL,

`Время заказа` varchar(50) NOT NULL,

`Клиент` int NOT NULL,

`Книга` int NOT NULL,

`Сотрудник` int NOT NULL,

PRIMARY KEY (`ID`),

KEY `requests\_FK` (`Книга`),

KEY `requests\_FK\_1` (`Клиент`),

KEY `requests\_FK\_2` (`Сотрудник`),

CONSTRAINT `requests\_FK` FOREIGN KEY (`Книга`) REFERENCES `books` (`Код книги`),

CONSTRAINT `requests\_FK\_1` FOREIGN KEY (`Клиент`) REFERENCES `clients` (`Код клиента`),

CONSTRAINT `requests\_FK\_2` FOREIGN KEY (`Сотрудник`) REFERENCES `employees` (`Код сотрудника`)

);

# **ПРИЛОЖЕНИЕ Б Иллюстрации форм**

Форма просмотра истории сотрудников представлена на рисунке Б.1.

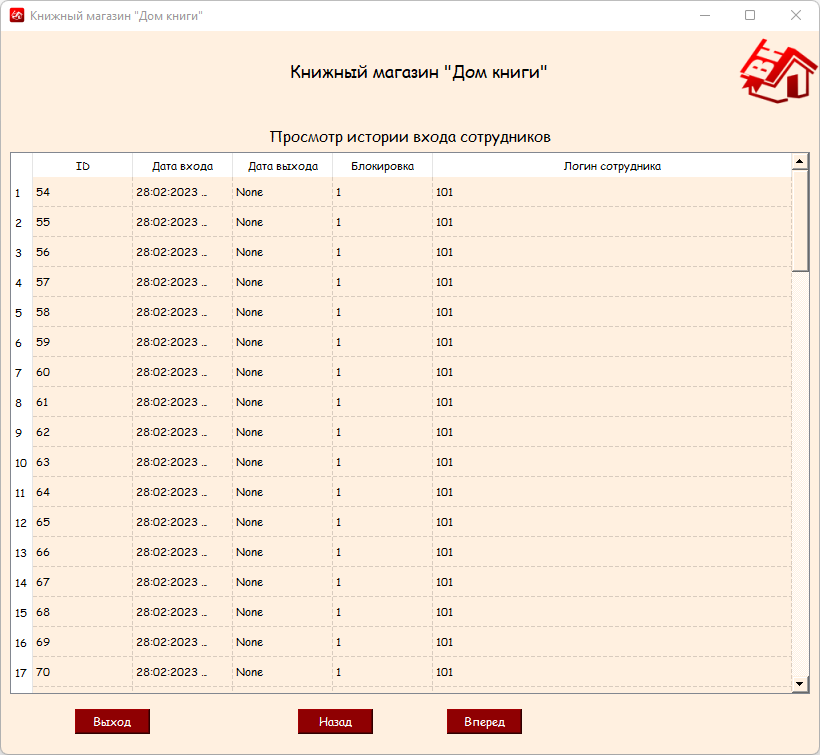


Рисунок Б.1 — Форма просмотра истории входа сотрудников

Форма просмотра книг представлена на рисунке Б.2.

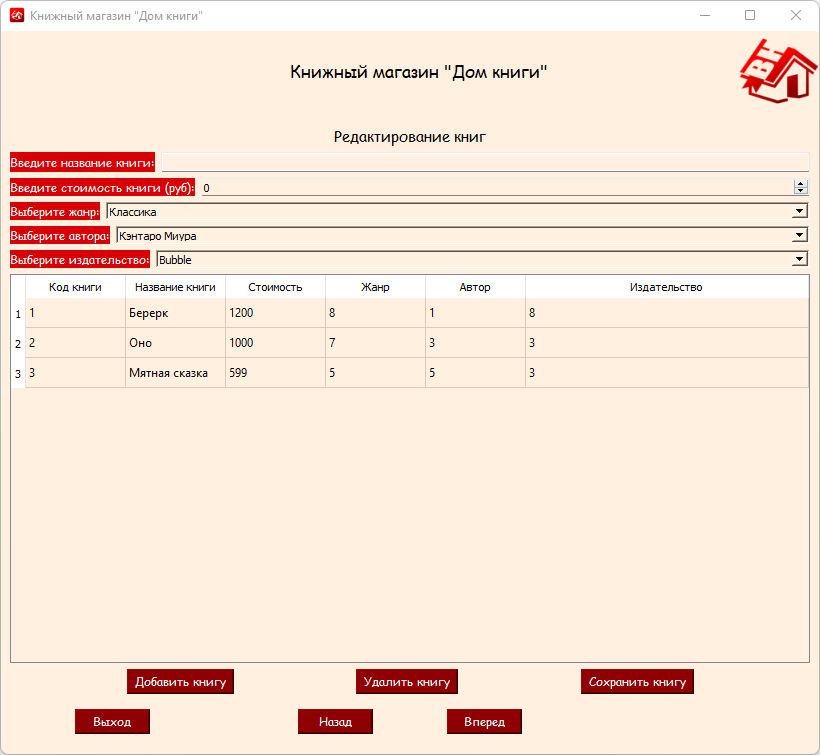


Рисунок Б.2 — Форма просмотра книг

Форма просмотра жанров представлена на рисунке Б.3.

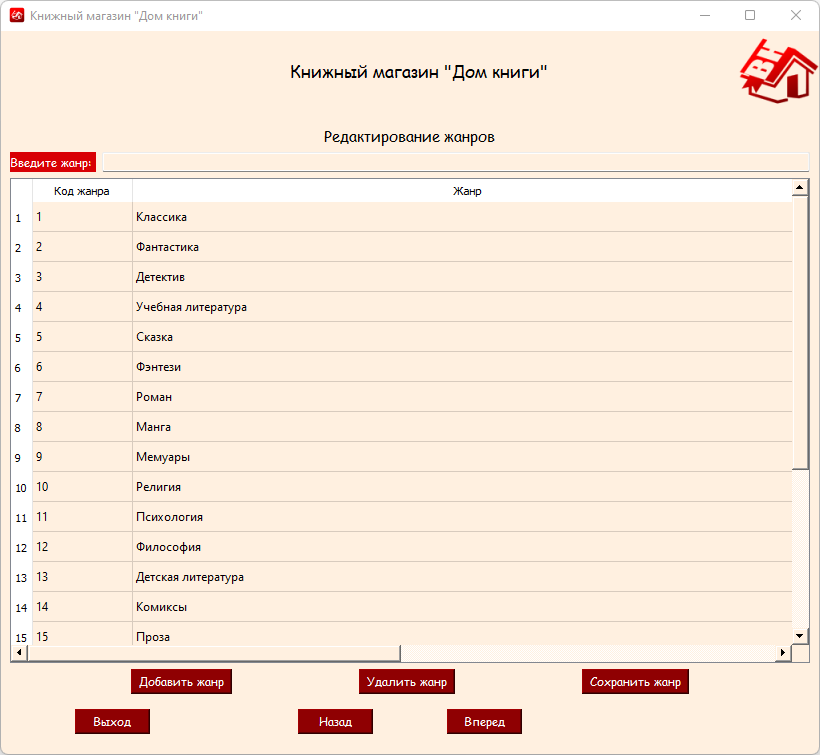


Рисунок Б.3 — Форма просмотра жанров

Форма просмотра авторов представлена на рисунке Б.4.

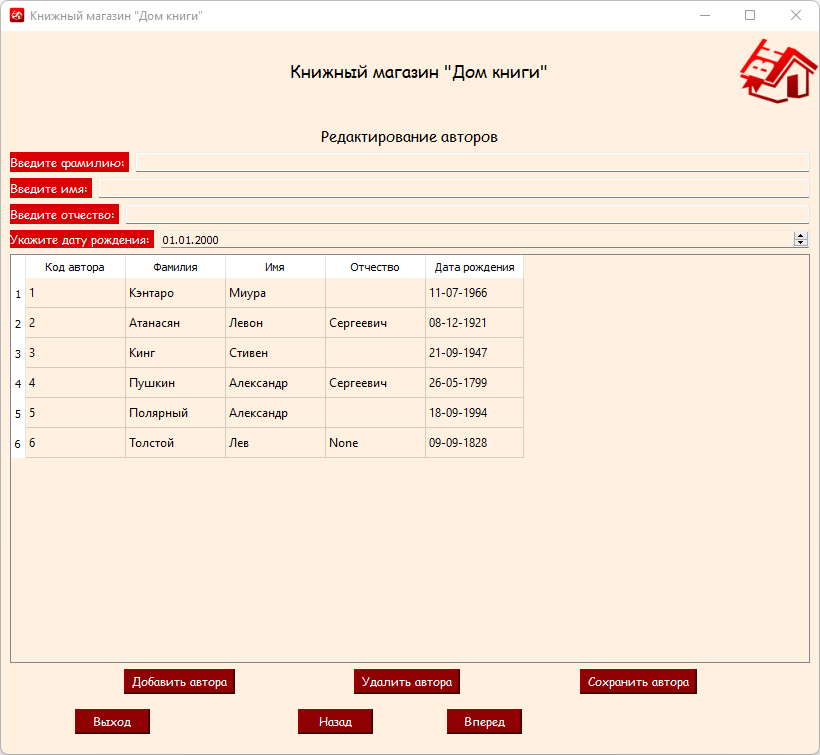


Рисунок Б.4 — Форма просмотра авторов

Форма просмотра издательств представлена на рисунке Б.5.

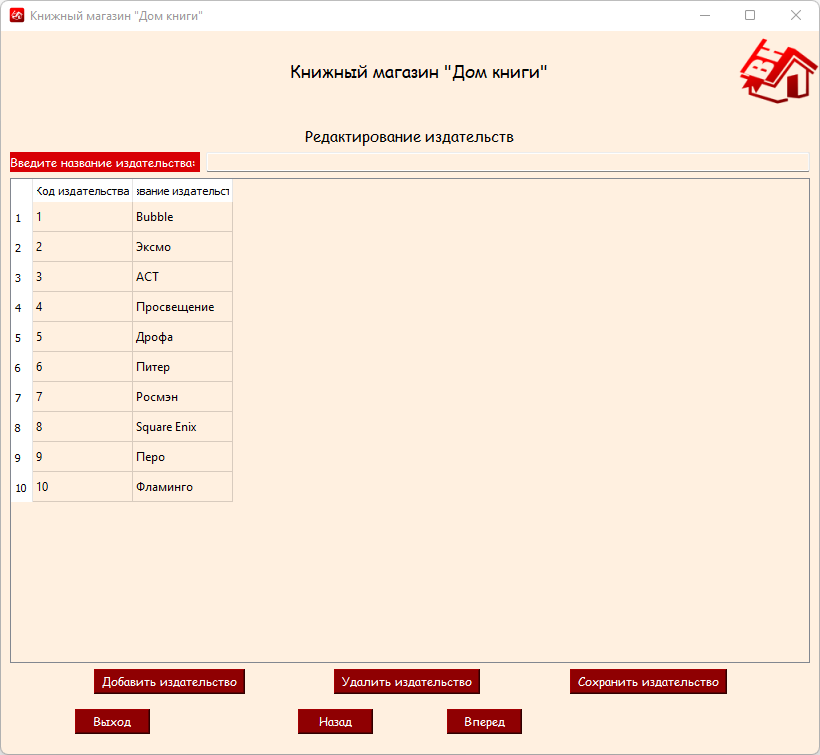


Рисунок Б.5 — Форма просмотра издательств

Форма сощдания заказа представлена на рисунке Б.6.

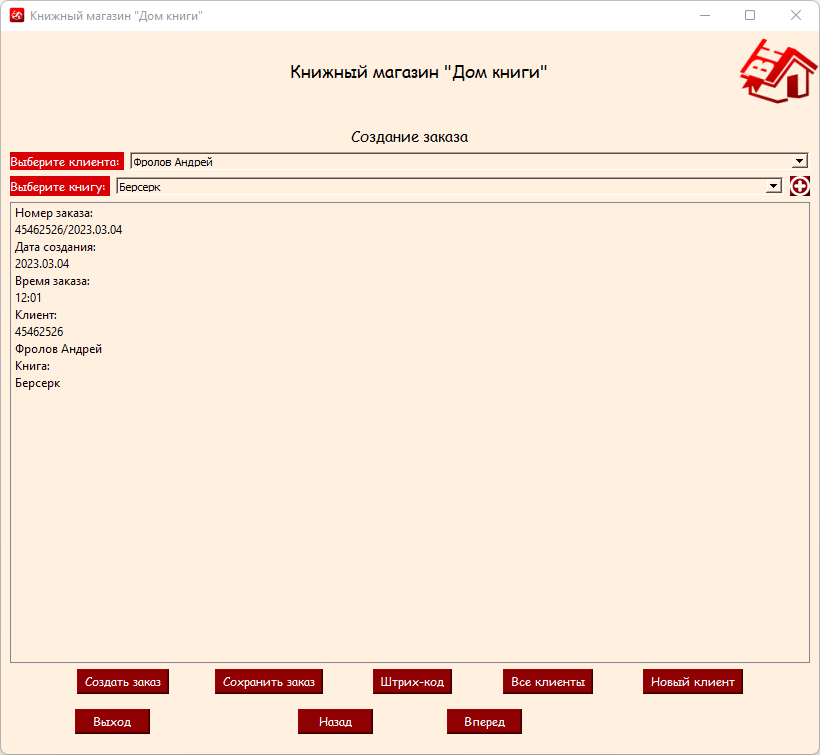


Рисунок Б.6 — Форма создания заказа

Форма создания отчётов представлена на рисунке Б.7.

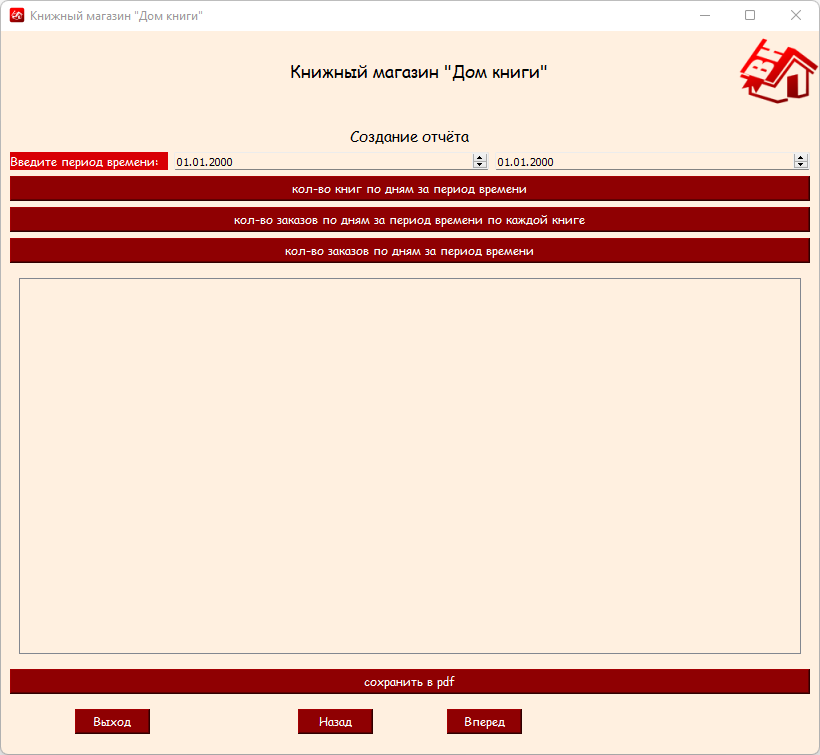


Рисунок Б.7 — Форма создания отчётов

Форма просмотра клиентов представлена на рисунке Б.8.

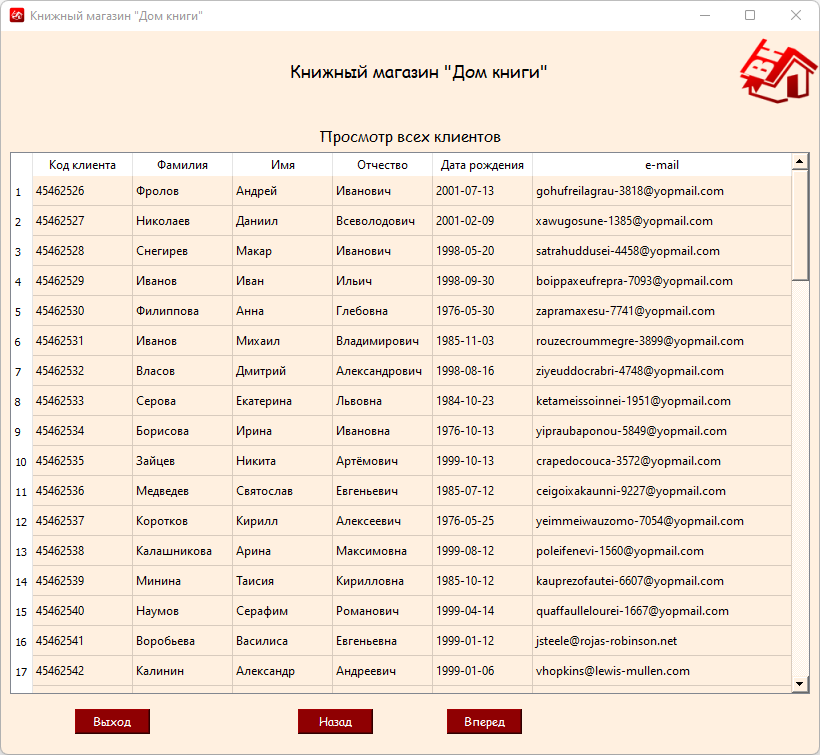


Рисунок Б.8 — Форма просмотра клиентов

Форма добавления нового клиента представлена на рисунке Б.9.

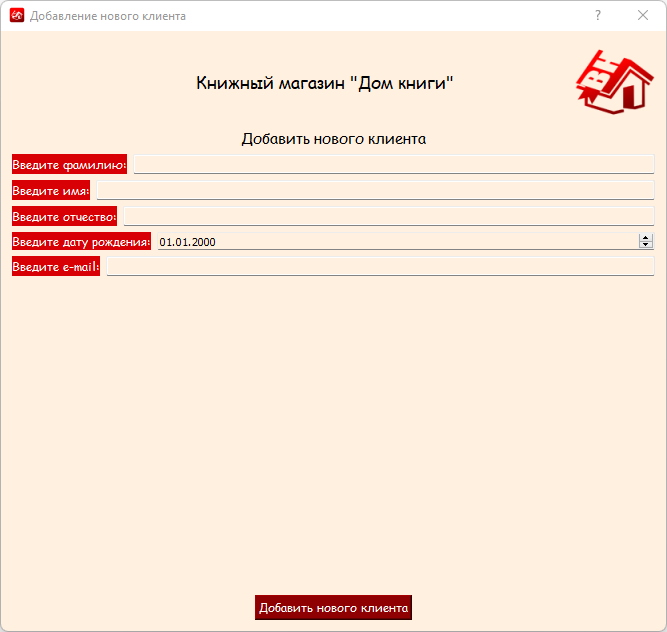


Рисунок Б.9 — Форма добавления нового клиента

# **ПРИЛОЖЕНИЕ В Исходный код форм в формате .py**

Auth.py

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_Dialog(object):

def setupUi(self, Dialog):

Dialog.setObjectName("Dialog")

Dialog.resize(450, 350)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(Dialog.sizePolicy().hasHeightForWidth())

Dialog.setSizePolicy(sizePolicy)

Dialog.setMaximumSize(QtCore.QSize(450, 350))

Dialog.setStyleSheet("background-color: rgb(255, 240, 224);")

self.verticalLayout\_2 = QtWidgets.QVBoxLayout(Dialog)

self.verticalLayout\_2.setObjectName("verticalLayout\_2")

self.verticalLayout = QtWidgets.QVBoxLayout()

self.verticalLayout.setObjectName("verticalLayout")

self.horizontalLayout\_3 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_3.setContentsMargins(12, -1, 0, -1)

self.horizontalLayout\_3.setObjectName("horizontalLayout\_3")

spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_3.addItem(spacerItem)

self.label\_5 = QtWidgets.QLabel(Dialog)

self.label\_5.setMinimumSize(QtCore.QSize(0, 0))

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(13)

self.label\_5.setFont(font)

self.label\_5.setAlignment(QtCore.Qt.AlignCenter)

self.label\_5.setObjectName("label\_5")

self.horizontalLayout\_3.addWidget(self.label\_5)

spacerItem1 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_3.addItem(spacerItem1)

self.logo = QtWidgets.QLabel(Dialog)

self.logo.setMaximumSize(QtCore.QSize(80, 80))

self.logo.setText("")

self.logo.setPixmap(QtGui.QPixmap("img/logo\_mini.PNG"))

self.logo.setScaledContents(True)

self.logo.setWordWrap(False)

self.logo.setOpenExternalLinks(False)

self.logo.setObjectName("logo")

self.horizontalLayout\_3.addWidget(self.logo)

self.horizontalLayout\_3.setStretch(0, 3)

self.horizontalLayout\_3.setStretch(1, 1)

self.horizontalLayout\_3.setStretch(2, 2)

self.horizontalLayout\_3.setStretch(3, 1)

self.verticalLayout.addLayout(self.horizontalLayout\_3)

self.label = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(11)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label.setFont(font)

self.label.setStyleSheet("font: 11pt \"Comic Sans MS\";")

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

self.verticalLayout.addWidget(self.label)

self.line = QtWidgets.QFrame(Dialog)

self.line.setFrameShape(QtWidgets.QFrame.HLine)

self.line.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line.setObjectName("line")

self.verticalLayout.addWidget(self.line)

self.horizontalLayout = QtWidgets.QHBoxLayout()

self.horizontalLayout.setObjectName("horizontalLayout")

self.label\_2 = QtWidgets.QLabel(Dialog)

self.label\_2.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_2.setObjectName("label\_2")

self.horizontalLayout.addWidget(self.label\_2)

self.edit\_login = QtWidgets.QLineEdit(Dialog)

self.edit\_login.setObjectName("edit\_login")

self.horizontalLayout.addWidget(self.edit\_login)

self.verticalLayout.addLayout(self.horizontalLayout)

self.horizontalLayout\_2 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")

self.label\_3 = QtWidgets.QLabel(Dialog)

self.label\_3.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_3.setObjectName("label\_3")

self.horizontalLayout\_2.addWidget(self.label\_3)

self.edit\_password = QtWidgets.QLineEdit(Dialog)

self.edit\_password.setInputMethodHints(QtCore.Qt.ImhHiddenText|QtCore.Qt.ImhNoAutoUppercase|QtCore.Qt.ImhNoPredictiveText|QtCore.Qt.ImhSensitiveData)

self.edit\_password.setEchoMode(QtWidgets.QLineEdit.Password)

self.edit\_password.setObjectName("edit\_password")

self.horizontalLayout\_2.addWidget(self.edit\_password)

self.btn\_hide\_password = QtWidgets.QPushButton(Dialog)

self.btn\_hide\_password.setMaximumSize(QtCore.QSize(25, 25))

self.btn\_hide\_password.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))

self.btn\_hide\_password.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_hide\_password.setText("")

icon = QtGui.QIcon()

icon.addPixmap(QtGui.QPixmap("img/eye.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.btn\_hide\_password.setIcon(icon)

self.btn\_hide\_password.setObjectName("btn\_hide\_password")

self.horizontalLayout\_2.addWidget(self.btn\_hide\_password)

self.verticalLayout.addLayout(self.horizontalLayout\_2)

self.horizontalLayout\_4 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_4.setObjectName("horizontalLayout\_4")

self.label\_4 = QtWidgets.QLabel(Dialog)

self.label\_4.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_4.setObjectName("label\_4")

self.horizontalLayout\_4.addWidget(self.label\_4)

self.edit\_captcha = QtWidgets.QLineEdit(Dialog)

self.edit\_captcha.setObjectName("edit\_captcha")

self.horizontalLayout\_4.addWidget(self.edit\_captcha)

self.btn\_new\_captcha = QtWidgets.QPushButton(Dialog)

self.btn\_new\_captcha.setMaximumSize(QtCore.QSize(25, 25))

self.btn\_new\_captcha.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))

self.btn\_new\_captcha.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_new\_captcha.setText("")

icon1 = QtGui.QIcon()

icon1.addPixmap(QtGui.QPixmap("img/new.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.btn\_new\_captcha.setIcon(icon1)

self.btn\_new\_captcha.setObjectName("btn\_new\_captcha")

self.horizontalLayout\_4.addWidget(self.btn\_new\_captcha)

self.verticalLayout.addLayout(self.horizontalLayout\_4)

self.draw\_captcha = QtWidgets.QGraphicsView(Dialog)

self.draw\_captcha.setObjectName("draw\_captcha")

self.verticalLayout.addWidget(self.draw\_captcha)

spacerItem2 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout.addItem(spacerItem2)

self.btn\_enter = QtWidgets.QPushButton(Dialog)

self.btn\_enter.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))

self.btn\_enter.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_enter.setObjectName("btn\_enter")

self.verticalLayout.addWidget(self.btn\_enter)

self.verticalLayout.setStretch(1, 1)

self.verticalLayout.setStretch(3, 1)

self.verticalLayout.setStretch(4, 1)

self.verticalLayout.setStretch(7, 10)

self.verticalLayout.setStretch(8, 1)

self.verticalLayout\_2.addLayout(self.verticalLayout)

self.retranslateUi(Dialog)

QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self, Dialog):

\_translate = QtCore.QCoreApplication.translate

Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))

self.label\_5.setText(\_translate("Dialog", "Книжный магазин \"Дом книги\""))

self.label.setText(\_translate("Dialog", "Авторизация"))

self.label\_2.setText(\_translate("Dialog", " Логин: "))

self.label\_3.setText(\_translate("Dialog", " Пароль: "))

self.label\_4.setText(\_translate("Dialog", " Введите капчу: "))

self.btn\_enter.setText(\_translate("Dialog", "Войти"))

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

import sys

app = QtWidgets.QApplication(sys.argv)

Dialog = QtWidgets.QDialog()

ui = Ui\_Dialog()

ui.setupUi(Dialog)

Dialog.show()

sys.exit(app.exec\_())

Main.py

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_BookShop(object):

def setupUi(self, BookShop):

BookShop.setObjectName("BookShop")

BookShop.resize(818, 723)

BookShop.setMinimumSize(QtCore.QSize(600, 550))

BookShop.setMaximumSize(QtCore.QSize(16777215, 16777215))

icon = QtGui.QIcon()

icon.addPixmap(QtGui.QPixmap("img/logo.PNG"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

BookShop.setWindowIcon(icon)

BookShop.setAutoFillBackground(False)

BookShop.setStyleSheet("background-color: rgb(255, 240, 224);")

self.centralwidget = QtWidgets.QWidget(BookShop)

self.centralwidget.setStyleSheet("")

self.centralwidget.setObjectName("centralwidget")

self.verticalLayout\_2 = QtWidgets.QVBoxLayout(self.centralwidget)

self.verticalLayout\_2.setObjectName("verticalLayout\_2")

self.horizontalLayout\_2 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_2.setContentsMargins(12, -1, 0, -1)

self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")

spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_2.addItem(spacerItem)

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setMinimumSize(QtCore.QSize(0, 0))

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(13)

self.label.setFont(font)

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

self.horizontalLayout\_2.addWidget(self.label)

spacerItem1 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_2.addItem(spacerItem1)

self.logo = QtWidgets.QLabel(self.centralwidget)

self.logo.setMaximumSize(QtCore.QSize(80, 80))

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(8)

self.logo.setFont(font)

self.logo.setText("")

self.logo.setTextFormat(QtCore.Qt.AutoText)

self.logo.setPixmap(QtGui.QPixmap("img/logo\_mini.PNG"))

self.logo.setScaledContents(True)

self.logo.setWordWrap(False)

self.logo.setOpenExternalLinks(False)

self.logo.setObjectName("logo")

self.horizontalLayout\_2.addWidget(self.logo)

self.horizontalLayout\_2.setStretch(0, 3)

self.horizontalLayout\_2.setStretch(1, 1)

self.horizontalLayout\_2.setStretch(2, 2)

self.horizontalLayout\_2.setStretch(3, 1)

self.verticalLayout\_2.addLayout(self.horizontalLayout\_2)

self.verticalLayout = QtWidgets.QVBoxLayout()

self.verticalLayout.setObjectName("verticalLayout")

self.stackedWidget\_main = QtWidgets.QStackedWidget(self.centralwidget)

self.stackedWidget\_main.setStyleSheet("font: \"Comic Sans MS\";")

self.stackedWidget\_main.setObjectName("stackedWidget\_main")

self.page\_1\_for\_all = QtWidgets.QWidget()

self.page\_1\_for\_all.setObjectName("page\_1\_for\_all")

self.verticalLayout\_3 = QtWidgets.QVBoxLayout(self.page\_1\_for\_all)

self.verticalLayout\_3.setObjectName("verticalLayout\_3")

self.verticalLayout\_4 = QtWidgets.QVBoxLayout()

self.verticalLayout\_4.setSizeConstraint(QtWidgets.QLayout.SetMaximumSize)

self.verticalLayout\_4.setSpacing(2)

self.verticalLayout\_4.setObjectName("verticalLayout\_4")

self.horizontalLayout\_6 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_6.setObjectName("horizontalLayout\_6")

self.label\_3 = QtWidgets.QLabel(self.page\_1\_for\_all)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_3.sizePolicy().hasHeightForWidth())

self.label\_3.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(11)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_3.setFont(font)

self.label\_3.setAlignment(QtCore.Qt.AlignCenter)

self.label\_3.setObjectName("label\_3")

self.horizontalLayout\_6.addWidget(self.label\_3)

self.verticalLayout\_4.addLayout(self.horizontalLayout\_6)

self.horizontalLayout\_4 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_4.setSizeConstraint(QtWidgets.QLayout.SetDefaultConstraint)

self.horizontalLayout\_4.setSpacing(0)

self.horizontalLayout\_4.setObjectName("horizontalLayout\_4")

self.label\_4 = QtWidgets.QLabel(self.page\_1\_for\_all)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_4.sizePolicy().hasHeightForWidth())

self.label\_4.setSizePolicy(sizePolicy)

self.label\_4.setMaximumSize(QtCore.QSize(35, 30))

self.label\_4.setBaseSize(QtCore.QSize(0, 30))

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(9)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_4.setFont(font)

self.label\_4.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_4.setObjectName("label\_4")

self.horizontalLayout\_4.addWidget(self.label\_4)

self.lbl\_fio = QtWidgets.QLabel(self.page\_1\_for\_all)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.lbl\_fio.sizePolicy().hasHeightForWidth())

self.lbl\_fio.setSizePolicy(sizePolicy)

self.lbl\_fio.setMinimumSize(QtCore.QSize(0, 0))

self.lbl\_fio.setMaximumSize(QtCore.QSize(16777213, 30))

self.lbl\_fio.setBaseSize(QtCore.QSize(0, 30))

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(9)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.lbl\_fio.setFont(font)

self.lbl\_fio.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.lbl\_fio.setObjectName("lbl\_fio")

self.horizontalLayout\_4.addWidget(self.lbl\_fio)

self.verticalLayout\_4.addLayout(self.horizontalLayout\_4)

self.horizontalLayout\_3 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_3.setSizeConstraint(QtWidgets.QLayout.SetMinimumSize)

self.horizontalLayout\_3.setSpacing(0)

self.horizontalLayout\_3.setObjectName("horizontalLayout\_3")

self.label\_2 = QtWidgets.QLabel(self.page\_1\_for\_all)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_2.sizePolicy().hasHeightForWidth())

self.label\_2.setSizePolicy(sizePolicy)

self.label\_2.setMaximumSize(QtCore.QSize(35, 30))

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(9)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_2.setFont(font)

self.label\_2.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_2.setObjectName("label\_2")

self.horizontalLayout\_3.addWidget(self.label\_2)

self.lbl\_role = QtWidgets.QLabel(self.page\_1\_for\_all)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.lbl\_role.sizePolicy().hasHeightForWidth())

self.lbl\_role.setSizePolicy(sizePolicy)

self.lbl\_role.setMinimumSize(QtCore.QSize(0, 0))

self.lbl\_role.setMaximumSize(QtCore.QSize(16777215, 30))

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(9)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.lbl\_role.setFont(font)

self.lbl\_role.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.lbl\_role.setObjectName("lbl\_role")

self.horizontalLayout\_3.addWidget(self.lbl\_role)

self.verticalLayout\_4.addLayout(self.horizontalLayout\_3)

self.horizontalLayout\_7 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_7.setObjectName("horizontalLayout\_7")

spacerItem2 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_7.addItem(spacerItem2)

self.lbl\_photo = QtWidgets.QLabel(self.page\_1\_for\_all)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.lbl\_photo.sizePolicy().hasHeightForWidth())

self.lbl\_photo.setSizePolicy(sizePolicy)

self.lbl\_photo.setText("")

self.lbl\_photo.setTextFormat(QtCore.Qt.AutoText)

self.lbl\_photo.setPixmap(QtGui.QPixmap("img/Миронов.jpeg"))

self.lbl\_photo.setScaledContents(True)

self.lbl\_photo.setAlignment(QtCore.Qt.AlignJustify|QtCore.Qt.AlignVCenter)

self.lbl\_photo.setWordWrap(False)

self.lbl\_photo.setObjectName("lbl\_photo")

self.horizontalLayout\_7.addWidget(self.lbl\_photo)

spacerItem3 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_7.addItem(spacerItem3)

self.horizontalLayout\_7.setStretch(0, 1)

self.horizontalLayout\_7.setStretch(1, 2)

self.horizontalLayout\_7.setStretch(2, 1)

self.verticalLayout\_4.addLayout(self.horizontalLayout\_7)

self.verticalLayout\_4.setStretch(0, 1)

self.verticalLayout\_4.setStretch(1, 1)

self.verticalLayout\_4.setStretch(2, 1)

self.verticalLayout\_4.setStretch(3, 10)

self.verticalLayout\_3.addLayout(self.verticalLayout\_4)

self.stackedWidget\_main.addWidget(self.page\_1\_for\_all)

self.page\_2\_for\_admin = QtWidgets.QWidget()

self.page\_2\_for\_admin.setStyleSheet("font: \"Comic Sans MS\";")

self.page\_2\_for\_admin.setObjectName("page\_2\_for\_admin")

self.verticalLayout\_7 = QtWidgets.QVBoxLayout(self.page\_2\_for\_admin)

self.verticalLayout\_7.setObjectName("verticalLayout\_7")

self.horizontalLayout\_9 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_9.setObjectName("horizontalLayout\_9")

self.label\_5 = QtWidgets.QLabel(self.page\_2\_for\_admin)

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(11)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_5.setFont(font)

self.label\_5.setStyleSheet("font: 11pt \"Comic Sans MS\";")

self.label\_5.setAlignment(QtCore.Qt.AlignCenter)

self.label\_5.setObjectName("label\_5")

self.horizontalLayout\_9.addWidget(self.label\_5)

self.verticalLayout\_7.addLayout(self.horizontalLayout\_9)

self.table\_entry = QtWidgets.QTableWidget(self.page\_2\_for\_admin)

self.table\_entry.setStatusTip("")

self.table\_entry.setStyleSheet("font: 8pt \"Comic Sans MS\";")

self.table\_entry.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.table\_entry.setAutoScroll(False)

self.table\_entry.setAutoScrollMargin(20)

self.table\_entry.setAlternatingRowColors(False)

self.table\_entry.setHorizontalScrollMode(QtWidgets.QAbstractItemView.ScrollPerItem)

self.table\_entry.setShowGrid(True)

self.table\_entry.setGridStyle(QtCore.Qt.DashLine)

self.table\_entry.setWordWrap(True)

self.table\_entry.setCornerButtonEnabled(True)

self.table\_entry.setObjectName("table\_entry")

self.table\_entry.setColumnCount(0)

self.table\_entry.setRowCount(0)

self.table\_entry.horizontalHeader().setDefaultSectionSize(100)

self.table\_entry.horizontalHeader().setStretchLastSection(True)

self.table\_entry.verticalHeader().setStretchLastSection(False)

self.verticalLayout\_7.addWidget(self.table\_entry)

self.stackedWidget\_main.addWidget(self.page\_2\_for\_admin)

self.page\_3\_not\_for\_admin = QtWidgets.QWidget()

self.page\_3\_not\_for\_admin.setObjectName("page\_3\_not\_for\_admin")

self.verticalLayout\_8 = QtWidgets.QVBoxLayout(self.page\_3\_not\_for\_admin)

self.verticalLayout\_8.setObjectName("verticalLayout\_8")

self.verticalLayout\_6 = QtWidgets.QVBoxLayout()

self.verticalLayout\_6.setObjectName("verticalLayout\_6")

self.label\_6 = QtWidgets.QLabel(self.page\_3\_not\_for\_admin)

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(11)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_6.setFont(font)

self.label\_6.setStyleSheet("font: 11pt \"Comic Sans MS\";")

self.label\_6.setAlignment(QtCore.Qt.AlignCenter)

self.label\_6.setObjectName("label\_6")

self.verticalLayout\_6.addWidget(self.label\_6)

self.horizontalLayout\_11 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_11.setObjectName("horizontalLayout\_11")

self.label\_8 = QtWidgets.QLabel(self.page\_3\_not\_for\_admin)

self.label\_8.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_8.setObjectName("label\_8")

self.horizontalLayout\_11.addWidget(self.label\_8)

self.comboBox\_clients = QtWidgets.QComboBox(self.page\_3\_not\_for\_admin)

self.comboBox\_clients.setObjectName("comboBox\_clients")

self.horizontalLayout\_11.addWidget(self.comboBox\_clients)

self.horizontalLayout\_11.setStretch(0, 1)

self.horizontalLayout\_11.setStretch(1, 10)

self.verticalLayout\_6.addLayout(self.horizontalLayout\_11)

self.horizontalLayout\_12 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_12.setObjectName("horizontalLayout\_12")

self.label\_9 = QtWidgets.QLabel(self.page\_3\_not\_for\_admin)

self.label\_9.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_9.setObjectName("label\_9")

self.horizontalLayout\_12.addWidget(self.label\_9)

self.comboBox\_book = QtWidgets.QComboBox(self.page\_3\_not\_for\_admin)

self.comboBox\_book.setObjectName("comboBox\_book")

self.horizontalLayout\_12.addWidget(self.comboBox\_book)

self.btn\_plus = QtWidgets.QPushButton(self.page\_3\_not\_for\_admin)

self.btn\_plus.setMaximumSize(QtCore.QSize(20, 20))

self.btn\_plus.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_plus.setText("")

icon1 = QtGui.QIcon()

icon1.addPixmap(QtGui.QPixmap("img/Добавить.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.btn\_plus.setIcon(icon1)

self.btn\_plus.setIconSize(QtCore.QSize(20, 20))

self.btn\_plus.setCheckable(False)

self.btn\_plus.setChecked(False)

self.btn\_plus.setAutoRepeat(False)

self.btn\_plus.setAutoExclusive(False)

self.btn\_plus.setObjectName("btn\_plus")

self.horizontalLayout\_12.addWidget(self.btn\_plus)

self.horizontalLayout\_12.setStretch(0, 1)

self.horizontalLayout\_12.setStretch(1, 100)

self.verticalLayout\_6.addLayout(self.horizontalLayout\_12)

self.add\_new\_field = QtWidgets.QListWidget(self.page\_3\_not\_for\_admin)

self.add\_new\_field.setIconSize(QtCore.QSize(500, 151))

self.add\_new\_field.setSelectionRectVisible(True)

self.add\_new\_field.setObjectName("add\_new\_field")

self.verticalLayout\_6.addWidget(self.add\_new\_field)

self.horizontalLayout\_13 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_13.setObjectName("horizontalLayout\_13")

spacerItem4 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_13.addItem(spacerItem4)

self.btn\_new\_order = QtWidgets.QPushButton(self.page\_3\_not\_for\_admin)

self.btn\_new\_order.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_new\_order.setObjectName("btn\_new\_order")

self.horizontalLayout\_13.addWidget(self.btn\_new\_order)

spacerItem5 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_13.addItem(spacerItem5)

self.btn\_save\_request = QtWidgets.QPushButton(self.page\_3\_not\_for\_admin)

self.btn\_save\_request.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_save\_request.setObjectName("btn\_save\_request")

self.horizontalLayout\_13.addWidget(self.btn\_save\_request)

spacerItem6 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_13.addItem(spacerItem6)

self.btn\_code = QtWidgets.QPushButton(self.page\_3\_not\_for\_admin)

self.btn\_code.setEnabled(True)

self.btn\_code.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_code.setObjectName("btn\_code")

self.horizontalLayout\_13.addWidget(self.btn\_code)

spacerItem7 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_13.addItem(spacerItem7)

self.btn\_all\_clients = QtWidgets.QPushButton(self.page\_3\_not\_for\_admin)

self.btn\_all\_clients.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_all\_clients.setObjectName("btn\_all\_clients")

self.horizontalLayout\_13.addWidget(self.btn\_all\_clients)

spacerItem8 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_13.addItem(spacerItem8)

self.btn\_new\_client = QtWidgets.QPushButton(self.page\_3\_not\_for\_admin)

self.btn\_new\_client.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_new\_client.setObjectName("btn\_new\_client")

self.horizontalLayout\_13.addWidget(self.btn\_new\_client)

spacerItem9 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_13.addItem(spacerItem9)

self.horizontalLayout\_13.setStretch(0, 3)

self.horizontalLayout\_13.setStretch(1, 1)

self.horizontalLayout\_13.setStretch(4, 2)

self.horizontalLayout\_13.setStretch(6, 2)

self.horizontalLayout\_13.setStretch(7, 1)

self.horizontalLayout\_13.setStretch(8, 2)

self.horizontalLayout\_13.setStretch(9, 1)

self.horizontalLayout\_13.setStretch(10, 3)

self.verticalLayout\_6.addLayout(self.horizontalLayout\_13)

self.verticalLayout\_8.addLayout(self.verticalLayout\_6)

self.stackedWidget\_main.addWidget(self.page\_3\_not\_for\_admin)

self.page\_5\_for\_page3 = QtWidgets.QWidget()

self.page\_5\_for\_page3.setObjectName("page\_5\_for\_page3")

self.verticalLayout\_10 = QtWidgets.QVBoxLayout(self.page\_5\_for\_page3)

self.verticalLayout\_10.setObjectName("verticalLayout\_10")

self.label\_11 = QtWidgets.QLabel(self.page\_5\_for\_page3)

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(11)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_11.setFont(font)

self.label\_11.setStyleSheet("font: 11pt \"Comic Sans MS\";")

self.label\_11.setAlignment(QtCore.Qt.AlignCenter)

self.label\_11.setObjectName("label\_11")

self.verticalLayout\_10.addWidget(self.label\_11)

self.table\_clients = QtWidgets.QTableWidget(self.page\_5\_for\_page3)

self.table\_clients.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.table\_clients.setDragDropOverwriteMode(True)

self.table\_clients.setGridStyle(QtCore.Qt.SolidLine)

self.table\_clients.setObjectName("table\_clients")

self.table\_clients.setColumnCount(7)

self.table\_clients.setRowCount(0)

item = QtWidgets.QTableWidgetItem()

self.table\_clients.setHorizontalHeaderItem(0, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients.setHorizontalHeaderItem(1, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients.setHorizontalHeaderItem(2, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients.setHorizontalHeaderItem(3, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients.setHorizontalHeaderItem(4, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients.setHorizontalHeaderItem(5, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients.setHorizontalHeaderItem(6, item)

self.table\_clients.horizontalHeader().setStretchLastSection(True)

self.verticalLayout\_10.addWidget(self.table\_clients)

self.stackedWidget\_main.addWidget(self.page\_5\_for\_page3)

self.page\_4\_for\_admin = QtWidgets.QWidget()

self.page\_4\_for\_admin.setObjectName("page\_4\_for\_admin")

self.verticalLayout\_9 = QtWidgets.QVBoxLayout(self.page\_4\_for\_admin)

self.verticalLayout\_9.setObjectName("verticalLayout\_9")

self.label\_10 = QtWidgets.QLabel(self.page\_4\_for\_admin)

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(11)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_10.setFont(font)

self.label\_10.setStyleSheet("font: 11pt \"Comic Sans MS\";")

self.label\_10.setAlignment(QtCore.Qt.AlignCenter)

self.label\_10.setObjectName("label\_10")

self.verticalLayout\_9.addWidget(self.label\_10)

self.horizontalLayout\_15 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_15.setObjectName("horizontalLayout\_15")

self.label\_12 = QtWidgets.QLabel(self.page\_4\_for\_admin)

self.label\_12.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_12.setObjectName("label\_12")

self.horizontalLayout\_15.addWidget(self.label\_12)

self.date\_start = QtWidgets.QDateTimeEdit(self.page\_4\_for\_admin)

self.date\_start.setMaximumDate(QtCore.QDate(9999, 12, 30))

self.date\_start.setObjectName("date\_start")

self.horizontalLayout\_15.addWidget(self.date\_start)

self.date\_end = QtWidgets.QDateTimeEdit(self.page\_4\_for\_admin)

self.date\_end.setObjectName("date\_end")

self.horizontalLayout\_15.addWidget(self.date\_end)

self.horizontalLayout\_15.setStretch(0, 1)

self.horizontalLayout\_15.setStretch(1, 2)

self.horizontalLayout\_15.setStretch(2, 2)

self.verticalLayout\_9.addLayout(self.horizontalLayout\_15)

self.btn\_count\_book = QtWidgets.QPushButton(self.page\_4\_for\_admin)

self.btn\_count\_book.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_count\_book.setObjectName("btn\_count\_book")

self.verticalLayout\_9.addWidget(self.btn\_count\_book)

self.btn\_count\_order\_book = QtWidgets.QPushButton(self.page\_4\_for\_admin)

self.btn\_count\_order\_book.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_count\_order\_book.setObjectName("btn\_count\_order\_book")

self.verticalLayout\_9.addWidget(self.btn\_count\_order\_book)

self.btn\_count\_order = QtWidgets.QPushButton(self.page\_4\_for\_admin)

self.btn\_count\_order.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_count\_order.setObjectName("btn\_count\_order")

self.verticalLayout\_9.addWidget(self.btn\_count\_order)

self.stackedWidget\_report = QtWidgets.QStackedWidget(self.page\_4\_for\_admin)

self.stackedWidget\_report.setObjectName("stackedWidget\_report")

self.page\_1\_table = QtWidgets.QWidget()

self.page\_1\_table.setObjectName("page\_1\_table")

self.horizontalLayout\_16 = QtWidgets.QHBoxLayout(self.page\_1\_table)

self.horizontalLayout\_16.setObjectName("horizontalLayout\_16")

self.verticalLayout\_11 = QtWidgets.QVBoxLayout()

self.verticalLayout\_11.setObjectName("verticalLayout\_11")

self.table\_graf = QtWidgets.QTableWidget(self.page\_1\_table)

self.table\_graf.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.table\_graf.setObjectName("table\_graf")

self.table\_graf.setColumnCount(0)

self.table\_graf.setRowCount(0)

self.table\_graf.horizontalHeader().setStretchLastSection(True)

self.verticalLayout\_11.addWidget(self.table\_graf)

self.horizontalLayout\_16.addLayout(self.verticalLayout\_11)

self.stackedWidget\_report.addWidget(self.page\_1\_table)

self.page\_2\_graf = QtWidgets.QWidget()

self.page\_2\_graf.setObjectName("page\_2\_graf")

self.verticalLayout\_13 = QtWidgets.QVBoxLayout(self.page\_2\_graf)

self.verticalLayout\_13.setObjectName("verticalLayout\_13")

self.verticalLayout\_12 = QtWidgets.QVBoxLayout()

self.verticalLayout\_12.setObjectName("verticalLayout\_12")

self.label\_13 = QtWidgets.QLabel(self.page\_2\_graf)

self.label\_13.setObjectName("label\_13")

self.verticalLayout\_12.addWidget(self.label\_13)

self.verticalLayout\_13.addLayout(self.verticalLayout\_12)

self.stackedWidget\_report.addWidget(self.page\_2\_graf)

self.verticalLayout\_9.addWidget(self.stackedWidget\_report)

self.horizontalLayout\_14 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_14.setObjectName("horizontalLayout\_14")

self.btn\_pdf = QtWidgets.QPushButton(self.page\_4\_for\_admin)

self.btn\_pdf.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_pdf.setObjectName("btn\_pdf")

self.horizontalLayout\_14.addWidget(self.btn\_pdf)

self.horizontalLayout\_14.setStretch(0, 1)

self.verticalLayout\_9.addLayout(self.horizontalLayout\_14)

self.stackedWidget\_main.addWidget(self.page\_4\_for\_admin)

self.page\_6\_for\_admin = QtWidgets.QWidget()

self.page\_6\_for\_admin.setObjectName("page\_6\_for\_admin")

self.verticalLayout\_14 = QtWidgets.QVBoxLayout(self.page\_6\_for\_admin)

self.verticalLayout\_14.setObjectName("verticalLayout\_14")

self.horizontalLayout\_17 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_17.setObjectName("horizontalLayout\_17")

self.label\_14 = QtWidgets.QLabel(self.page\_6\_for\_admin)

self.label\_14.setStyleSheet("font: 11pt \"Comic Sans MS\";")

self.label\_14.setAlignment(QtCore.Qt.AlignCenter)

self.label\_14.setObjectName("label\_14")

self.horizontalLayout\_17.addWidget(self.label\_14)

self.verticalLayout\_14.addLayout(self.horizontalLayout\_17)

self.verticalLayout\_5 = QtWidgets.QVBoxLayout()

self.verticalLayout\_5.setObjectName("verticalLayout\_5")

self.horizontalLayout\_18 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_18.setObjectName("horizontalLayout\_18")

self.label\_15 = QtWidgets.QLabel(self.page\_6\_for\_admin)

self.label\_15.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_15.setObjectName("label\_15")

self.horizontalLayout\_18.addWidget(self.label\_15)

self.edit\_title\_book = QtWidgets.QLineEdit(self.page\_6\_for\_admin)

self.edit\_title\_book.setObjectName("edit\_title\_book")

self.horizontalLayout\_18.addWidget(self.edit\_title\_book)

self.horizontalLayout\_18.setStretch(1, 100)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_18)

self.horizontalLayout\_19 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_19.setObjectName("horizontalLayout\_19")

self.label\_18 = QtWidgets.QLabel(self.page\_6\_for\_admin)

self.label\_18.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_18.setObjectName("label\_18")

self.horizontalLayout\_19.addWidget(self.label\_18)

self.spin\_cost = QtWidgets.QSpinBox(self.page\_6\_for\_admin)

self.spin\_cost.setMaximum(100000)

self.spin\_cost.setObjectName("spin\_cost")

self.horizontalLayout\_19.addWidget(self.spin\_cost)

self.horizontalLayout\_19.setStretch(1, 100)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_19)

self.horizontalLayout\_24 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_24.setObjectName("horizontalLayout\_24")

self.label\_19 = QtWidgets.QLabel(self.page\_6\_for\_admin)

self.label\_19.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_19.setObjectName("label\_19")

self.horizontalLayout\_24.addWidget(self.label\_19)

self.comboBox\_genre = QtWidgets.QComboBox(self.page\_6\_for\_admin)

self.comboBox\_genre.setObjectName("comboBox\_genre")

self.horizontalLayout\_24.addWidget(self.comboBox\_genre)

self.horizontalLayout\_24.setStretch(1, 100)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_24)

self.horizontalLayout\_23 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_23.setObjectName("horizontalLayout\_23")

self.label\_17 = QtWidgets.QLabel(self.page\_6\_for\_admin)

self.label\_17.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_17.setObjectName("label\_17")

self.horizontalLayout\_23.addWidget(self.label\_17)

self.comboBox\_author = QtWidgets.QComboBox(self.page\_6\_for\_admin)

self.comboBox\_author.setObjectName("comboBox\_author")

self.horizontalLayout\_23.addWidget(self.comboBox\_author)

self.horizontalLayout\_23.setStretch(1, 100)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_23)

self.horizontalLayout\_22 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_22.setObjectName("horizontalLayout\_22")

self.label\_16 = QtWidgets.QLabel(self.page\_6\_for\_admin)

self.label\_16.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_16.setObjectName("label\_16")

self.horizontalLayout\_22.addWidget(self.label\_16)

self.comboBox\_ph = QtWidgets.QComboBox(self.page\_6\_for\_admin)

self.comboBox\_ph.setObjectName("comboBox\_ph")

self.horizontalLayout\_22.addWidget(self.comboBox\_ph)

self.horizontalLayout\_22.setStretch(1, 100)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_22)

self.verticalLayout\_14.addLayout(self.verticalLayout\_5)

self.table\_book = QtWidgets.QTableWidget(self.page\_6\_for\_admin)

self.table\_book.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.table\_book.setObjectName("table\_book")

self.table\_book.setColumnCount(0)

self.table\_book.setRowCount(0)

self.table\_book.horizontalHeader().setStretchLastSection(True)

self.verticalLayout\_14.addWidget(self.table\_book)

self.horizontalLayout\_21 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_21.setObjectName("horizontalLayout\_21")

spacerItem10 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_21.addItem(spacerItem10)

self.btn\_new\_book = QtWidgets.QPushButton(self.page\_6\_for\_admin)

self.btn\_new\_book.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_new\_book.setObjectName("btn\_new\_book")

self.horizontalLayout\_21.addWidget(self.btn\_new\_book)

spacerItem11 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_21.addItem(spacerItem11)

self.btn\_delete\_book = QtWidgets.QPushButton(self.page\_6\_for\_admin)

self.btn\_delete\_book.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_delete\_book.setObjectName("btn\_delete\_book")

self.horizontalLayout\_21.addWidget(self.btn\_delete\_book)

spacerItem12 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_21.addItem(spacerItem12)

self.btn\_save\_book = QtWidgets.QPushButton(self.page\_6\_for\_admin)

self.btn\_save\_book.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_save\_book.setObjectName("btn\_save\_book")

self.horizontalLayout\_21.addWidget(self.btn\_save\_book)

spacerItem13 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_21.addItem(spacerItem13)

self.verticalLayout\_14.addLayout(self.horizontalLayout\_21)

self.stackedWidget\_main.addWidget(self.page\_6\_for\_admin)

self.page = QtWidgets.QWidget()

self.page.setObjectName("page")

self.verticalLayout\_15 = QtWidgets.QVBoxLayout(self.page)

self.verticalLayout\_15.setObjectName("verticalLayout\_15")

self.horizontalLayout\_5 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_5.setObjectName("horizontalLayout\_5")

self.label\_20 = QtWidgets.QLabel(self.page)

self.label\_20.setStyleSheet("font: 11pt \"Comic Sans MS\";")

self.label\_20.setAlignment(QtCore.Qt.AlignCenter)

self.label\_20.setObjectName("label\_20")

self.horizontalLayout\_5.addWidget(self.label\_20)

self.verticalLayout\_15.addLayout(self.horizontalLayout\_5)

self.horizontalLayout\_8 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_8.setObjectName("horizontalLayout\_8")

self.label\_21 = QtWidgets.QLabel(self.page)

self.label\_21.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_21.setObjectName("label\_21")

self.horizontalLayout\_8.addWidget(self.label\_21)

self.edit\_genre = QtWidgets.QLineEdit(self.page)

self.edit\_genre.setObjectName("edit\_genre")

self.horizontalLayout\_8.addWidget(self.edit\_genre)

self.verticalLayout\_15.addLayout(self.horizontalLayout\_8)

self.table\_genre = QtWidgets.QTableWidget(self.page)

self.table\_genre.setObjectName("table\_genre")

self.table\_genre.setColumnCount(0)

self.table\_genre.setRowCount(0)

self.verticalLayout\_15.addWidget(self.table\_genre)

self.horizontalLayout\_20 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_20.setObjectName("horizontalLayout\_20")

spacerItem14 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_20.addItem(spacerItem14)

self.btn\_new\_genre = QtWidgets.QPushButton(self.page)

self.btn\_new\_genre.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_new\_genre.setObjectName("btn\_new\_genre")

self.horizontalLayout\_20.addWidget(self.btn\_new\_genre)

spacerItem15 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_20.addItem(spacerItem15)

self.btn\_delete\_genre = QtWidgets.QPushButton(self.page)

self.btn\_delete\_genre.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_delete\_genre.setObjectName("btn\_delete\_genre")

self.horizontalLayout\_20.addWidget(self.btn\_delete\_genre)

spacerItem16 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_20.addItem(spacerItem16)

self.btn\_save\_genre = QtWidgets.QPushButton(self.page)

self.btn\_save\_genre.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_save\_genre.setObjectName("btn\_save\_genre")

self.horizontalLayout\_20.addWidget(self.btn\_save\_genre)

spacerItem17 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_20.addItem(spacerItem17)

self.verticalLayout\_15.addLayout(self.horizontalLayout\_20)

self.stackedWidget\_main.addWidget(self.page)

self.page\_2 = QtWidgets.QWidget()

self.page\_2.setObjectName("page\_2")

self.verticalLayout\_16 = QtWidgets.QVBoxLayout(self.page\_2)

self.verticalLayout\_16.setObjectName("verticalLayout\_16")

self.horizontalLayout\_25 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_25.setObjectName("horizontalLayout\_25")

self.label\_22 = QtWidgets.QLabel(self.page\_2)

self.label\_22.setStyleSheet("font: 11pt \"Comic Sans MS\";")

self.label\_22.setAlignment(QtCore.Qt.AlignCenter)

self.label\_22.setObjectName("label\_22")

self.horizontalLayout\_25.addWidget(self.label\_22)

self.verticalLayout\_16.addLayout(self.horizontalLayout\_25)

self.verticalLayout\_17 = QtWidgets.QVBoxLayout()

self.verticalLayout\_17.setObjectName("verticalLayout\_17")

self.horizontalLayout\_29 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_29.setObjectName("horizontalLayout\_29")

self.label\_24 = QtWidgets.QLabel(self.page\_2)

self.label\_24.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_24.setObjectName("label\_24")

self.horizontalLayout\_29.addWidget(self.label\_24)

self.edit\_f = QtWidgets.QLineEdit(self.page\_2)

self.edit\_f.setObjectName("edit\_f")

self.horizontalLayout\_29.addWidget(self.edit\_f)

self.horizontalLayout\_29.setStretch(1, 100)

self.verticalLayout\_17.addLayout(self.horizontalLayout\_29)

self.horizontalLayout\_30 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_30.setObjectName("horizontalLayout\_30")

self.label\_25 = QtWidgets.QLabel(self.page\_2)

self.label\_25.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_25.setObjectName("label\_25")

self.horizontalLayout\_30.addWidget(self.label\_25)

self.edit\_i = QtWidgets.QLineEdit(self.page\_2)

self.edit\_i.setObjectName("edit\_i")

self.horizontalLayout\_30.addWidget(self.edit\_i)

self.horizontalLayout\_30.setStretch(1, 100)

self.verticalLayout\_17.addLayout(self.horizontalLayout\_30)

self.horizontalLayout\_31 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_31.setObjectName("horizontalLayout\_31")

self.label\_26 = QtWidgets.QLabel(self.page\_2)

self.label\_26.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_26.setObjectName("label\_26")

self.horizontalLayout\_31.addWidget(self.label\_26)

self.edit\_o = QtWidgets.QLineEdit(self.page\_2)

self.edit\_o.setObjectName("edit\_o")

self.horizontalLayout\_31.addWidget(self.edit\_o)

self.horizontalLayout\_31.setStretch(1, 100)

self.verticalLayout\_17.addLayout(self.horizontalLayout\_31)

self.horizontalLayout\_32 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_32.setObjectName("horizontalLayout\_32")

self.label\_23 = QtWidgets.QLabel(self.page\_2)

self.label\_23.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_23.setObjectName("label\_23")

self.horizontalLayout\_32.addWidget(self.label\_23)

self.date\_author = QtWidgets.QDateEdit(self.page\_2)

self.date\_author.setObjectName("date\_author")

self.horizontalLayout\_32.addWidget(self.date\_author)

self.horizontalLayout\_32.setStretch(1, 100)

self.verticalLayout\_17.addLayout(self.horizontalLayout\_32)

self.verticalLayout\_16.addLayout(self.verticalLayout\_17)

self.table\_author = QtWidgets.QTableWidget(self.page\_2)

self.table\_author.setObjectName("table\_author")

self.table\_author.setColumnCount(0)

self.table\_author.setRowCount(0)

self.verticalLayout\_16.addWidget(self.table\_author)

self.horizontalLayout\_27 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_27.setObjectName("horizontalLayout\_27")

spacerItem18 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_27.addItem(spacerItem18)

self.btn\_new\_author = QtWidgets.QPushButton(self.page\_2)

self.btn\_new\_author.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_new\_author.setObjectName("btn\_new\_author")

self.horizontalLayout\_27.addWidget(self.btn\_new\_author)

spacerItem19 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_27.addItem(spacerItem19)

self.btn\_delete\_author = QtWidgets.QPushButton(self.page\_2)

self.btn\_delete\_author.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_delete\_author.setObjectName("btn\_delete\_author")

self.horizontalLayout\_27.addWidget(self.btn\_delete\_author)

spacerItem20 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_27.addItem(spacerItem20)

self.btn\_save\_author = QtWidgets.QPushButton(self.page\_2)

self.btn\_save\_author.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_save\_author.setObjectName("btn\_save\_author")

self.horizontalLayout\_27.addWidget(self.btn\_save\_author)

spacerItem21 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_27.addItem(spacerItem21)

self.verticalLayout\_16.addLayout(self.horizontalLayout\_27)

self.stackedWidget\_main.addWidget(self.page\_2)

self.page\_3 = QtWidgets.QWidget()

self.page\_3.setObjectName("page\_3")

self.verticalLayout\_19 = QtWidgets.QVBoxLayout(self.page\_3)

self.verticalLayout\_19.setObjectName("verticalLayout\_19")

self.verticalLayout\_18 = QtWidgets.QVBoxLayout()

self.verticalLayout\_18.setObjectName("verticalLayout\_18")

self.label\_27 = QtWidgets.QLabel(self.page\_3)

self.label\_27.setStyleSheet("font: 11pt \"Comic Sans MS\";")

self.label\_27.setAlignment(QtCore.Qt.AlignCenter)

self.label\_27.setObjectName("label\_27")

self.verticalLayout\_18.addWidget(self.label\_27)

self.verticalLayout\_19.addLayout(self.verticalLayout\_18)

self.horizontalLayout\_26 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_26.setObjectName("horizontalLayout\_26")

self.label\_28 = QtWidgets.QLabel(self.page\_3)

self.label\_28.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_28.setObjectName("label\_28")

self.horizontalLayout\_26.addWidget(self.label\_28)

self.edit\_ph = QtWidgets.QLineEdit(self.page\_3)

self.edit\_ph.setObjectName("edit\_ph")

self.horizontalLayout\_26.addWidget(self.edit\_ph)

self.verticalLayout\_19.addLayout(self.horizontalLayout\_26)

self.table\_ph = QtWidgets.QTableWidget(self.page\_3)

self.table\_ph.setObjectName("table\_ph")

self.table\_ph.setColumnCount(0)

self.table\_ph.setRowCount(0)

self.verticalLayout\_19.addWidget(self.table\_ph)

self.horizontalLayout\_33 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_33.setObjectName("horizontalLayout\_33")

spacerItem22 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_33.addItem(spacerItem22)

self.btn\_new\_ph = QtWidgets.QPushButton(self.page\_3)

self.btn\_new\_ph.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_new\_ph.setObjectName("btn\_new\_ph")

self.horizontalLayout\_33.addWidget(self.btn\_new\_ph)

spacerItem23 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_33.addItem(spacerItem23)

self.btn\_delete\_ph = QtWidgets.QPushButton(self.page\_3)

self.btn\_delete\_ph.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_delete\_ph.setObjectName("btn\_delete\_ph")

self.horizontalLayout\_33.addWidget(self.btn\_delete\_ph)

spacerItem24 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_33.addItem(spacerItem24)

self.btn\_save\_ph = QtWidgets.QPushButton(self.page\_3)

self.btn\_save\_ph.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_save\_ph.setObjectName("btn\_save\_ph")

self.horizontalLayout\_33.addWidget(self.btn\_save\_ph)

spacerItem25 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_33.addItem(spacerItem25)

self.verticalLayout\_19.addLayout(self.horizontalLayout\_33)

self.stackedWidget\_main.addWidget(self.page\_3)

self.verticalLayout.addWidget(self.stackedWidget\_main)

self.verticalLayout\_2.addLayout(self.verticalLayout)

self.horizontalLayout = QtWidgets.QHBoxLayout()

self.horizontalLayout.setSpacing(0)

self.horizontalLayout.setObjectName("horizontalLayout")

spacerItem26 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout.addItem(spacerItem26)

self.btn\_exit = QtWidgets.QPushButton(self.centralwidget)

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(9)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.btn\_exit.setFont(font)

self.btn\_exit.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_exit.setObjectName("btn\_exit")

self.horizontalLayout.addWidget(self.btn\_exit)

spacerItem27 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout.addItem(spacerItem27)

self.btn\_back = QtWidgets.QPushButton(self.centralwidget)

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(9)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.btn\_back.setFont(font)

self.btn\_back.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_back.setObjectName("btn\_back")

self.horizontalLayout.addWidget(self.btn\_back)

spacerItem28 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout.addItem(spacerItem28)

self.btn\_next = QtWidgets.QPushButton(self.centralwidget)

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(9)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.btn\_next.setFont(font)

self.btn\_next.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_next.setObjectName("btn\_next")

self.horizontalLayout.addWidget(self.btn\_next)

spacerItem29 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout.addItem(spacerItem29)

self.horizontalLayout.setStretch(0, 1)

self.horizontalLayout.setStretch(1, 1)

self.horizontalLayout.setStretch(2, 2)

self.horizontalLayout.setStretch(3, 1)

self.horizontalLayout.setStretch(4, 1)

self.horizontalLayout.setStretch(5, 1)

self.horizontalLayout.setStretch(6, 4)

self.verticalLayout\_2.addLayout(self.horizontalLayout)

BookShop.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(BookShop)

self.menubar.setGeometry(QtCore.QRect(0, 0, 818, 21))

self.menubar.setObjectName("menubar")

BookShop.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(BookShop)

self.statusbar.setObjectName("statusbar")

BookShop.setStatusBar(self.statusbar)

self.retranslateUi(BookShop)

self.stackedWidget\_main.setCurrentIndex(2)

self.add\_new\_field.setCurrentRow(-1)

self.stackedWidget\_report.setCurrentIndex(0)

QtCore.QMetaObject.connectSlotsByName(BookShop)

def retranslateUi(self, BookShop):

\_translate = QtCore.QCoreApplication.translate

BookShop.setWindowTitle(\_translate("BookShop", "Книжный магазин \"Дом книги\""))

self.label.setText(\_translate("BookShop", "Книжный магазин \"Дом книги\""))

self.label\_3.setText(\_translate("BookShop", "Персональные данные сотрудника"))

self.label\_4.setText(\_translate("BookShop", "ФИО: "))

self.lbl\_fio.setText(\_translate("BookShop", "Федоров Фёдор Фёдорович (для примера)"))

self.label\_2.setText(\_translate("BookShop", "Роль:"))

self.lbl\_role.setText(\_translate("BookShop", "администратор"))

self.label\_5.setText(\_translate("BookShop", "Просмотр истории входа сотрудников"))

self.table\_entry.setSortingEnabled(True)

self.label\_6.setText(\_translate("BookShop", "Создание заказа"))

self.label\_8.setText(\_translate("BookShop", "Выберите клиента: "))

self.label\_9.setText(\_translate("BookShop", "Выберите книгу: "))

self.btn\_new\_order.setText(\_translate("BookShop", " Создать заказ "))

self.btn\_save\_request.setText(\_translate("BookShop", " Сохранить заказ "))

self.btn\_code.setText(\_translate("BookShop", " Штрих-код "))

self.btn\_all\_clients.setText(\_translate("BookShop", " Все клиенты "))

self.btn\_new\_client.setText(\_translate("BookShop", " Новый клиент "))

self.label\_11.setText(\_translate("BookShop", "Просмотр всех клиентов"))

self.table\_clients.setSortingEnabled(False)

item = self.table\_clients.horizontalHeaderItem(0)

item.setText(\_translate("BookShop", "id"))

item = self.table\_clients.horizontalHeaderItem(1)

item.setText(\_translate("BookShop", "ФИО"))

item = self.table\_clients.horizontalHeaderItem(2)

item.setText(\_translate("BookShop", "Код клиента"))

item = self.table\_clients.horizontalHeaderItem(3)

item.setText(\_translate("BookShop", "Паспортные данные"))

item = self.table\_clients.horizontalHeaderItem(4)

item.setText(\_translate("BookShop", "Дата рождения"))

item = self.table\_clients.horizontalHeaderItem(5)

item.setText(\_translate("BookShop", "Адрес"))

item = self.table\_clients.horizontalHeaderItem(6)

item.setText(\_translate("BookShop", "email"))

self.label\_10.setText(\_translate("BookShop", "Создание отчёта"))

self.label\_12.setText(\_translate("BookShop", "Введите период времени:"))

self.date\_start.setDisplayFormat(\_translate("BookShop", "dd.MM.yyyy"))

self.date\_end.setDisplayFormat(\_translate("BookShop", "dd.MM.yyyy"))

self.btn\_count\_book.setText(\_translate("BookShop", "кол-во книг по дням за период времени"))

self.btn\_count\_order\_book.setText(\_translate("BookShop", "кол-во заказов по дням за период времени по каждой книге"))

self.btn\_count\_order.setText(\_translate("BookShop", "кол-во заказов по дням за период времени"))

self.label\_13.setText(\_translate("BookShop", "тут должен быть график (можно сделать через matplotlib)"))

self.btn\_pdf.setText(\_translate("BookShop", "сохранить в pdf"))

self.label\_14.setText(\_translate("BookShop", "Редактирование книг"))

self.label\_15.setText(\_translate("BookShop", "Введите название книги:"))

self.label\_18.setText(\_translate("BookShop", "Введите стоимость книги (руб):"))

self.label\_19.setText(\_translate("BookShop", "Выберите жанр:"))

self.label\_17.setText(\_translate("BookShop", "Выберите автора:"))

self.label\_16.setText(\_translate("BookShop", "Выберите издательство:"))

self.btn\_new\_book.setText(\_translate("BookShop", " Добавить книгу "))

self.btn\_delete\_book.setText(\_translate("BookShop", " Удалить книгу "))

self.btn\_save\_book.setText(\_translate("BookShop", " Сохранить книгу "))

self.label\_20.setText(\_translate("BookShop", "Редактирование жанров"))

self.label\_21.setText(\_translate("BookShop", "Введите жанр: "))

self.btn\_new\_genre.setText(\_translate("BookShop", " Добавить жанр "))

self.btn\_delete\_genre.setText(\_translate("BookShop", " Удалить жанр "))

self.btn\_save\_genre.setText(\_translate("BookShop", " Сохранить жанр "))

self.label\_22.setText(\_translate("BookShop", "Редактирование авторов"))

self.label\_24.setText(\_translate("BookShop", "Введите фамилию: "))

self.label\_25.setText(\_translate("BookShop", "Введите имя: "))

self.label\_26.setText(\_translate("BookShop", "Введите отчество: "))

self.label\_23.setText(\_translate("BookShop", "Укажите дату рождения: "))

self.btn\_new\_author.setText(\_translate("BookShop", " Добавить автора "))

self.btn\_delete\_author.setText(\_translate("BookShop", " Удалить автора "))

self.btn\_save\_author.setText(\_translate("BookShop", " Сохранить автора "))

self.label\_27.setText(\_translate("BookShop", "Редактирование издательств"))

self.label\_28.setText(\_translate("BookShop", "Введите название издательства: "))

self.btn\_new\_ph.setText(\_translate("BookShop", " Добавить издательство "))

self.btn\_delete\_ph.setText(\_translate("BookShop", " Удалить издательство "))

self.btn\_save\_ph.setText(\_translate("BookShop", " Сохранить издательство "))

self.btn\_exit.setText(\_translate("BookShop", " Выход "))

self.btn\_back.setText(\_translate("BookShop", " Назад "))

self.btn\_next.setText(\_translate("BookShop", " Вперед "))

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

import sys

app = QtWidgets.QApplication(sys.argv)

BookShop = QtWidgets.QMainWindow()

ui = Ui\_BookShop()

ui.setupUi(BookShop)

BookShop.show()

sys.exit(app.exec\_())

New\_client.py

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_Dialog(object):

def setupUi(self, Dialog):

Dialog.setObjectName("Dialog")

Dialog.resize(665, 600)

Dialog.setStyleSheet("background-color: rgb(255, 240, 224);")

self.verticalLayout = QtWidgets.QVBoxLayout(Dialog)

self.verticalLayout.setObjectName("verticalLayout")

self.horizontalLayout\_2 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_2.setContentsMargins(12, -1, 0, -1)

self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")

spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_2.addItem(spacerItem)

self.label = QtWidgets.QLabel(Dialog)

self.label.setMinimumSize(QtCore.QSize(0, 0))

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(13)

self.label.setFont(font)

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

self.horizontalLayout\_2.addWidget(self.label)

spacerItem1 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_2.addItem(spacerItem1)

self.logo = QtWidgets.QLabel(Dialog)

self.logo.setMaximumSize(QtCore.QSize(80, 80))

self.logo.setText("")

self.logo.setPixmap(QtGui.QPixmap("img/logo\_mini.png"))

self.logo.setScaledContents(True)

self.logo.setObjectName("logo")

self.horizontalLayout\_2.addWidget(self.logo)

self.horizontalLayout\_2.setStretch(0, 3)

self.horizontalLayout\_2.setStretch(1, 2)

self.horizontalLayout\_2.setStretch(2, 2)

self.horizontalLayout\_2.setStretch(3, 1)

self.verticalLayout.addLayout(self.horizontalLayout\_2)

self.verticalLayout\_2 = QtWidgets.QVBoxLayout()

self.verticalLayout\_2.setObjectName("verticalLayout\_2")

self.label\_2 = QtWidgets.QLabel(Dialog)

self.label\_2.setStyleSheet("font: 11pt \"Comic Sans MS\";")

self.label\_2.setAlignment(QtCore.Qt.AlignCenter)

self.label\_2.setObjectName("label\_2")

self.verticalLayout\_2.addWidget(self.label\_2)

self.verticalLayout\_5 = QtWidgets.QVBoxLayout()

self.verticalLayout\_5.setObjectName("verticalLayout\_5")

self.horizontalLayout\_4 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_4.setObjectName("horizontalLayout\_4")

self.label\_5 = QtWidgets.QLabel(Dialog)

self.label\_5.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_5.setObjectName("label\_5")

self.horizontalLayout\_4.addWidget(self.label\_5)

self.edit\_surname = QtWidgets.QLineEdit(Dialog)

self.edit\_surname.setObjectName("edit\_surname")

self.horizontalLayout\_4.addWidget(self.edit\_surname)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_4)

self.horizontalLayout\_20 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_20.setObjectName("horizontalLayout\_20")

self.label\_3 = QtWidgets.QLabel(Dialog)

self.label\_3.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_3.setObjectName("label\_3")

self.horizontalLayout\_20.addWidget(self.label\_3)

self.edit\_name = QtWidgets.QLineEdit(Dialog)

self.edit\_name.setObjectName("edit\_name")

self.horizontalLayout\_20.addWidget(self.edit\_name)

self.horizontalLayout\_20.setStretch(1, 100)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_20)

self.horizontalLayout\_19 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_19.setObjectName("horizontalLayout\_19")

self.label\_18 = QtWidgets.QLabel(Dialog)

self.label\_18.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_18.setObjectName("label\_18")

self.horizontalLayout\_19.addWidget(self.label\_18)

self.edit\_lastname = QtWidgets.QLineEdit(Dialog)

self.edit\_lastname.setObjectName("edit\_lastname")

self.horizontalLayout\_19.addWidget(self.edit\_lastname)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_19)

self.horizontalLayout\_5 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_5.setSpacing(6)

self.horizontalLayout\_5.setObjectName("horizontalLayout\_5")

self.label\_7 = QtWidgets.QLabel(Dialog)

self.label\_7.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_7.setObjectName("label\_7")

self.horizontalLayout\_5.addWidget(self.label\_7)

self.date\_birth = QtWidgets.QDateEdit(Dialog)

self.date\_birth.setObjectName("date\_birth")

self.horizontalLayout\_5.addWidget(self.date\_birth)

self.horizontalLayout\_5.setStretch(1, 100)

self.verticalLayout\_5.addLayout(self.horizontalLayout\_5)

self.horizontalLayout = QtWidgets.QHBoxLayout()

self.horizontalLayout.setObjectName("horizontalLayout")

self.label\_4 = QtWidgets.QLabel(Dialog)

self.label\_4.setStyleSheet("background-color: rgb(216, 0, 4);\n"

"color: rgb(255, 255, 255);\n"

"font: 9pt \"Comic Sans MS\";")

self.label\_4.setObjectName("label\_4")

self.horizontalLayout.addWidget(self.label\_4)

self.edit\_email = QtWidgets.QLineEdit(Dialog)

self.edit\_email.setObjectName("edit\_email")

self.horizontalLayout.addWidget(self.edit\_email)

self.verticalLayout\_5.addLayout(self.horizontalLayout)

self.verticalLayout\_2.addLayout(self.verticalLayout\_5)

spacerItem2 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

self.verticalLayout\_2.addItem(spacerItem2)

self.horizontalLayout\_3 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_3.setObjectName("horizontalLayout\_3")

spacerItem3 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_3.addItem(spacerItem3)

self.btn\_add\_client = QtWidgets.QPushButton(Dialog)

self.btn\_add\_client.setStyleSheet("background-color: rgb(143, 0, 2);\n"

"font: 9pt \"Comic Sans MS\";\n"

"color: rgb(255, 255, 255);")

self.btn\_add\_client.setObjectName("btn\_add\_client")

self.horizontalLayout\_3.addWidget(self.btn\_add\_client)

spacerItem4 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_3.addItem(spacerItem4)

self.verticalLayout\_2.addLayout(self.horizontalLayout\_3)

self.verticalLayout.addLayout(self.verticalLayout\_2)

self.retranslateUi(Dialog)

QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self, Dialog):

\_translate = QtCore.QCoreApplication.translate

Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))

self.label.setText(\_translate("Dialog", "Книжный магазин \"Дом книги\""))

self.label\_2.setText(\_translate("Dialog", "Добавить нового клиента"))

self.label\_5.setText(\_translate("Dialog", "Введите фамилию:"))

self.label\_3.setText(\_translate("Dialog", "Введите имя:"))

self.label\_18.setText(\_translate("Dialog", "Введите отчество:"))

self.label\_7.setText(\_translate("Dialog", "Введите дату рождения:"))

self.label\_4.setText(\_translate("Dialog", "Введите e-mail:"))

self.btn\_add\_client.setText(\_translate("Dialog", "Добавить нового клиента"))

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

import sys

app = QtWidgets.QApplication(sys.argv)

Dialog = QtWidgets.QDialog()

ui = Ui\_Dialog()

ui.setupUi(Dialog)

Dialog.show()

sys.exit(app.exec\_())

# **ПРИЛОЖЕНИЕ Г Исходный код программы**

main.py

from io import BytesIO

import barcode

from barcode.writer import ImageWriter

import img2pdf

from facade import Facade

import random

from PyQt5.QtGui import QFont, QPixmap

from PyQt5 import QtWidgets

from PyQt5 import uic, QtCore, QtGui

from PyQt5.QtCore import Qt, QTimer, QTime, QDateTime

from PyQt5.QtWidgets import QGraphicsScene, QListWidgetItem, QAction

from PyQt5.QtWidgets import QMainWindow, QApplication, QTableWidget, QTableWidgetItem, QDialog

import sys

import time

import datetime

import logging

from PyQt5.QtWidgets import QMessageBox

logging.basicConfig(level=logging.INFO)

class MainWindow(QMainWindow):

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

"""

Подключением к кнопкам, объявление переменных, заполнение таблиц,

получение списка страниц StackedWidget.

"""

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

self.facade = Facade()

self.ui = uic.loadUi("main.ui", self)

self.page = self.ui.stackedWidget\_main

self.page\_id = [0]

self.now\_page = 0

self.block\_req = True

self.page.setCurrentIndex(self.page\_id[self.now\_page])

self.ui.btn\_next.clicked.connect(self.next\_page)

self.ui.btn\_back.clicked.connect(self.back\_page)

self.ui.btn\_all\_clients.clicked.connect(self.page\_all\_clients)

self.ui.btn\_exit.clicked.connect(lambda: self.exit('Неуспешно'))

self.ui.btn\_new\_book.clicked.connect(self.new\_book)

self.ui.btn\_delete\_book.clicked.connect(self.delete\_book)

self.ui.btn\_save\_book.clicked.connect(self.save\_book)

self.ui.btn\_new\_genre.clicked.connect(self.new\_genre)

self.ui.btn\_delete\_genre.clicked.connect(self.delete\_genre)

self.ui.btn\_save\_genre.clicked.connect(self.save\_genre)

self.ui.btn\_new\_author.clicked.connect(self.new\_author)

self.ui.btn\_delete\_author.clicked.connect(self.delete\_author)

self.ui.btn\_save\_author.clicked.connect(self.save\_author)

self.ui.btn\_new\_ph.clicked.connect(self.new\_ph)

self.ui.btn\_delete\_ph.clicked.connect(self.delete\_ph)

self.ui.btn\_save\_ph.clicked.connect(self.save\_ph)

self.build\_combobox\_clients()

self.build\_combobox\_books()

self.build\_combobox\_genre()

self.build\_combobox\_author()

self.build\_combobox\_ph()

self.ui.btn\_new\_order.clicked.connect(self.add\_new\_request)

self.ui.btn\_save\_request.clicked.connect(self.save\_request)

self.ui.btn\_plus.clicked.connect(self.add\_book\_to\_request)

self.ui.btn\_code.clicked.connect(self.generateCode)

self.ui.btn\_new\_client.clicked.connect(self.open\_new\_client)

self.ui.btn\_count\_order.clicked.connect(self.count\_order)

self.ui.btn\_count\_book.clicked.connect(self.count\_book)

self.ui.btn\_count\_order\_book.clicked.connect(self.count\_order\_book)

self.type = 0

self.dict = {}

self.ui.btn\_pdf.clicked.connect(lambda: self.order\_pdf(self.type, self.dict))

self.updateTableBook()

self.updateTableGenre()

self.updateTableAuthor()

self.updateTablePH()

self.updateTableHistory()

quit = QAction("Quit", self)

quit.triggered.connect(self.closeEvent)

logging.log(logging.INFO, 'Приложение запущено')

def exit(self, block):

"""

Выход из программы.

:param block: блокировка

"""

self.now\_page = 0

self.page.setCurrentIndex(self.page\_id[self.now\_page])

t = time.localtime()

time\_string = time.strftime("%d.%m.%Y %H:%M:%S", t) # время выхода

self.facade.insert\_time\_exit(self.emp, time\_string, block)

self.hide()

self.open\_auth()

logging.log(logging.INFO, 'Произведён выход через кнопку "Выход"')

def closeEvent(self, event):

"""

Выход из программы.

:param event: событие

"""

block = 'Неуспешно'

self.now\_page = 0

self.page.setCurrentIndex(self.page\_id[self.now\_page])

t = time.localtime()

time\_string = time.strftime("%d.%m.%Y %H:%M:%S", t)

self.facade.insert\_time\_exit(self.emp, time\_string, block)

event.accept()

self.hide()

self.open\_auth()

logging.log(logging.INFO, 'Произведён выход через кнопку "Х"')

def page\_all\_clients(self):

"""

Переход к странице с таблицей клиентов.

"""

self.updateTableClient()

self.page.setCurrentIndex(3)

logging.log(logging.INFO, 'Переход к таблице "Клиент"')

def updateTableClient(self):

"""

Обновление таблицы `Клиент`.

"""

self.table\_book.clear()

rec = self.facade.read\_clients()

self.ui.table\_clients.setColumnCount(6)

self.ui.table\_clients.setRowCount(len(rec))

self.ui.table\_clients.setHorizontalHeaderLabels(['Код клиента', 'Фамилия', 'Имя', 'Отчество', 'Дата рождения', 'e-mail'])

for i, client in enumerate(rec):

for x, field in enumerate(client):

item = QTableWidgetItem()

item.setText(str(field))

if x == 0:

item.setFlags(Qt.ItemIsEnabled)

self.ui.table\_clients.setItem(i, x, item)

logging.log(logging.INFO, 'Таблица "Клиент" обновлена')

def updateTableBook(self):

"""

Обновление таблицы `Книга`.

"""

self.table\_book.clear()

rec = self.facade.read\_books()

self.ui.table\_book.setColumnCount(6)

self.ui.table\_book.setRowCount(len(rec))

self.ui.table\_book.setHorizontalHeaderLabels(['Код книги', 'Название книги', 'Стоимость', 'Жанр', 'Автор', 'Издательство'])

for i, book in enumerate(rec):

for x, field in enumerate(book):

item = QTableWidgetItem()

item.setText(str(field))

if x == 0:

item.setFlags(Qt.ItemIsEnabled)

self.ui.table\_book.setItem(i, x, item)

logging.log(logging.INFO, 'Таблица "Книга" обновлена')

def updateTableGenre(self):

"""

Обновление таблицы `Жанр`.

"""

self.table\_genre.clear()

rec = self.facade.read\_genre()

self.ui.table\_genre.setColumnCount(2)

self.ui.table\_genre.setRowCount(len(rec))

self.ui.table\_genre.setHorizontalHeaderLabels(['Код жанра', 'Жанр'])

for i, book in enumerate(rec):

for x, field in enumerate(book):

item = QTableWidgetItem()

item.setText(str(field))

if x == 0:

item.setFlags(Qt.ItemIsEnabled)

self.ui.table\_genre.setItem(i, x, item)

logging.log(logging.INFO, 'Таблица "Жанр" обновлена')

def updateTableAuthor(self):

"""

Обновление таблицы `Автор`.

"""

self.table\_author.clear()

rec = self.facade.read\_author()

self.ui.table\_author.setColumnCount(5)

self.ui.table\_author.setRowCount(len(rec))

self.ui.table\_author.setHorizontalHeaderLabels(['Код автора', 'Фамилия', 'Имя', 'Отчество', 'Дата рождения'])

for i, book in enumerate(rec):

for x, field in enumerate(book):

item = QTableWidgetItem()

item.setText(str(field))

if x == 0:

item.setFlags(Qt.ItemIsEnabled)

self.ui.table\_author.setItem(i, x, item)

logging.log(logging.INFO, 'Таблица "Автор" обновлена')

def updateTablePH(self):

"""

Обновление таблицы `Издательство`.

"""

self.table\_ph.clear()

rec = self.facade.read\_ph()

self.ui.table\_ph.setColumnCount(2)

self.ui.table\_ph.setRowCount(len(rec))

self.ui.table\_ph.setHorizontalHeaderLabels(['Код издательства', 'Название издательства'])

for i, book in enumerate(rec):

for x, field in enumerate(book):

item = QTableWidgetItem()

item.setText(str(field))

if x == 0:

item.setFlags(Qt.ItemIsEnabled)

self.ui.table\_ph.setItem(i, x, item)

logging.log(logging.INFO, 'Таблица "Издательство" обновлена')

def updateTableHistory(self):

"""

Обновление таблицы `История входа`.

"""

self.table\_entry.clear()

rec = self.facade.read\_history()

self.table\_entry.setColumnCount(5)

self.table\_entry.setRowCount(len(rec))

self.table\_entry.setHorizontalHeaderLabels(['ID', 'Дата входа', 'Дата выхода', 'Блокировка', 'Логин сотрудника'])

for i, employee in enumerate(rec):

for x, info in enumerate(employee):

item = QTableWidgetItem()

item.setText(str(info))

if x == 0:

item.setFlags(Qt.ItemIsEnabled)

self.ui.table\_entry.setItem(i, x, item)

logging.log(logging.INFO, 'Таблица "История входа" обновлена')

def new\_book(self):

"""

Добавление книги.

"""

title\_book = self.ui.edit\_title\_book.text()

cost\_book = self.ui.spin\_cost.value()

genre = self.facade.get\_genre\_id(self.ui.comboBox\_genre.currentText())

au = self.ui.comboBox\_author.currentText()

author = self.facade.get\_author\_id(au)

ph = self.facade.get\_ph\_id(self.ui.comboBox\_ph.currentText())

if title\_book != '' and cost\_book != '' and genre != '' and author != '' and ph != '':

self.facade.insert\_book(title\_book, cost\_book, genre, author, ph)

self.updateTableBook()

logging.log(logging.INFO, 'Добавлены данные в таблицу "Книга"')

def delete\_book(self):

"""

Удаление выбранной книги.

"""

SelectedRow = self.table\_book.currentRow()

rowcount = self.table\_book.rowCount()

colcount = self.table\_book.columnCount()

if rowcount == 0:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Warning)

msg.setText("В таблице нет данных!")

msg.setWindowTitle("Ошибка")

msg.setStandardButtons(QMessageBox.Ok)

retval = msg.exec\_()

elif SelectedRow == -1:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Warning)

msg.setText("Выберите поле для удаления!")

msg.setWindowTitle("Ошибка")

msg.setStandardButtons(QMessageBox.Ok)

retval = msg.exec\_()

else:

for col in range(1, colcount):

self.table\_book.setItem(SelectedRow, col, QTableWidgetItem(''))

ix = self.table\_book.model().index(-1, -1)

self.table\_book.setCurrentIndex(ix)

logging.log(logging.INFO, 'Удалены данные из таблицы "Книга"')

def get\_from\_table\_book(self):

"""

Получение данных из таблицы `Книга` для записи в базу данных.

:return: data

"""

rows = self.table\_book.rowCount()

cols = self.table\_book.columnCount()

data = []

for row in range(rows):

tmp = []

for col in range(cols):

tmp.append(self.table\_book.item(row, col).text())

data.append(tmp)

return data

logging.log(logging.INFO, 'Получены данные для сохранения в таблицу "Книга"')

def save\_book(self):

"""

Сохранение данных о книгах в базу данных.

"""

data = self.get\_from\_table\_book()

for string in data:

if string[1] != '':

self.facade.update\_book(int(string[0]), string[1], string[2], string[3], string[4], string[5])

else:

self.facade.delete\_book(int(string[0]))

self.updateTableBook()

logging.log(logging.INFO, 'Данные сохранены в таблицу "Книга"')

def new\_genre(self):

"""

Добавление жанра.

"""

title\_genre = self.ui.edit\_genre.text()

if title\_genre != '':

self.facade.insert\_genre(title\_genre)

self.updateTableGenre()

logging.log(logging.INFO, 'Добавлены данные в таблицу "Жанр"')

def delete\_genre(self):

"""

Удаление выбранного жанра.

"""

SelectedRow = self.table\_genre.currentRow()

rowcount = self.table\_genre.rowCount()

colcount = self.table\_genre.columnCount()

if rowcount == 0:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Warning)

msg.setText("В таблице нет данных!")

msg.setWindowTitle("Ошибка")

msg.setStandardButtons(QMessageBox.Ok)

retval = msg.exec\_()

elif SelectedRow == -1:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Warning)

msg.setText("Выберите поле для удаления!")

msg.setWindowTitle("Ошибка")

msg.setStandardButtons(QMessageBox.Ok)

retval = msg.exec\_()

else:

for col in range(1, colcount):

self.table\_genre.setItem(SelectedRow, col, QTableWidgetItem(''))

ix = self.table\_genre.model().index(-1, -1)

self.table\_genre.setCurrentIndex(ix)

logging.log(logging.INFO, 'Удалены данные из таблицы "Жанр"')

def get\_from\_table\_genre(self):

"""

Получение данных из таблицы `Жанр` для записи в базу данных.

:return: data

"""

rows = self.table\_genre.rowCount()

cols = self.table\_genre.columnCount()

data = []

for row in range(rows):

tmp = []

for col in range(cols):

tmp.append(self.table\_genre.item(row, col).text())

data.append(tmp)

return data

logging.log(logging.INFO, 'Получены данные для сохранения в таблицу "Жанр"')

def save\_genre(self):

"""

Сохранение данных о жанрах в базу данных.

"""

data = self.get\_from\_table\_genre()

for string in data:

if string[1] != '':

self.facade.update\_genre(int(string[0]), string[1])

else:

self.facade.delete\_genre(int(string[0]))

self.updateTableGenre()

logging.log(logging.INFO, 'Данные сохранены в таблицу "Жанр"')

def new\_author(self):

"""

Добавление автора.

"""

fio = []

f = self.ui.edit\_f.text()

i = self.ui.edit\_i.text()

o = self.ui.edit\_o.text()

date = self.ui.date\_author.dateTime().toString('dd-MM-yyyy')

fio.append(f)

fio.append(i)

fio.append(o)

if f != '' and i != '' and date != '':

self.facade.insert\_author(fio, date)

self.updateTableAuthor()

logging.log(logging.INFO, 'Добавлены данные в таблицу "Автор"')

def delete\_author(self):

"""

Удаление выбранного автора.

"""

SelectedRow = self.table\_author.currentRow()

rowcount = self.table\_author.rowCount()

colcount = self.table\_author.columnCount()

if rowcount == 0:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Warning)

msg.setText("В таблице нет данных!")

msg.setWindowTitle("Ошибка")

msg.setStandardButtons(QMessageBox.Ok)

retval = msg.exec\_()

elif SelectedRow == -1:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Warning)

msg.setText("Выберите поле для удаления!")

msg.setWindowTitle("Ошибка")

msg.setStandardButtons(QMessageBox.Ok)

retval = msg.exec\_()

else:

for col in range(1, colcount):

self.table\_author.setItem(SelectedRow, col, QTableWidgetItem(''))

ix = self.table\_author.model().index(-1, -1)

self.table\_author.setCurrentIndex(ix)

logging.log(logging.INFO, 'Удалены данные из таблицы "Автор"')

def get\_from\_table\_author(self):

"""

Получение данных из таблицы `Автор` для записи в базу данных.

:return: data

"""

rows = self.table\_author.rowCount()

cols = self.table\_author.columnCount()

data = []

for row in range(rows):

tmp = []

for col in range(cols):

tmp.append(self.table\_author.item(row, col).text())

data.append(tmp)

return data

logging.log(logging.INFO, 'Получены данные для сохранения в таблицу "Автор"')

def save\_author(self):

"""

Сохранение данных об авторах в базу данных.

"""

fio = []

data = self.get\_from\_table\_author()

for string in data:

if string[1] != '':

fio.append(string[1])

fio.append(string[2])

fio.append(string[3])

self.facade.update\_author(int(string[0]), fio, string[4])

fio.clear()

else:

self.facade.delete\_author(int(string[0]))

self.updateTableAuthor()

logging.log(logging.INFO, 'Данные сохранены в таблицу "Автор"')

def new\_ph(self):

"""

Добавление издательства.

"""

title\_ph = self.ui.edit\_ph.text()

if title\_ph != '':

self.facade.insert\_ph(title\_ph)

self.updateTablePH()

logging.log(logging.INFO, 'Добавлены данные в таблицу "Издательство"')

def delete\_ph(self):

"""

Удаление выбранного издательства.

"""

SelectedRow = self.table\_ph.currentRow()

rowcount = self.table\_ph.rowCount()

colcount = self.table\_ph.columnCount()

if rowcount == 0:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Warning)

msg.setText("В таблице нет данных!")

msg.setWindowTitle("Ошибка")

msg.setStandardButtons(QMessageBox.Ok)

retval = msg.exec\_()

elif SelectedRow == -1:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Warning)

msg.setText("Выберите поле для удаления!")

msg.setWindowTitle("Ошибка")

msg.setStandardButtons(QMessageBox.Ok)

retval = msg.exec\_()

else:

for col in range(1, colcount):

self.table\_ph.setItem(SelectedRow, col, QTableWidgetItem(''))

ix = self.table\_ph.model().index(-1, -1)

self.table\_ph.setCurrentIndex(ix)

logging.log(logging.INFO, 'Удалены данные из таблицы "Издательство"')

def get\_from\_table\_ph(self):

"""

Получение данных из таблицы `Издательство` для записи в базу данных.

:return: data

"""

rows = self.table\_ph.rowCount()

cols = self.table\_ph.columnCount()

data = []

for row in range(rows):

tmp = []

for col in range(cols):

tmp.append(self.table\_ph.item(row, col).text())

data.append(tmp)

return data

logging.log(logging.INFO, 'Получены данные для сохранения в таблицу "Издательство"')

def save\_ph(self):

"""

Сохранение данных об издательствах в базу данных

"""

data = self.get\_from\_table\_ph()

for string in data:

if string[1] != '':

self.facade.update\_ph(int(string[0]), string[1])

else:

self.facade.delete\_ph(int(string[0]))

self.updateTablePH()

logging.log(logging.INFO, 'Данные сохранены в таблицу "Издательство"')

def build\_combobox\_clients(self):

"""

Добавление списка клиентов в Combo Box.

"""

clients = self.facade.get\_clients()

self.comboBox\_clients.clear()

if self.comboBox\_clients is not None:

self.comboBox\_clients.addItems(clients)

logging.log(logging.INFO, 'Combo Box "Клиенты" обновлён')

def build\_combobox\_books(self):

"""

Добавление списка книг в Combo Box.

"""

books = self.facade.get\_books()

self.comboBox\_book.clear()

if self.comboBox\_book is not None:

self.comboBox\_book.addItems(books)

logging.log(logging.INFO, 'Combo Box "Книги" обновлён')

def build\_combobox\_genre(self):

"""

Добавление списка жанров в Combo Box.

"""

genres = self.facade.get\_genres()

self.comboBox\_genre.clear()

if self.comboBox\_genre is not None:

self.comboBox\_genre.addItems(genres)

logging.log(logging.INFO, 'Combo Box "Жанры" обновлён')

def build\_combobox\_author(self):

"""

Добавление списка авторов в Combo Box.

"""

authors = self.facade.get\_authors()

self.comboBox\_author.clear()

if self.comboBox\_author is not None:

self.comboBox\_author.addItems(authors)

logging.log(logging.INFO, 'ComboBox "Авторы" обновлён')

def build\_combobox\_ph(self):

"""

Добавление списка издательств в Combo Box.

"""

ph = self.facade.get\_ph()

self.comboBox\_ph.clear()

if self.comboBox\_ph is not None:

self.comboBox\_ph.addItems(ph)

logging.log(logging.INFO, 'Combo Box "Издательства" обновлён')

def add\_new\_request(self):

"""

Оформление нового заказа и его показ в ListWidget.

"""

client\_title = QListWidgetItem("Клиент:")

client = QListWidgetItem(self.comboBox\_clients.currentText())

cl1 = self.comboBox\_clients.currentText()

cl2 = self.facade.get\_id\_client(cl1)

client\_code = QListWidgetItem(cl2)

number\_title = QListWidgetItem("Номер заказа:")

number = str(cl2) + '/' + str(self.date\_req)

book\_title = QListWidgetItem("Книга:")

book = QListWidgetItem(self.comboBox\_book.currentText())

date\_req\_title = QListWidgetItem("Дата создания:")

time\_req\_title = QListWidgetItem("Время заказа:")

date\_now = datetime.datetime.now()

date\_req = str(date\_now.strftime("%Y.%m.%d"))

time\_req = str(date\_now.strftime("%H:%M"))

self.add\_new\_field.clear()

if self.client != '' and self.book != '':

self.add\_new\_field.addItem(number\_title)

self.add\_new\_field.addItem(number)

self.add\_new\_field.addItem(date\_req\_title)

self.add\_new\_field.addItem(date\_req)

self.add\_new\_field.addItem(time\_req\_title)

self.add\_new\_field.addItem(time\_req)

self.add\_new\_field.addItem(client\_title)

self.add\_new\_field.addItem(client\_code)

self.add\_new\_field.addItem(client)

self.add\_new\_field.addItem(book\_title)

self.add\_new\_field.addItem(book)

logging.log(logging.INFO, 'Заказ сформирован в ListWidget')

def add\_book\_to\_request(self):

"""

Добавление книги в заказ.

"""

if self.block\_req is True:

book = QListWidgetItem(self.comboBox\_book.currentText())

self.add\_new\_field.addItem(book)

logging.log(logging.INFO, 'Добавлена книга в заказ (ListWidget)')

else:

self.mes\_box('Заказ уже сформирован')

def save\_request(self):

"""

Сохранение заказа в базу данных.

"""

ignore\_serv = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

ignore2 = [1, 3, 5, 7]

count = self.add\_new\_field.count()

list\_serv = [ind for ind in range(count) if ind not in ignore\_serv]

list\_req = [ind for ind in range(count) if ind in ignore2]

request = []

book = []

for i in list\_serv:

name = self.add\_new\_field.item(i).text()

id\_book = self.facade.get\_book\_id(name)

book.append(id\_book)

for j in list\_req:

if j == 7:

id\_cl = str(self.add\_new\_field.item(j).text())

request.append(id\_cl)

else:

request.append(self.add\_new\_field.item(j).text())

for a in book:

if a != 'o':

self.facade.create\_request(request[0], request[1], request[2], request[3], self.emp, a)

else:

continue

logging.log(logging.INFO, 'Заказ добавлен в базу данных')

def generateCode(self):

"""

Создание штрих-кода по номеру, дате и времени заказа

"""

if self.block\_req != False:

rv = BytesIO()

EAN13 = barcode.get\_barcode\_class('code39')

EAN13(str(100000902922), writer=ImageWriter()).write(rv)

temp = str(self.spin\_num\_order.value()) + str(self.date\_req) + str(self.time\_req)

temp\_middle = temp.replace(".", "")

temp\_end = temp\_middle.replace(":", "")

name = "code" + temp\_end

with open("codes/" + name + '.png', "wb") as f:

EAN13(temp\_end, writer=ImageWriter(), add\_checksum=False).write(f)

a4\_page\_size = [img2pdf.in\_to\_pt(8.3), img2pdf.in\_to\_pt(11.7)]

layout\_function = img2pdf.get\_layout\_fun(a4\_page\_size)

pdf = img2pdf.convert("codes/" + name + '.png', layout\_fun=layout\_function)

with open("codes/" + name + '.pdf', 'wb') as f:

f.write(pdf)

icon = QtGui.QIcon('codes/' + name + '.png')

item = QtWidgets.QListWidgetItem(icon, "")

self.add\_new\_field.addItem(item)

self.block\_req = False

logging.log(logging.INFO, 'Создан штрих-код')

else:

self.mes\_box('Штрих-код уже создан')

def otchyot(self, report):

"""

Выборка данных для создания отчётов.

:return: [count\_order, count\_book, count\_order\_book]

"""

start = self.date\_start.dateTime().toString('yyyy.MM.dd')

end = self.date\_end.dateTime().toString('yyyy.MM.dd')

data = list(self.facade.get\_data\_book())

count\_book = {} # 1

count\_order\_book = {} # 2

count\_order = {} # 3

if report == 1:

for i, date in enumerate(data):

data[i] = list(data[i])

if data[i][0] >= start and data[i][0] <= end:

try:

count\_book[date[0]] += 1

except KeyError:

count\_book[date[0]] = 1

return [count\_book]

elif report == 3:

for i, date in enumerate(data):

data[i] = list(data[i])

if data[i][0] >= start and data[i][0] <= end:

try:

if i == 0:

continue

else:

if data[i-1][2] == data[i][2]:

continue

else:

count\_order[date[0]] += 1

except KeyError:

count\_order[date[0]] = 1

return [count\_order]

elif report == 2:

for i, date in enumerate(data):

data[i] = list(data[i])

if data[i][0] >= start and data[i][0] <= end:

try:

count\_order\_book[date[0]]

except KeyError:

count\_order\_book[date[0]] = {}

if date[1]:

try:

count\_order\_book[date[0]][date[1]] += 1

except KeyError:

count\_order\_book[date[0]][date[1]] = 1

return [count\_order\_book]

logging.log(logging.INFO, 'Получены данные для отчёта')

def count\_order(self):

"""

Создание отчёта "Количество заказов по дням за период времени".

"""

self.table\_graf.setRowCount(0)

self.table\_graf.setColumnCount(0)

temp = self.otchyot(3)

rec = temp[0]

self.dict = temp[0]

self.table\_graf.setColumnCount(2)

self.table\_graf.setRowCount(len(rec))

self.table\_graf.setHorizontalHeaderLabels(['Дата', 'Количество заказов'])

data = []

count = []

for d, c in rec.items():

data.append(d)

count.append(c)

for row, value1 in enumerate(data):

item = QTableWidgetItem()

item.setText(str(value1))

self.table\_graf.setItem(row, 0, item)

for row, value2 in enumerate(count):

item = QTableWidgetItem()

item.setText(str(value2))

self.table\_graf.setItem(row, 1, item)

self.type = 3

logging.log(logging.INFO, 'Создан отчёт "Количество заказов по дням за период времени"')

def count\_book(self):

"""

Создание отчёта "Количество оказанных услуг по дням за период времени".

"""

self.table\_graf.setRowCount(0)

self.table\_graf.setColumnCount(0)

temp = self.otchyot(1)

rec = temp[0]

self.dict = temp[0]

self.table\_graf.setColumnCount(2)

self.table\_graf.setRowCount(len(rec))

self.table\_graf.setHorizontalHeaderLabels(['Дата', 'Количество книг'])

data = []

count = []

for d, c in rec.items():

data.append(d)

count.append(c)

for row, value1 in enumerate(data):

item = QTableWidgetItem()

item.setText(str(value1))

self.table\_graf.setItem(row, 0, item)

for row, value2 in enumerate(count):

item = QTableWidgetItem()

item.setText(str(value2))

self.table\_graf.setItem(row, 1, item)

self.type = 1

logging.log(logging.INFO, 'Создан отчёт "Количество оказанных услуг по дням за период времени"')

def count\_order\_book(self):

"""

Создание отчёта "Количество заказов по дням за период времени по каждой услуге".

"""

self.table\_graf.setRowCount(0)

self.table\_graf.setColumnCount(0)

temp = self.otchyot(2)

rec = temp[0]

self.dict = temp[0]

day = []

book = []

count = []

lenght = 0

for d in rec.items():

for i in d:

if i == d[1]:

temp = []

temp2 = []

for j in i:

temp.append(j)

book.append(temp)

for k in i:

temp2.append(i[k])

count.append(temp2)

else:

day.append(i)

for i in book:

lenght += len(i)

x = 0

x2 = 0

self.table\_graf.setRowCount(lenght)

self.table\_graf.setColumnCount(3)

self.table\_graf.setHorizontalHeaderLabels(['Дата', 'Книга', 'Количество книг'])

for row, value1 in enumerate(day):

temp\_book = book[row]

temp\_count = count[row]

len\_book = len(temp\_book)

self.table\_graf.setSpan(x, 0, len\_book, 1)

item = QTableWidgetItem(str(value1))

self.table\_graf.setItem(x, 0, item)

for i in temp\_book:

name\_book = self.facade.get\_book\_name(i)

item\_book = QTableWidgetItem(name\_book[1:-1])

self.table\_graf.setItem(x, 1, item\_book)

x += 1

for j in temp\_count:

item\_count = QTableWidgetItem(str(j))

self.table\_graf.setItem(x2, 2, item\_count)

x2 += 1

self.type = 2

logging.log(logging.INFO, 'Создан отчёт "Количество заказов по дням за период времени по каждой услуге"')

def order\_pdf(self, type, data):

"""

Создание отчёта в pdf-формате.

:param type: вид отчёта

:param data: данные

"""

from fpdf import FPDF

x, y = 10, 60

pdf = FPDF()

pdf.add\_page()

pdf.add\_font('Calibri Regular', '', 'C:\Windows\Fonts\calibri.ttf', uni=True)

pdf.set\_font('Calibri Regular', size=25)

pdf.image('img/logo\_mini.png', w=20, h=20)

if type == 1:

pdf.cell(200, -20, txt="Отчёт по кол-ву проданных книг по дням", ln=1, align="C")

pdf.set\_font('Calibri Regular', size=14)

pdf.text(x, y, 'Дата')

pdf.text(x + 60, y, 'Кол-во проданных книг')

for d in data:

y += 10

if y == 300:

y, x = 10, 10

pdf.add\_page()

pdf.text(x, y, 'Дата')

pdf.text(x + 60, y, 'Кол-во проданных книг')

y += 10

pdf.text(x, y, d)

pdf.text(x + 60, y, str(data[d]))

pdf.output("pdf/report\_book.pdf")

logging.log(logging.INFO, 'Сгенерирован отчёт "Количество проданных книг по дням за период времени" в pdf-формате')

elif type == 2:

pdf.cell(200, -20, txt="Отчёт по кол-ву заказов по каждой книге", ln=1, align="C")

pdf.set\_font('Calibri Regular', style=b, size=14)

pdf.text(x, y, 'Дата')

pdf.text(x + 60, y, 'Книга')

pdf.text(x + 120, y, 'Кол-во проданных книг')

for d in data:

pdf.text(x, y + 10, d)

for book in data[d]:

y += 10

if y == 300:

y, x = 10, 10

pdf.add\_page()

pdf.set\_font()

pdf.text(x, y, 'Дата')

pdf.text(x + 60, y, 'Книга')

pdf.text(x + 120, y, 'Кол-во проданных книг')

y += 10

book\_name = self.facade.get\_book\_name(book)

pdf.text(x + 60, y, str(book\_name[1:-1]))

pdf.text(x + 123, y, str(data[d][book]))

pdf.output("pdf/report\_order\_book.pdf")

logging.log(logging.INFO, 'Сгенерирован отчёт "Количество заказов по дням за период времени по каждой книге" в pdf-формате')

elif type == 3:

pdf.cell(200, -20, txt="Отчёт по кол-ву заказов по дням", ln=1, align="C")

pdf.set\_font('Calibri Regular', size=14)

pdf.text(x, y, 'Дата')

pdf.text(x + 60, y, 'Кол-во заказов')

for d in data:

y += 10

if y == 300:

y, x = 10, 10

pdf.add\_page()

pdf.text(x, y, 'Дата')

pdf.text(x + 60, y, 'Кол-во заказов')

y += 10

pdf.text(x, y, d)

pdf.text(x + 60, y, str(data[d]))

pdf.output("pdf/report\_order.pdf")

logging.log(logging.INFO, 'Сгенерирован отчёт "Количество заказов по дням за период времени" в pdf-формате')

def next\_page(self):

"""

Переход к следующей странице StackedWidget.

"""

if self.now\_page != len(self.page\_id)-1:

self.now\_page += 1

self.page.setCurrentIndex(self.page\_id[self.now\_page])

logging.log(logging.INFO, 'Переход на следующую страницу')

def back\_page(self):

"""

Переход к предыдущей странице StackedWidget.

"""

if self.now\_page != 0:

self.now\_page -= 1

self.page.setCurrentIndex(self.page\_id[self.now\_page])

logging.log(logging.INFO, 'Переход на предыдущую страницу')

def open\_auth(self):

"""

Отображение диалогового окна "Авторизация".

"""

dialog = DialogAuth(self)

dialog.setWindowTitle("Авторизация")

dialog.show()

dialog.exec\_()

logging.log(logging.INFO, 'Открыто окно авторизации')

def open\_new\_client(self):

"""

Отображение диалогового окна "Добавление нового клиента".

"""

dialog\_client = DialogNewClient(self)

dialog\_client.setWindowTitle("Добавление нового клиента")

dialog\_client.show()

dialog\_client.exec\_()

logging.log(logging.INFO, 'Открыто окно добавления нового клиента')

def mes\_box(self, text):

"""

Создание messagebox с переданным текстом.

:param text: текст для вывода в messagebox

"""

messagebox = QMessageBox(self)

messagebox.setWindowTitle("Штрих-код")

messagebox.setText(text)

messagebox.setStandardButtons(QMessageBox.Ok)

messagebox.show()

logging.log(logging.INFO, 'Открыто окно MessageBox')

class DialogAuth(QDialog):

def \_\_init\_\_(self, parent=None):

"""

Подключением к кнопкам, объявление переменных, создание сцены для GraphicsView.

"""

super(DialogAuth, self).\_\_init\_\_(parent)

self.ui = uic.loadUi("auth.ui", self)

self.facade = Facade()

self.scene = QGraphicsScene(0, 0, 300, 80)

self.ui.draw\_captcha.setScene(self.scene)

self.ui.btn\_enter.clicked.connect(self.enter)

self.ui.btn\_new\_captcha.clicked.connect(self.captcha\_generation)

self.ui.btn\_hide\_password.clicked.connect(self.vis\_pas)

self.visible\_captcha(False)

self.count\_try\_entry = 0

self.now\_captcha = None

self.next\_try = 0

self.vis\_p = False

logging.log(logging.INFO, 'Запущено окно авторизации')

def vis\_pas(self):

"""

Скрывает и показывает пароль.

"""

ed = self.ui.edit\_password

if self.vis\_p:

self.vis\_p = False

ed.setEchoMode(QtWidgets.QLineEdit.Password)

logging.log(logging.INFO, 'Пароль скрыт')

else:

self.vis\_p = True

ed.setEchoMode(QtWidgets.QLineEdit.Normal)

logging.log(logging.INFO, 'Пароль показан')

def visible\_captcha(self, visible=True):

"""

Вызывается в \_\_init\_\_ (с параметром False) и при второй неуспешной попытки входа

(неправильный ввод пароля или логина) с параметом True.

:param visible: отображение поля

При False скрывает поле ввода, кнопку обновления и сцену для отрисовки капчи.

При True - показывает поле ввода, кнопку обновления и сцену для отрисовки капчи.

"""

self.ui.draw\_captcha.setVisible(visible)

self.ui.edit\_captcha.setVisible(visible)

self.ui.label\_4.setVisible(visible)

self.ui.btn\_new\_captcha.setVisible(visible)

logging.log(logging.INFO, 'Показано поле с капчей')

def captcha\_generation(self):

"""

Вызывается при второй неуспешной попытке входа и при нажатии на кнопку «btn\_new\_captcha».

Выводит капчу в «graphicsView» и возвращает значение капчи в переменной self.now\_captcha.

"""

self.scene.clear()

syms = 'qwertyuiopasdfghjklzxcvbnm1234567890'

count\_syms = 3

now\_syms = ['']\*count\_syms

x, y = 30, 20

self.scene.addLine(0, random.randint(20, 45), 200, random.randint(30, 60))

for i in range(count\_syms):

now\_syms[i] = syms[random.randint(0, 35)]

x+=20

text = self.scene.addText(f"{now\_syms[i]}")

text.setFont(QFont("MS Shell Dlg 2", 15))

text.moveBy(x, y+random.randint(-10, 20))

self.now\_captcha = ''.join(now\_syms)

logging.log(logging.INFO, 'Капча сгенерирована')

def mes\_box(self, text):

"""

Создание messagebox с переданным текстом.

:param text: текст для вывода в messagebox

"""

messagebox = QMessageBox(self)

messagebox.setWindowTitle("Ошибка")

messagebox.setText(text)

messagebox.setStandardButtons(QMessageBox.Ok)

messagebox.show()

logging.log(logging.INFO, 'Открыто окно MessageBox')

def enter(self):

"""

Вызывается при нажатии на кнопку btn\_enter.

Обрабатывает все случаи ввода данных (капчи, логина, пароля) и считает неуспешные попытки входа.

Проверяет есть ли у пользователя блокировка и до скольки она длиться.

При успешном входе передает в фасад время и логин успешного входа (для записи в бд),

записывает индексы доступных страничек «Stacked Widget»

(у разных сотрудников могут быть разные странички).

"""

t = time.localtime()

now\_time = time.mktime(t)

auth\_log = self.ui.edit\_login.text()

auth\_pas = self.ui.edit\_password.text()

if auth\_log == '' or auth\_pas == '':

logging.log(logging.INFO, 'Ошибка. Заполните все поля!')

self.mes\_box('Заполните все поля!')

elif self.now\_captcha is not None and self.ui.edit\_captcha.text() == '':

logging.log(logging.INFO, 'Ошибка. Введите капчу!')

self.mes\_box('Введите капчу!')

else:

password, role, last\_exit, block, f, i, o, photo = self.parent().facade.get\_for\_authorization(auth\_log)

fio = f + ' ' + i + ' ' + o

id\_emp = self.parent().facade.get\_id\_emp(f,i,o)

pix = QPixmap(f'img/{photo}')

if last\_exit is not None and block:

last\_exit = last\_exit.split()

day, mon, year = map(int, last\_exit[0].split('.'))

hour, mi, sec = map(int, last\_exit[1].split(':'))

time\_block = time.mktime(

(year, mon, day, hour, mi + 3, sec, 0, 0, 0))

if time\_block > now\_time:

logging.log(logging.INFO, 'Ошибка. Подождите, время нового сеанса еще не пришло.')

self.mes\_box('Подождите, время нового сеанса еще не пришло.')

return

if self.count\_try\_entry >= 3 and self.next\_try > now\_time:

logging.log(logging.INFO, 'Ошибка. Подождите, прежде чем пытаться вводить снова.')

self.mes\_box('Подождите, прежде чем пытаться вводить снова.')

return

if self.now\_captcha is not None and self.now\_captcha != self.ui.edit\_captcha.text():

logging.log(logging.INFO, 'Ошибка. Неправильно введена капча.')

self.mes\_box('Неправильно введена капча.')

elif password != auth\_pas:

self.count\_try\_entry += 1

if self.count\_try\_entry >= 3:

self.next\_try = now\_time + 10

if password != '':

time\_entry = time.strftime("%d.%m.%Y %H:%M:%S", t)

self.parent().facade.insert\_time\_entry(id\_emp, time\_entry, 'Неуспешно')

if self.count\_try\_entry == 2:

self.visible\_captcha(True)

self.captcha\_generation()

logging.log(logging.INFO, 'Ошибка. Вторая неуспешная попытка входа. Теперь введите капчу.')

self.mes\_box('Вторая неуспешная попытка входа. Теперь введите капчу.')

else:

logging.log(logging.INFO, 'Ошибка. Неправильно введены данные.')

self.mes\_box('Неправильно введены данные.')

elif password == auth\_pas:

time\_entry = time.strftime("%d.%m.%Y %H:%M:%S", t)

self.parent().facade.insert\_time\_entry(id\_emp, time\_entry, 'Успешно')

logging.log(logging.INFO, 'Вход выполнен')

self.parent().ui.lbl\_role.setText(role)

if role == 'Старший смены' or role == 'Продавец':

self.parent().hide()

self.parent().page\_id = [0, 2]

logging.log(logging.INFO, 'Авторизован пользователь "Страший смены" или "Продавец"')

else:

self.parent().hide()

self.parent().page\_id = [0, 1, 4, 5, 6, 7, 8]

logging.log(logging.INFO, 'Авторизован пользователь "Администратор"')

self.parent().show()

self.parent().ui.lbl\_photo.setPixmap(pix)

self.parent().now\_login = auth\_log

self.parent().emp = id\_emp

self.close()

logging.log(logging.INFO, 'Закрыто окно авторизации')

class DialogNewClient(QDialog):

def \_\_init\_\_(self, parent=None):

"""

Подключение к кнопке "Добавить"

"""

super(DialogNewClient, self).\_\_init\_\_(parent)

self.ui = uic.loadUi("new\_client.ui", self)

self.facade = Facade()

self.ui.btn\_add\_client.clicked.connect(self.add)

logging.log(logging.INFO, 'Открыто окно добавления нового пользователя')

def add(self):

"""

Добавление клиента в базу данных.

"""

fio = []

self.surname = self.ui.edit\_surname.text()

self.name = self.ui.edit\_name.text()

self.lastname = self.ui.edit\_lastname.text()

self.email = self.ui.edit\_email.text()

self.dateOfBirth = self.ui.date\_birth.dateTime().toString('dd.MM.yyyy')

fio.append(self.surname)

fio.append(self.name)

fio.append(self.lastname)

if self.surname != '' and self.name != '' and self.dateOfBirth != '' and self.email != '':

self.facade.insert\_client(fio, self.dateOfBirth, self.email)

logging.log(logging.INFO, 'Клиент добавлен в базу данных')

else:

self.mes\_box('Заполните все поля!')

logging.log(logging.INFO, 'Не все поля заполнены')

def mes\_box(self, text):

"""

Создание messagebox с переданным текстом.

:param text: текст для вывода в messagebox

"""

messagebox = QMessageBox(self)

messagebox.setWindowTitle("Ошибка")

messagebox.setText(text)

messagebox.setStandardButtons(QMessageBox.Ok)

messagebox.show()

logging.log(logging.INFO, 'Открыто окно MessageBox')

class Builder:

"""

Паттерн строитель.

"""

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

self.qapp = QApplication(sys.argv)

self.window = MainWindow()

self.auth()

def auth(self):

self.window.open\_auth()

self.qapp.exec()

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

B = Builder()

database.py

import mysql

from mysql.connector import connect

class Database:

"""

Класс с функциями для взаимодействия с базой данных

"""

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

"""

Подключение к базе данных MySQL

"""

self.conn = mysql.connector.connect(host='localhost', port=3306, user='root', password="iejahjoU1",

database='bookshop')

# INSERT

def insert\_client(self, fio, dateOfBirth, email):

"""

Добавление клиента

:param fio: ФИО клиента

:param dateOfBirth: дата рождения клиента

:param email: электронная почта

"""

cursor = self.conn.cursor()

if len(fio) == 2:

cursor.execute("INSERT INTO clients VALUES (NULL, %s, %s, NULL, %s, %s)",

(fio[0], fio[1], dateOfBirth, email))

else:

cursor.execute("INSERT INTO clients VALUES (NULL, %s, %s, %s, %s, %s)",

(fio[0], fio[1], fio[2], dateOfBirth, email))

self.conn.commit()

def insert\_genre(self, genre):

"""

Добавление жанра

:param genre: наименование жанра

"""

cursor = self.conn.cursor()

cursor.execute("INSERT INTO genre VALUES (NULL, %s)", (genre,))

self.conn.commit()

def insert\_author(self, fio, date):

"""

Добавление автора

:param fio: ФИО автора

:param date: дата рождения автора

"""

cursor = self.conn.cursor()

if len(fio) == 2:

cursor.execute("INSERT INTO author VALUES (NULL, %s, %s, %s, %s)", (fio[0], fio[1], '', date))

else:

cursor.execute("INSERT INTO author VALUES (NULL, %s, %s, %s, %s)", (fio[0], fio[1], fio[2], date))

self.conn.commit()

def insert\_ph(self, name):

"""

Добавление издательства

:param name: название издательства

"""

cursor = self.conn.cursor()

cursor.execute("INSERT INTO publishing\_house VALUES (NULL, %s)", (name,))

self.conn.commit()

def insert\_book(self, name, cost, genre, author, ph):

"""

Добавление книги

:param name: название книги

:param cost: стоимость книги

:param genre: код жанра

:param author: код автора

:param ph: код издательства

"""

cursor = self.conn.cursor()

cursor.execute("INSERT INTO books VALUES (NULL, %s, %s, %s, %s, %s)", (name, cost, genre, author, ph))

self.conn.commit()

def insert\_request(self, number, date, time, client, employee, book):

"""

Добавление заказа

:param number: номер заказа

:param date: дата создания

:param time: время создания

:param client: код клиента

:param employee: код сотрудника

:param book: код книги

"""

cursor = self.conn.cursor()

cursor.execute("INSERT INTO requests VALUES (NULL, %s, %s, %s, %s, %s, %s)",

(number, date, time, client, book, employee))

self.conn.commit()

def insert\_time\_entry(self, login, time, success):

"""

Добавление времени входа сотрудника

:param login: логин сотрудника

:param time: дата и время

:param success: успешная/ошибочная попытка входа

"""

cursor = self.conn.cursor()

cursor.execute(f"INSERT INTO history VALUES (NULL, %s, NULL, %s, %s)", (time, success, login))

cursor.execute(f"UPDATE employees set `Последний вход`='{time}', `Тип входа`='{success}' WHERE `Код сотрудника`='{login}'")

self.conn.commit()

def insert\_time\_exit(self, login, time, block):

"""

Добавление времени выхода сотрудника

:param login: логин

:param time: дата и время

:param block: блокировка входа (при необходимости)

"""

cursor = self.conn.cursor()

cursor.execute(f"INSERT INTO history VALUES (NULL, NULL, %s, %s, %s)", (time, block, login))

self.conn.commit()

# UPDATE

def update\_genre(self, id, genre):

"""

Обновление жанра

:param id: код жанра

:param genre: наименование жанра

"""

cursor = self.conn.cursor()

cursor.execute(f"UPDATE genre set `Жанр`='{genre}' WHERE `Код жанра`='{id}'")

self.conn.commit()

def update\_author(self, id, fio, date):

"""

Обновление автора

:param id: код автора

:param fio: ФИО автора

:param date: дата рождения автора

"""

cursor = self.conn.cursor()

if len(fio) == 2:

cursor.execute(

f"UPDATE author set `Фамилия`='{fio[0]}', `Имя`='{fio[1]}', `Отчество`='', `Дата рождения`='{date}' "

f"WHERE `Код автора`='{id}'")

else:

cursor.execute(

f"UPDATE author set `Фамилия`='{fio[0]}', `Имя`='{fio[1]}', `Отчество`='{fio[2]}', "

f"`Дата рождения`='{date}' WHERE `Код автора`='{id}'")

self.conn.commit()

def update\_ph(self, id, ph):

"""

Обновление издательства

:param id: код издательства

:param ph: название издательства

"""

cursor = self.conn.cursor()

cursor.execute(f"UPDATE publishing\_house set `Название издательства`='{ph}' WHERE 'Код издательсвта'='{id}'")

self.conn.commit()

def update\_book(self, id, name, cost, genre, author, ph):

"""

Обновление книги

:param id: код книги

:param name: название книги

:param cost: стоимость книги

:param genre: код жанра

:param author: код автора

:param ph: код издательства

"""

cursor = self.conn.cursor()

cursor.execute(

f"UPDATE books set `Наименование книги`='{name}', `Стоимость`='{cost}', `Жанр`='{genre}', "

f"`Автор`='{author}', `Издательство`='{ph}' WHERE 'Код книги'='{id}'")

self.conn.commit()

# DELETE

def delete\_client(self, id):

"""

Удаление клиента

:param id: код клиента

"""

cursor = self.conn.cursor()

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=0")

cursor.execute(f"DELETE FROM clients WHERE `Код клиента`='{id}'")

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=1")

self.conn.commit()

def delete\_genre(self, id):

"""

Удаление жанра

:param id: код жанра

"""

cursor = self.conn.cursor()

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=0")

cursor.execute(f"DELETE FROM genre WHERE `Код жанра`='{id}'")

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=1")

self.conn.commit()

def delete\_author(self, id):

"""

Удаление автора

:param id: код автора

"""

cursor = self.conn.cursor()

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=0")

cursor.execute(f"DELETE FROM author WHERE `Код автора`='{id}'")

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=1")

self.conn.commit()

def delete\_ph(self, id):

"""

Удаление издательства

:param id: код издательства

"""

cursor = self.conn.cursor()

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=0")

cursor.execute(f"DELETE FROM publishing\_house WHERE `Код издательства`='{id}'")

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=1")

self.conn.commit()

def delete\_book(self, id):

"""

Удаление книги

:param id: код книги

"""

cursor = self.conn.cursor()

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=0")

cursor.execute(f"DELETE FROM books WHERE `Код книги`='{id}'")

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=1")

self.conn.commit()

# SELECT

def select\_clients(self):

"""

Получение списка клиентов

:return: clients

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT \* FROM clients")

clients = cursor.fetchall()

return clients

def select\_genre(self):

"""

Получение списка жанров

:return: genres

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT \* FROM genre")

genres = cursor.fetchall()

return genres

def select\_author(self):

"""

Получение списка авторов

:return: authors

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT \* FROM author")

authors = cursor.fetchall()

return authors

def select\_ph(self):

"""

Получение списка издательств

:return: ph

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT \* FROM publishing\_house")

ph = cursor.fetchall()

return ph

def select\_books(self):

"""

Получение списка книг

:return: books

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT \* FROM books")

books = cursor.fetchall()

return books

def select\_history(self):

"""

Получение истории входа сотрудников

:return: history

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT \* FROM history")

history = cursor.fetchall()

return history

def get\_logins(self):

"""

Получение списка логинов сотрудников

:return: logins

"""

logins = []

cursor = self.conn.cursor()

cursor.execute(f"""SELECT Логин FROM employees""")

rows = cursor.fetchall()

for i in rows:

for j in i:

logins.append(j)

return logins

def get\_info(self, login):

"""

Получение информации о сотруднике

:param login: логин сотрудника

:return: info

"""

info = []

cursor = self.conn.cursor()

cursor.execute(f"SELECT Пароль, Должность, `Последний вход`, `Тип входа`, Фамилия, Имя, Отчество, Фото FROM employees WHERE Логин = '{login}'")

rows = cursor.fetchall()

for i in rows:

for j in i:

info.append(j)

return info

def get\_clients(self):

"""

Получение списка ФИО клиентов

:return: clients

"""

clients = []

cursor = self.conn.cursor()

cursor.execute(f"SELECT Фамилия, Имя, Отчество FROM clients")

rows = cursor.fetchall()

for i in rows:

if i[2] is None:

clients.append(str(i[0] + ' ' + i[1]))

else:

clients.append(str(i[0] + ' ' + i[1] + ' '))

return clients

def get\_genres(self):

"""

Получение наименований жанров

:return: books

"""

genres = []

cursor = self.conn.cursor()

cursor.execute(f"SELECT `Жанр` FROM genre")

rows = cursor.fetchall()

for i in rows:

genres.append(str(i)[2:-3])

return genres

def get\_authors(self):

"""

Получение ФИО авторов

:return: authors

"""

authors = []

cursor = self.conn.cursor()

cursor.execute(f"SELECT Фамилия, Имя, Отчество FROM author")

rows = cursor.fetchall()

for i in rows:

if i[2] is None:

authors.append(str(i[0] + ' ' + i[1]))

else:

authors.append(str(i[0] + ' ' + i[1] + ' ' + i[2]))

return authors

def get\_ph(self):

"""

Получение названий издательств

:return: phs

"""

phs = []

cursor = self.conn.cursor()

cursor.execute(f"SELECT `Название издательства` FROM publishing\_house")

rows = cursor.fetchall()

for i in rows:

phs.append(str(i)[2:-3])

return phs

def get\_books(self):

"""

Получение названий книг

:return: books

"""

books = []

cursor = self.conn.cursor()

cursor.execute(f"SELECT `Наименование книги` FROM books")

rows = cursor.fetchall()

for i in rows:

books.append(str(i)[2:-3])

return books

def get\_client\_id(self, fio):

"""

Получение кода клиента

:param fio: ФИО клиента

:return: code

"""

cursor = self.conn.cursor()

client = fio.split()

if len(client) == 2:

cursor.execute(

f"""SELECT `Код клиента` FROM clients WHERE `Фамилия`='{client[0]}' and `Имя`='{client[1]}'""")

else:

cursor.execute(

f"""SELECT `Код клиента` FROM clients WHERE `Фамилия`='{client[0]}' and `Имя`='{client[1]}' and

`Отчество`='{client[2]}'""")

code = str(cursor.fetchone())

return code[1:-2]

def get\_emp\_id(self, surname, name, lastname):

"""

Получение кода сотрудника

:param surname: фамилия

:param name: имя

:param lastname: отчество

:return: code

"""

cursor = self.conn.cursor()

cursor.execute(

f"SELECT `Код сотрудника` FROM employees WHERE Фамилия='{surname}' and Имя='{name}' and Отчество='{lastname}'")

code = str(cursor.fetchone())

return code[1:-2]

def get\_genre\_id(self, name):

"""

Получение кода жанра

:param name: жанр

:return: code

"""

cursor = self.conn.cursor(buffered=True)

cursor.execute(f"SELECT `Код жанра` FROM genre WHERE `Жанр`='{name}'")

code = str(cursor.fetchone())

return code[1:-2]

def get\_author\_id(self, fio):

"""

Получение кода автора

:param fio: ФИО автора

:return: code

"""

cursor = self.conn.cursor(buffered=True)

au = fio.split()

if len(au) == 2:

cursor.execute(f"SELECT `Код автора` FROM author WHERE `Фамилия`='{au[0]}' and `Имя`='{au[1]}'")

else:

cursor.execute(

f"SELECT `Код автора` FROM author WHERE `Фамилия`='{au[0]}' and `Имя`='{au[1]}' and `Отчество`='{au[2]}'")

code = str(cursor.fetchone())

return code[1:-2]

def get\_ph\_id(self, name):

"""

Получение кода издательства

:param name: название издательства

:return: code

"""

cursor = self.conn.cursor(buffered=True)

cursor.execute(f"SELECT `Код издательства` FROM publishing\_house WHERE `Название издательства`='{name}'")

code = str(cursor.fetchone())

return code[1:-2]

def get\_book\_id(self, name):

"""

Получение кода книги

:param name: название книги

:return: code

"""

cursor = self.conn.cursor(buffered=True)

cursor.execute(f"SELECT `Код книги` FROM books WHERE `Наименование книги`='{name}'")

code = str(cursor.fetchone())

return code[1:-2]

def get\_data\_book(self):

"""

Получение данных о заказе

:return: data

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT `Дата создания`, `Книга`, `Код заказа` FROM requests")

data = cursor.fetchall()

return data

def get\_book\_name(self, code):

"""

Получение названия книги

:return: name

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT `Наименование книги` FROM books WHERE `Код книги`='{code}'")

name = str(cursor.fetchone())

return name[1:-2]

facade.py

from database import Database

class Facade:

"""

Класс фасада

"""

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

self.db = Database()

# SELECT FOR AUTHORIZATION

def get\_logins(self):

return self.db.get\_logins()

def get\_for\_authorization(self, login):

log = self.db.get\_info(login)

if log == []:

return '', '', '', '', '', '', '', ''

password, role, last\_exit, block, f, i, o, photo = log[0], log[1], log[2], log[3], log[4], log[5], log[6], log[7]

return password, role, last\_exit, block, f, i, o, photo

# INSERT

def create\_request(self, number, date, time, client, emp, book):

self.db.insert\_request(number, date, time, client, emp, book)

def insert\_client(self, fio, dateOfBirth, email):

self.db.insert\_client(fio, dateOfBirth, email)

def insert\_genre(self, genre):

self.db.insert\_genre(genre)

def insert\_author(self, fio, date):

self.db.insert\_author(fio, date)

def insert\_ph(self, name):

self.db.insert\_ph(name)

def insert\_book(self, name, cost, genre, author, ph):

self.db.insert\_book(name, cost, genre, author, ph)

def insert\_time\_entry(self, login, time, success):

self.db.insert\_time\_entry(login, time, success)

def insert\_time\_exit(self, login, time, block):

self.db.insert\_time\_exit(login, time, block)

# UPDATE

def update\_genre(self, id, genre):

self.db.update\_genre(id, genre)

def update\_author(self, id, fio, date):

self.db.update\_author(id, fio, date)

def update\_ph(self, id, ph):

self.db.update\_ph(id, ph)

def update\_book(self, id, name, cost, genre, author, ph):

self.db.update\_book(id, name, cost, genre, author, ph)

# DELETE

def delete\_genre(self, id):

self.db.delete\_genre(id)

def delete\_author(self, id):

self.db.delete\_author(id)

def delete\_ph(self, id):

self.db.delete\_ph(id)

def delete\_book(self, id):

self.db.delete\_book(id)

# SELECT ALL

def read\_clients(self):

return self.db.select\_clients()

def read\_genre(self):

return self.db.select\_genre()

def read\_author(self):

return self.db.select\_author()

def read\_ph(self):

return self.db.select\_ph()

def read\_books(self):

return self.db.select\_books()

def read\_history(self):

return self.db.select\_history()

def get\_data\_book(self):

return self.db.get\_data\_book()

# SELECT NAME

def get\_clients(self):

return self.db.get\_clients()

def get\_genres(self):

return self.db.get\_genres()

def get\_authors(self):

return self.db.get\_authors()

def get\_ph(self):

return self.db.get\_ph()

def get\_books(self):

return self.db.get\_books()

def get\_book\_name(self, id):

return self.db.get\_book\_name(id)

# SELECT ID

def get\_id\_client(self, cl):

return self.db.get\_client\_id(cl)

def get\_id\_emp(self, surname, name, lastname):

return self.db.get\_emp\_id(surname, name, lastname)

def get\_genre\_id(self, name):

return self.db.get\_genre\_id(name)

def get\_author\_id(self, au):

return self.db.get\_author\_id(au)

def get\_ph\_id(self, name):

return self.db.get\_ph\_id(name)

def get\_book\_id(self, name):

return self.db.get\_book\_id(name)

# **ПРИЛОЖЕНИЕ Д Исходный код тестов**

import sys

from unittest import TestCase

from PyQt5 import QtCore

from PyQt5.QtCore import QDate, QItemSelectionModel

from PyQt5.QtTest import QTest

from PyQt5.QtWidgets import QApplication

from database import Database

from facade import Facade

from gui import MainWindow, DialogNewClient

class Test1Push(TestCase):

def setUp(self):

self.qapp = QApplication(sys.argv)

self.facade = Facade()

self.db = Database()

self.window = MainWindow()

self.emp = DialogNewClient()

def test\_push\_client(self):

btn\_add = self.emp.ui.btn\_add\_client

self.emp.ui.edit\_surname.setText("test\_test")

self.emp.ui.edit\_name.setText("test\_test")

self.emp.ui.edit\_lastname.setText("test\_test")

self.emp.ui.date\_birth.setDate(QDate.fromString("2022-01-01"))

self.emp.ui.edit\_email.setText("test\_test")

QTest.mouseClick(btn\_add, QtCore.Qt.MouseButton.LeftButton)

def test\_push\_genre(self):

btn\_add = self.window.ui.btn\_new\_genre

self.window.ui.edit\_genre.setText("test\_genre")

QTest.mouseClick(btn\_add, QtCore.Qt.MouseButton.LeftButton)

def test\_push\_author(self):

btn\_add = self.window.ui.btn\_new\_author

self.window.ui.edit\_f.setText("test")

self.window.ui.edit\_i.setText("test\_test")

self.window.ui.edit\_o.setText("test\_test\_test")

self.window.ui.date\_author.setDate(QDate.fromString("2022-01-01"))

QTest.mouseClick(btn\_add, QtCore.Qt.MouseButton.LeftButton)

def test\_push\_ph(self):

btn\_add = self.window.ui.btn\_new\_ph

self.window.ui.edit\_ph.setText("test\_ph")

QTest.mouseClick(btn\_add, QtCore.Qt.MouseButton.LeftButton)

def test\_push\_book(self):

btn\_add = self.window.ui.btn\_new\_book

self.window.ui.edit\_title\_book.setText("test\_book")

self.window.ui.spin\_cost.setValue(700)

QTest.mouseClick(btn\_add, QtCore.Qt.MouseButton.LeftButton)

class Test2DeleteAndSave(TestCase):

def setUp(self):

self.qapp = QApplication(sys.argv)

self.facade = Facade()

self.db = Database()

self.window = MainWindow()

def test\_delete\_genre(self):

rowcount = self.window.table\_genre.rowCount()

self.window.table\_genre.setCurrentCell(rowcount-1, 1, QItemSelectionModel.SelectionFlag.Select)

btn\_del = self.window.ui.btn\_delete\_genre

QTest.mouseClick(btn\_del, QtCore.Qt.MouseButton.LeftButton)

QTest.mouseClick(self.window.ui.btn\_save\_genre, QtCore.Qt.MouseButton.LeftButton)

def test\_delete\_author(self):

rowcount = self.window.table\_author.rowCount()

self.window.table\_author.setCurrentCell(rowcount-1, 1, QItemSelectionModel.SelectionFlag.Select)

btn\_del = self.window.ui.btn\_delete\_author

QTest.mouseClick(btn\_del, QtCore.Qt.MouseButton.LeftButton)

QTest.mouseClick(self.window.ui.btn\_save\_author, QtCore.Qt.MouseButton.LeftButton)

def test\_delete\_ph(self):

rowcount = self.window.table\_ph.rowCount()

self.window.table\_ph.setCurrentCell(rowcount-1, 1, QItemSelectionModel.SelectionFlag.Select)

btn\_del = self.window.ui.btn\_delete\_ph

QTest.mouseClick(btn\_del, QtCore.Qt.MouseButton.LeftButton)

QTest.mouseClick(self.window.ui.btn\_save\_ph, QtCore.Qt.MouseButton.LeftButton)

def test\_delete\_book(self):

rowcount = self.window.table\_book.rowCount()

self.window.table\_book.setCurrentCell(rowcount-1, 1, QItemSelectionModel.SelectionFlag.Select)

btn\_del = self.window.ui.btn\_delete\_book

QTest.mouseClick(btn\_del, QtCore.Qt.MouseButton.LeftButton)

QTest.mouseClick(self.window.ui.btn\_save\_book, QtCore.Qt.MouseButton.LeftButton)

# **ПРИЛОЖЕНИЕ Е Скрипт для создания инсталлятора приложения**

; Script generated by the Inno Setup Script Wizard.

; SEE THE DOCUMENTATION FOR DETAILS ON CREATING INNO SETUP SCRIPT FILES!

#define MyAppName "BookShop"

#define MyAppVersion "1.0"

#define MyAppPublisher "Makarova Darya"

#define MyAppURL "https://github.com/ssmlnsk/CourseWork"

#define MyAppExeName "gui.exe"

[Setup]

; NOTE: The value of AppId uniquely identifies this application. Do not use the same AppId value in installers for other applications.

; (To generate a new GUID, click Tools | Generate GUID inside the IDE.)

AppId={{C96331EA-D79C-46CB-9EEA-6F70436AFC63}

AppName={#MyAppName}

AppVersion={#MyAppVersion}

;AppVerName={#MyAppName} {#MyAppVersion}

AppPublisher={#MyAppPublisher}

AppPublisherURL={#MyAppURL}

AppSupportURL={#MyAppURL}

AppUpdatesURL={#MyAppURL}

DefaultDirName={autopf}\{#MyAppName}

DefaultGroupName={#MyAppName}

LicenseFile=C:\Users\ssmlnsk\Desktop\Курсовая работа\BookShop.rtf

; Uncomment the following line to run in non administrative install mode (install for current user only.)

;PrivilegesRequired=lowest

OutputBaseFilename=BookShop

SetupIconFile=C:\Users\ssmlnsk\Desktop\Курсовая работа\img\logo.ico

Compression=lzma

SolidCompression=yes

WizardStyle=modern

[Languages]

Name: "russian"; MessagesFile: "compiler:Languages\Russian.isl"

[Tasks]

Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked

[Files]

Source: "C:\Users\ssmlnsk\Desktop\Курсовая работа\dist\gui\{#MyAppExeName}"; DestDir: "{app}"; Flags: ignoreversion

Source: "C:\Users\ssmlnsk\Desktop\Курсовая работа\auth.ui"; DestDir: "{app}"; Flags: ignoreversion

Source: "C:\Users\ssmlnsk\Desktop\Курсовая работа\main.ui"; DestDir: "{app}"; Flags: ignoreversion

Source: "C:\Users\ssmlnsk\Desktop\Курсовая работа\new\_client.ui"; DestDir: "{app}"; Flags: ignoreversion

Source: "C:\Users\ssmlnsk\Desktop\Курсовая работа\bookshop.sql"; DestDir: "{app}"; Flags: ignoreversion

Source: "C:\Users\ssmlnsk\Desktop\Курсовая работа\test\_gui.py"; DestDir: "{app}"; Flags: ignoreversion

Source: "C:\Users\ssmlnsk\Desktop\Курсовая работа\dist\img\\*"; DestDir: "{app}"; Flags: ignoreversion recursesubdirs createallsubdirs

Source: "C:\Users\ssmlnsk\Desktop\Курсовая работа\dist\codes"; DestDir: "{app}"; Flags: ignoreversion recursesubdirs createallsubdirs

; NOTE: Don't use "Flags: ignoreversion" on any shared system files

[Icons]

Name: "{group}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"

Name: "{autodesktop}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"; Tasks: desktopicon

[Run]

Filename: "{app}\{#MyAppExeName}"; Description: "{cm:LaunchProgram,{#StringChange(MyAppName, '&', '&&')}}"; Flags: nowait postinstall skipifsilent