****

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

**Филиал «РКТ» МАИ в г. Химки Московской области**

**Специальность**а09.02.07 Информационные системы и программирование

**Группа**  ИСП41-19

**Квалификация** Программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К**

**ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ**

**На тему:** Разработка приложения книжного магазина «Дом книги»

**Автор квалификационной работы**  Макарова Д.А. / /

(Фамилия Имя Отчество) (подпись)

**Руководитель**  Медведев А.В. / /

(Фамилия Имя Отчество) (подпись)

Консультанты:

а) Чевела Ольга Николаевна / /

(Фамилия Имя Отчество) (подпись)

б) / /

(Фамилия Имя Отчество) (подпись)

**К защите допустить**

Зам. директора филиала Ворникова И.А. / /

(Фамилия Имя Отчество) (подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

****

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

**Филиал «РКТ» МАИ в г. Химки Московской области**

**Специальность**а09.02.07 Информационные системы и программирование

**Группа**  ИСП41-19

**Квалификация** Программист

**УТВЕРЖДАЮ**

**Зам. директора филиала**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/И.А.Ворникова /**

**«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023г.**

**РАССМОТРЕНО**

**На заседании ПЦК**

**Председатель ПЦК**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Т.А. Жилина /**

**«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023г.**

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

**Студенту**  Макаровой Дарье Андреевне

(Фамилия Имя Отчество)

**Руководитель**  Медведев Алексей Владимирович

(Фамилия Имя Отчество, ученая степень, ученое звание, должность и место работы)

**1. Наименование темы:**  Разработка приложения книжного магазина «Дом книги»

**2. Срок сдачи студентом законченной работы**а22 июня 2023г.

**3. Техническое задание и исходные данные к работе \_**Разработать приложение для книжного магазина “Дом книги” \_\_\_\_\_

Программный продукт предназначен для обработки заказов, учёта книг на складе и управления закупками. \_\_\_\_\_  
Данный продукт позволяет создать новый заказ, расширять каталог книг, подавать заявку на поставку книг. \_\_\_\_\_

Инструментальные средства: Python 3.9, PyCharm Community Edition, Qt Designer, MySQL Workbench 8.0 CE, MS Word 2016, MS Excel 2016, Inno Setup Compiler

В экономической части

**4. Перечень подлежащих разработке разделов и этапы выполнения работы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п/п | Наименование раздела или этапа | Трудоёмкость в % от  полной трудоёмкости  ВКР | Срок  выполнения | Примечание |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Общая часть | 30 |  |  |
| 2 | Специальная часть | 40 |  |  |
| 3 | Экономическая часть | 20 |  |  |
| 4 | Мероприятия по технике безопасности | 10 |  |  |

**5. Перечень иллюстративно-графических материалов:**

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Наименование** | **Количество листов** |
| 1 | Схема программы |  |
| 2 | Структурная схема базы данных |  |
| 3 | Схемы алгоритма выполнения программы |  |
| 4 | Расчет себестоимости разработки |  |
| 5 | Презентация |  |

**6. Исходные материалы и пособия**



**7. Консультанты по работе с указанием относящихся к ним разделов работы**

|  |  |  |  |
| --- | --- | --- | --- |
| Раздел | Консультант | Подпись, дата | |
|  |  | Задание выдал | Задание принял |
| Экономическая часть | Чевела О.Н. |  |  |

**8. Дата выдачи задания** 17.03.2023

Руководитель Медведев А.В. / /

(подпись)

Задание принял к исполнению Макарова Д.А. / /

(подпись)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc137411992)

[1 Общая часть 8](#_Toc137411993)

[1.1 Анализ методов решения 8](#_Toc137411994)

[1.2 Анализ инструментальных средств 8](#_Toc137411995)

[1.3 Методы исследования 14](#_Toc137411996)

[1.4 Описание системы управления базами данных 15](#_Toc137411997)

[2 Специальная часть 17](#_Toc137411998)

[2.1 Постановка задачи 17](#_Toc137411999)

[2.2 Схема алгоритма выполнения программы 18](#_Toc137412000)

[2.3 Планирование базы данных и разработка схем реализации базы данных 20](#_Toc137412001)

[2.4 Текст программы 22](#_Toc137412002)

[2.4.1 Создание интерфейса программного продукта 22](#_Toc137412003)

[2.4.2 Разработка модуля базы данных 29](#_Toc137412004)

[2.4.3 Разработка модуля графического интерфейса 34](#_Toc137412005)

[2.4.4 Организация взаимосвязи модулей программы 37](#_Toc137412006)

[2.5 Схемы программы 39](#_Toc137412007)

[2.5.1 Схема структуры программы 39](#_Toc137412008)

[2.5.2 Архитектура программы 40](#_Toc137412009)

[2.6 Инструкция по эксплуатации программы 41](#_Toc137412010)

[2.6.1 Общие сведения 41](#_Toc137412011)

[2.6.1.1 Назначение программы 41](#_Toc137412012)

[2.6.1.2 Начало работы 42](#_Toc137412013)

[2.6.2 Установка и запуск программы 43](#_Toc137412014)

[2.6.3 Работа с основными функциями 45](#_Toc137412015)

[2.7 Тестирование программных модулей 50](#_Toc137412016)

[3 Экономическая часть 54](#_Toc137412017)

[3.1 Расчет себестоимости разработки методики информационной системы книжного магазина 54](#_Toc137412018)

[4 мероприятия по технике безопасности 58](#_Toc137412019)

[ЗАКЛЮЧЕНИЕ 60](#_Toc137412020)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 61](#_Toc137412021)

[ПРИЛОЖЕНИЕ А. Иллюстрации форм 62](#_Toc137412022)

[ПРИЛОЖЕНИЕ Б. Листинг форм в формате .py 69](#_Toc137412023)

[ПРИЛОЖЕНИЕ В. Исходный код программы 114](#_Toc137412024)

[ПРИЛОЖЕНИЕ Г. Исходный код тестов 165](#_Toc137412025)

[ПРИЛОЖЕНИЕ Д. Скрипт для создания инсталлятора приложения 168](#_Toc137412026)

[ПРИЛОЖЕНИЕ Е. Запросы на создание таблиц и добавление в них данных 170](#_Toc137412027)

**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.

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

Разрабатываемым приложением 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.

Разрабатываемым приложением будет являться программное обеспечение, имеющее в своем составе следующие подсистемы:

* подсистема управления заказами;
* подсистема управления закупками книг;
* подсистема учета книг на складе.

Каждая подсистема должна работать четко и слаженно. Для работы каждой из них вести отчетность по срокам выполнения работ. Подсистема управления заказами должна обеспечивать возможность создать новый заказ для покупателя, выбрав желаемую книгу из списка имеющихся, вносить данные о клиенте.

Подсистема управления закупками книг необходима для обеспечения книжного магазина товаром — должно обеспечиваться добавление новых книг, информацией о них — название, жанр, автор, издательство, а также работа с поставщиками — добавление или удаление поставщика, оформление закупки книг в определённом количестве.

Подсистема учета книг на складе должна предоставлять пользователю сводку данных по книгам — сколько книг всего на складе, список книг по жанрам, по авторам, по издательствам, а также отображает те книги, количество которых подходит к концу, для того чтобы восполнить запасы на складе.

Главная цель выпускной квалификационной работы — разработка системы управления для книжного магазина.

В процессе разработки системы необходимо учесть и выполнить следующие задачи:

* разработать базу данных, содержащую в себе все сведения необходимые для работы системы;
* реализовать модули управления заказами, управления закупками книг, учета книг на складе;
* сформировать отчеты по деятельности каждого модуля;
* провести проверку корректности работы каждого модуля.

1. **Общая часть**
   1. **Анализ методов решения**

Всего можно выделить два метода решения задачи — ручной и автоматизированный.

Ручной метод решения задачи подразумевает собой организацию всей системы силами сотрудников предприятия, базируясь на принципах разделения труда, ведения документации на бумажных носителях, проводя расчеты допотопными технологиями. Такой метод неактуален, неудобен, влечет за собой огромные траты на организацию и поддержку работоспособности, нестабилен, так как имеет место быть человеческий фактор, а вместе с ним велик риск катастрофических ошибок рабочего процесса.

Автоматизированный метод представляет собой разработку программного обеспечения, обеспечивающего постоянную, быструю, стабильную работу системы книжного магазина, не требует больших людских затрат, позволяет хранить всю информацию в базе данных, чтобы по первому требованию пользователь смог получить необходимые сведения.

Для решения задачи выпускной квалификационной работы целесообразно выбрать автоматизированный метод, поскольку в век современных технологий деятельность любой организации, предприятия должна быть автоматизирована.

Программное обеспечение должно быть способно обрабатывать заказы клиентов книжного магазина мгновенно, показывать имеющиеся в наличии книги, осуществлять поставки новых книг на склад магазина, ускорять процесс работы всей системы, устраняя затраты и ошибки персонала, используя имеющуюся систему защиты данных.

Для разработки программного обеспечения необходимы следующие программные продукты:

* среда разработки PyCharm;
* операционная система Windows;
* СУБД MySQL;
* система создания инсталляторов Inno Setup.
  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 — Сравнительный 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, например, из браузеров нескольких пользователей. | 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 |
| 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. |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | C# | Java | Python |
| 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  себя множество инструментов для тестирования совместимости. | 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.  Изначально Eclipse использовалась только для Java, но сейчас, благодаря плагинам и расширениям, ее функции значительно расширились. | 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  используемых сред разработки. |

Продолжение таблицы 3

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | Microsoft Visual Studio | Eclipse | PyCharm |
|  |  | Именно из-за возможности расширить Eclipse своими модулями эта платформа и завоевала свою популярность среди разработчиков. |  |
| 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  Широко используется в большинстве компаний по разработке программных продуктов. |

Продолжение таблицы 3

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | Microsoft Visual Studio | Eclipse | PyCharm |
| 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. **Методы исследования**

Метод исследования — это способ получения информации, помогающий решить задачи и достигнуть цели выпускной квалификационной работы.

Методы исследования делятся на две большие группы: теоретические и эмпирические. Это деление основано на используемом способе получения информации: эмпирические методы построены на основе наблюдений за объектом, а теоретические — на основе полученной информации.

В качестве теоретического метода исследования был выбран метод анализа.

Анализ — это изучение объекта (предметной области) на основе исследования его составляющих.

В процессе анализа изучалась предметная область книжного магазина — из каких частей она состоит, для чего нужна, её актуальность.

В составе предметной области книжного магазина в результате анализа были выделены следующие объекты:

* книга;
* поставщик;
* поставка;
* сотрудник;
* клиент;
* заказ;
* автор;
* жанр;
* издательство.

Для каждого объекта также были определены атрибуты — некие свойства объекта, отличающие его от других:

* для объекта «Поставщик» — код поставщика, наименование компании, номер телефона, юридический адрес;
* для объекта «Поставка» — код поставки, код поставщика, код книги, дата создания, количество книг, стоимость;
* для объекта «Сотрудник» — код сотрудника, фамилия, имя, отчество, должность, логин, пароль, фото, последний вход, тип входа;
* для объекта «Клиент» — код клиента, фамилия, имя, отчество, дата рождения, номер телефона, email;
* для объекта «Книги» — код книги, наименование книги, жанр, автор, издательство, стоимость, год издания, количество книг;
* для объекта «Заказ» — код заказа, номер заказа, книга, клиент, дата создания, время создания, сотрудник;
* для объекта «Жанр» — код жанра, жанр;
* для объекта «Издательство» — код издательства, название издательства;
* для объекта «Автор» — код автора, фамилия, имя, отчество, дата рождения.

Таким образом, в результат анализа предметной области были определены её объекты и атрибуты этих объектов. Отчасти, данный процесс можно отнести к другому методу тестирования — моделированию. При моделировании все абстрактные объекты представляются в виде некой модели, чаще всего — схемы или диаграммы.

* 1. **Описание системы управления базами данных**

MySQL — это система управления реляционными базами данных.

В реляционной базе данных данные хранятся не все скопом, а в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. SQL как часть системы MySQL можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

Для чего нужна MySQL:

* удобный доступ к базам данных;
* корректное распределенное хранение данных на сервере;
* быстрый поиск нужной информации в базе с помощью языка SQL;
* идентификация и обработка отдельных данных, их преобразование и отправка;
* создание, редактирование и удаление записей, которые есть в базе;
* отправка транзакций — “пакетов” из нескольких запросов к базе;
* множественный доступ к базе с разных устройств — например, из браузеров нескольких пользователей;
* выдача разных прав доступа различным пользователям;
* обеспечение безопасности данных: их защита, шифрование и контроль доступа;
* возможность контролировать версии базы данных: делать бэкапы, обновлять базу или откатывать назад;
* контроль состояния базы данных.

1. **Специальная часть**
   1. **Постановка задачи**

Главная цель выпускной квалификационной работы — разработка системы управления для книжного магазина.

В процессе разработки системы необходимо учесть и выполнить следующие задачи:

* разработать базу данных, содержащую в себе все сведения необходимые для работы системы;
* реализовать модули управления заказами, управления закупками книг, учета книг на складе;
* сформировать отчеты по деятельности каждого модуля;
* провести проверку корректности работы каждого модуля.

Для информационной системы книжного магазина были разработаны следующие требования:

1. требования к функционалу ПО:

* программное обеспечение должно взаимодействовать с СУБД MySQL для хранения данных о книгах, клиентах, поставщиках;
* все функции ПО должны работать четко, слаженно;
* функцию разделенного доступа к данным в БД в программном обеспечении необходимо реализовать с помощью функции авторизации в системе, разделить функционал между должностями сотрудников — пользователей;
* вести учет истории входа-выхода пользователей.

1. требования к интерфейсу ПО:

* интерфейс ПО должен отображать специфику книжного магазина;
* должен быть интуитивно понятен;
* все необходимые разделы, функциональные кнопки должны быть легкодоступны;
* при необходимости изменить разрешение окна, все объекты интерфейса должны автоматически адаптироваться под установившийся размер окна, не изменив своего положения относительно общего плана.

1. требования к уровню доступа пользователей ПО к данным системы:

* для пользователя “Администратор” обеспечить возможность полного взаимодействия со всеми функциями ПО;
* для пользователей “Старший смены” и “Продавец” обеспечить доступ к взаимодействию с разделами создания и управления заказами, учета товара на складе.

1. требования к производительности системы:

Программное обеспечение предназначено для работы на компьютерах следующей комплектации:

* операционная система — Windows 10/11;
* процессор — AMD/Intel с тактовой частотой 2Ghz и более;
* объем ОЗУ — 8гб и более;
* объем памяти на жестком диске — не менее 2гб.
  1. **Схема алгоритма выполнения программы**

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное число действий. В старой трактовке вместо слова “порядок” использовалось слово “последовательность”, но по мере развития параллельности в работе компьютеров слово “последовательность” стали заменять более общим словом “порядок”. Это связано с тем, что работа каких-то инструкций алгоритма может быть зависима от других инструкций или результатов их работы.

Таким образом, некоторые инструкции должны выполняться строго после завершения работы инструкций, от которых они зависят. Независимые инструкции или инструкции, ставшие независимыми из-за завершения работы инструкций, от которых они зависят, могут выполняться в произвольном порядке, параллельно или одновременно, если это позволяют используемые процессор и операционная система.

Для записи алгоритма решения задачи применяются следующие изобразительные способы их представления:

* словесно-формульное описание;
* блок-схема (схема графических символов);
* алгоритмические языки;
* операторные схемы;
* псевдокод.

Для информационной системы книжного магазина был составлен следующий алгоритм:

1. начало работы программы;
2. авторизация в системе — на данном этапе происходит проверка введенных данных на корректность, если данные верны — происходит вход в систему, если нет — программа возвращается к этапу авторизации;
3. после входа в систему пользователю, в зависимости от его уровня доступа к данным, предоставляются разделы системы:

* в разделе “Управление заказами” пользователь может просмотреть все заказы, создать новый заказ, удалить существующий, сохранить все изменения в базу данных;
* в разделе “Управление поставками” пользователь может добавить нового поставщика, удалить существующего, оформить новую поставку, просмотреть информацию по поставщикам и по поставкам;
* в разделе “Учет книг на складе” пользователю предоставляется вывод на экран всей информации по имеющимся в наличии книгам.

1. после сохранения изменений пользователь может продолжить или завершить работу, выйдя из системы.

На рисунке 1 представлено схематичное изображение алгоритма работы программы.

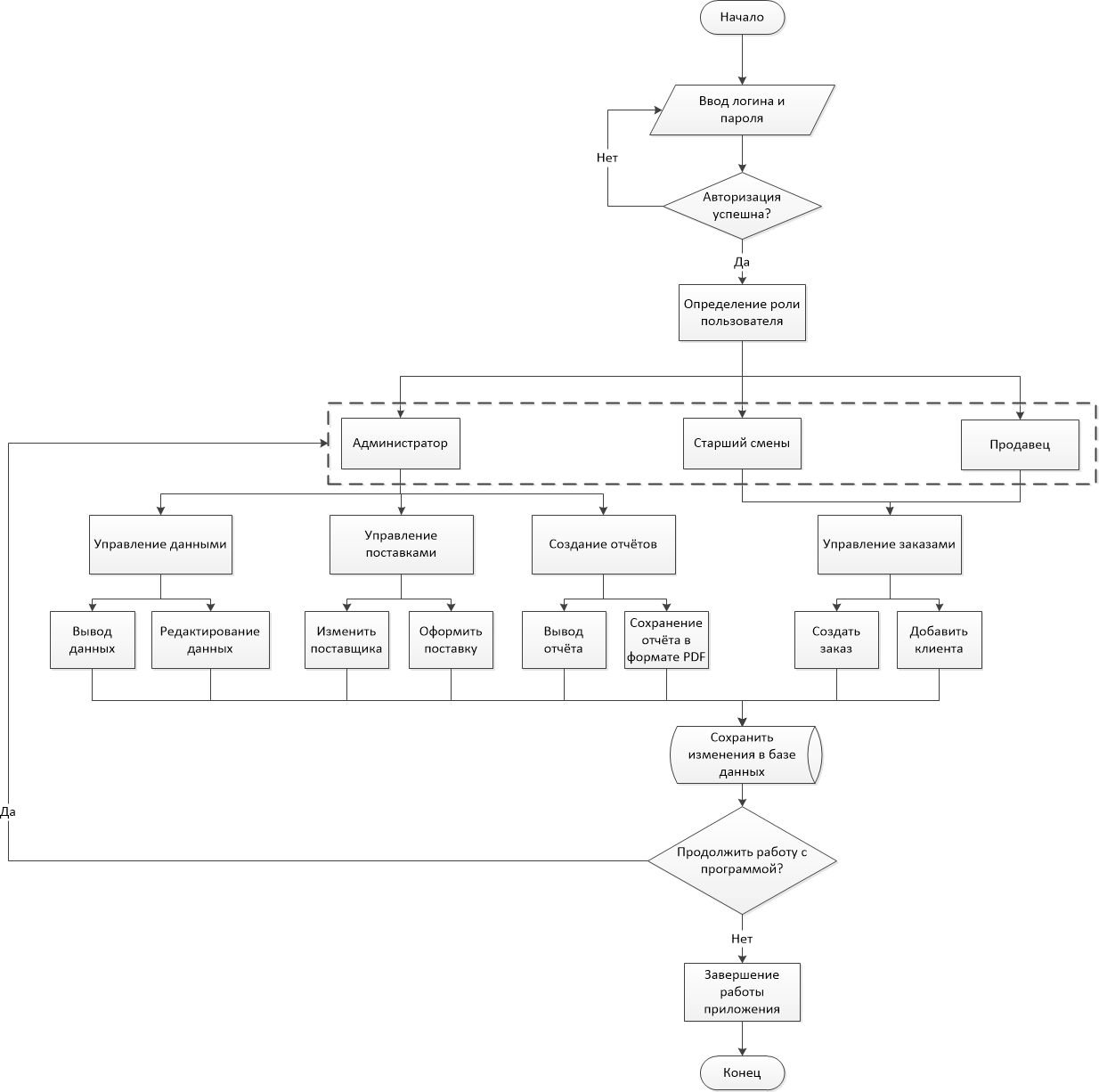


Рисунок 1 — Алгоритм программного средства

* 1. **Планирование базы данных и разработка схем реализации базы данных**

В составе предметной области книжного магазина в результате анализа были выделены следующие объекты:

* книга;
* поставщик;
* поставка;
* сотрудник;
* клиент;
* заказ;
* автор;
* жанр;
* издательство.

Для каждого объекта также были определены атрибуты — некие свойства объекта, отличающие его от других:

* для объекта «Поставщик» — код поставщика, наименование компании, номер телефона, юридический адрес;
* для объекта «Поставка» — код поставки, код поставщика, код книги, дата создания, количество книг, стоимость;
* для объекта «Сотрудник» — код сотрудника, фамилия, имя, отчество, должность, логин, пароль, фото, последний вход, тип входа;
* для объекта «Клиент» — код клиента, фамилия, имя, отчество, дата рождения, номер телефона, email;
* для объекта «Книги» — код книги, наименование книги, жанр, автор, издательство, стоимость, год издания, количество книг;
* для объекта «Заказ» — код заказа, номер заказа, книга, клиент, дата создания, время создания, сотрудник;
* для объекта «Жанр» — код жанра, жанр;
* для объекта «Издательство» — код издательства, название издательства;
* для объекта «Автор» — код автора, фамилия, имя, отчество, дата рождения.

Таким образом, в результат анализа предметной области были определены её объекты и атрибуты этих объектов. Отчасти, данный процесс можно отнести к другому методу тестирования — моделированию.

При моделировании все абстрактные объекты представляются в виде некой модели, чаще всего — схемы или диаграммы.

В результате моделирования была спроектирована модель — ER-диаграмма, представленная на рисунке 2.

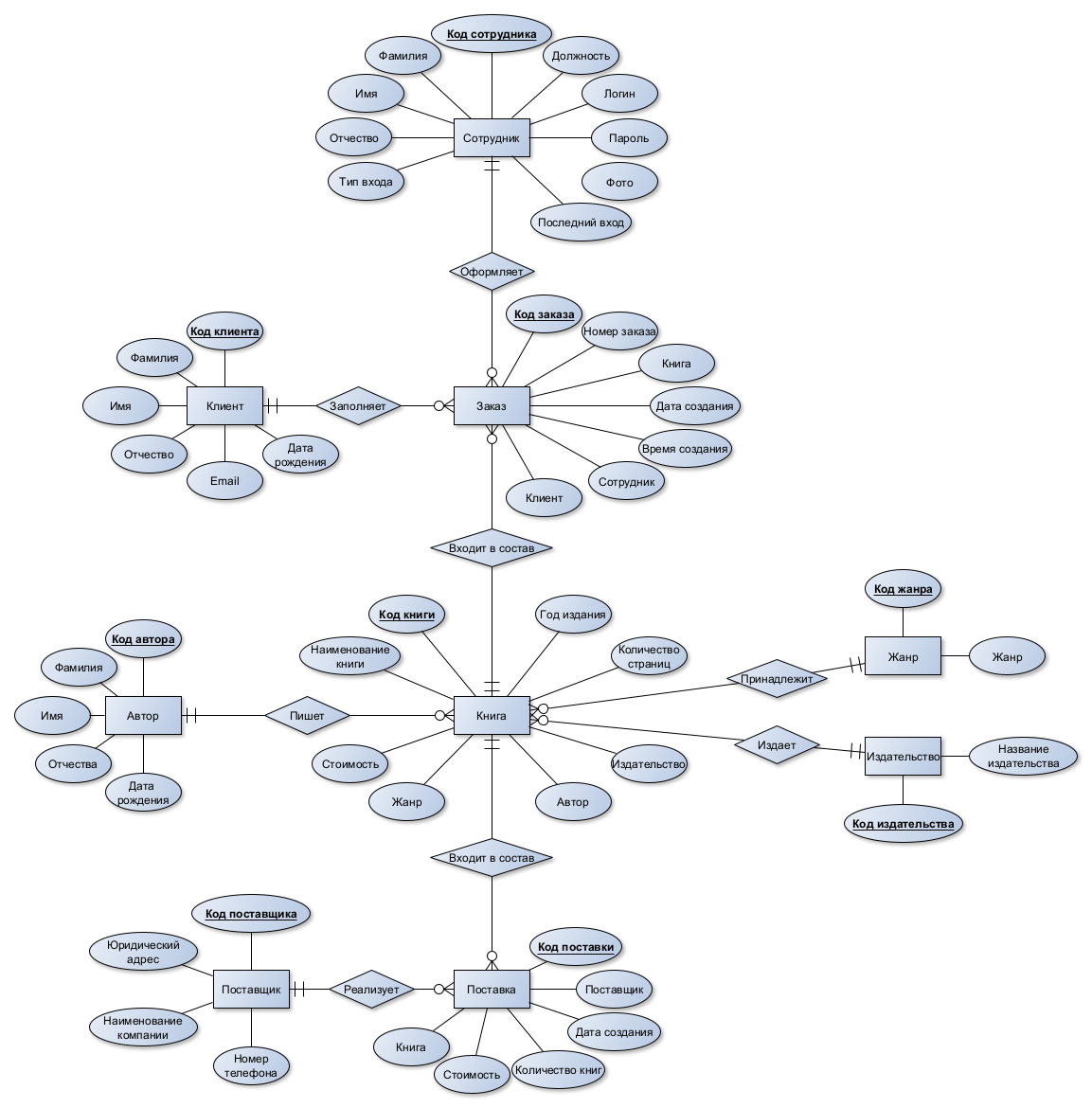


Рисунок 2 — ER-диаграмма

* 1. **Текст программы**
     1. **Создание интерфейса программного продукта**

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 3 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 3 — Макет формы авторизации

Затем проектируется макет формы главного окна приложения. Результат представлен на рисунке 4. В верхней части макета указан заголовок, во вторичном окне выводятся данные, ниже расположены кнопки перехода по формам.



Рисунок 4 — Макет формы главного окна приложения

Следующий макет — макет формы окна вывода данных. Представлен на рисунке 5. 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 5 — Макет формы окна вывода данных

Далее следует проектирование макета окна вывода данных без кнопок. Представлен на рисунке 6. В верхней части представлен заголовок. Ниже расположено поле для вывода данных из базы данных.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 — Макет формы окна вывода данных

Следующий макет — макет формы окна добавления данных. Представлен на рисунке 7. 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 7 — Макет формы окна добавления данных

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

Для создания форм интерфейса используется инструмент QtDesigner [1] 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 Buttons» 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 программы;
* QTabWidget —для перехода по формам 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 8.

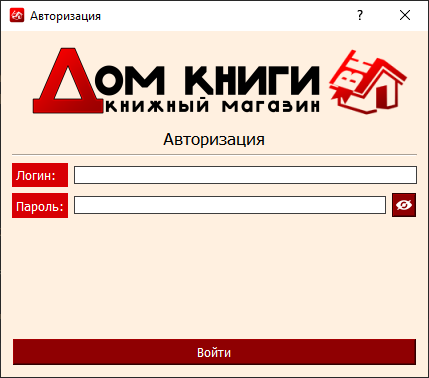


Рисунок 8 — Форма авторизации

В зависимости от должности сотрудника, в программе реализовано разделение уровня доступа к содержимому базы данных. На рисунке 9 показана форма окна информации о пользователе.

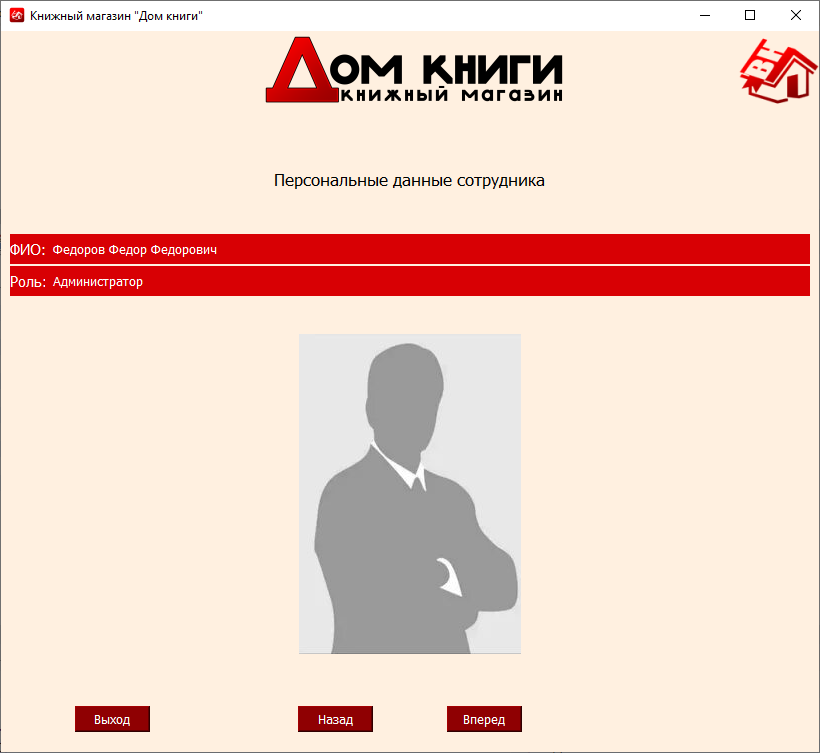


Рисунок 9 — Форма окна информации о пользователе

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

* просмотр истории входа сотрудников;
* просмотр всех заказов. Выводится список всех заказов.
* создание отчётов. Возможность создать отчёты по “количеству книг по дням за период времени”, “количеству заказов по дням за период времени по каждой книге”, “количеству заказов по дням за период времени;
* просмотр книг. Также реализована возможность просмотра всех книг, добавления новых и удаления имеющихся, сохранения изменений;
* просмотр жанров. В форме реализована возможность добавить новый жанр или же удалить выбранный жанр и сохранить все изменения в базу данных;
* просмотр авторов. Возможно просмотреть всех авторов, 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 клиент и книга. Также реализована возможность создания штрих-кода;
* добавление нового клиента. Открывается окно создания нового клиента. Указывается фамилия, имя, отчество клиента, дата рождения, email;
* просмотр всех клиентов. Выводится список всех клиентов;
* просмотр всех заказов. Выводится список всех заказов.

Формы окон приложения представлены в приложении А. Листинг форм в формате “.py” представлен в приложении Б.

* + 1. **Разработка модуля базы данных**

Для полноценной работы программы необходимо реализовать модуль для взаимодействия с базой данных.

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. 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 “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.

В окне оформления заявки на поставку книг реализуются следующие запросы:

* получение списка поставщиков;
* получение списка книг;
* сохранение заявки.

В окне просмотра поставщиков реализуются следующие запросы:

* изменение поставщика;
* добавление поставщика;
* удаление поставщика;
* получение списка поставщиков.

В окне просмотра заявок реализуются следующие запросы:

* изменение заявки;
* удаление заявки;
* сохранение заявки;
* получение списка поставщиков.

В окне просмотра заказов реализуются следующие запросы:

* изменение заказа;
* удаление заказа;
* сохранение заказа;
* получение списка заказов.

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.

Для реализации данной системы был использован один из самых используемых подходов к программированию — объектно-ориентированное программирование (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;
* insert\_genre(self, genre) — добавление нового жанра;
* insert\_author(self, fio, date) — добавление нового автора;
* insert\_ph(self, name) — добавление нового издательства;
* insert\_provider(self, name, address, phone) — добавление нового поставщика;
* insert\_delivery(self, provider, date, book, quantity, cost) — добавление новой поставки;
* update\_genre(self, id, genre) — обновление жанра;
* update\_author(self, id, fio, date) — обновление автора;
* update\_ph(self, id, ph) — обновление списка издательств;
* update\_provider(self, id, name, address, phone) — обновление списка поставщиков;
* update\_req(self, id, number, date, time, client, employee, book) — обновление списка заказов;
* update\_pr\_od(self, id, provider, date, book, quantity, cost) — обновление списка поставок;
* delete\_client(self, id) — удаление клиента;
* delete\_genre(self, id) — удаление жанра;
* delete\_author(self, id) — удаление автора;
* delete\_ph(self, id) — удаление издательства;
* delete\_provider(self, id) — удаление поставщика;
* delete\_req(self, id) — удаление заказа;
* delete\_pr\_od(self, id) — удаление поставки;
* select\_genre(self) — получение списка жанров;
* select\_author(self) —получение списка авторов;
* select\_ph(self) — получение списка издательств;
* select\_providers(self) — получение списка поставщиков;
* select\_provider\_order(self) — получение списка поставок;
* select\_orders(self) — получение списка заказов;
* get\_genres(self) — получение наименований жанров;
* get\_authors(self) — получение ФИО авторов;
* get\_ph(self) — получение названий издательств;
* get\_providers(self) — получение названий поставщиков;
* get\_emp\_id(self, surname, name, lastname) — получение кода сотрудника;
* get\_genre\_id(self, name) — получение кода жанра;
* get\_author\_id(self, fio) — получение кода автора;
* get\_ph\_id(self, name) — получение кода издательства;
* get\_book\_id(self, name) — получение кода книги;
* get\_data\_book(self) — получение данных о заказе;
* get\_book\_name(self, code) — получение названия книги;
* get\_book\_cost(self, code) — получение стоимости книги;
* get\_provider\_id(self, name) — получение кода поставщика;
* get\_provider\_name(self, code) — получение названия поставщика;
* 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.
  + 1. **Разработка модуля графического интерфейса**

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;
* closeEvent(self, event) — выход из программы;
* page\_all\_pr\_od(self) — переход к странице с таблицей заказов;
* page\_all\_requests(self) — переход к странице с таблицей заказов;
* page\_add\_pr\_od(self) — переход к странице с таблицей заказов;
* 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;
* updateTableGenre(self) — обновление таблицы “Жанр”;
* updateTableAuthor(self) — обновление таблицы “Автор”;
* updateTablePH(self) — обновление таблицы “Издательство”;
* updateTableProvider(self) — обновление таблицы “Поставщик”;
* updateTableProviderOrder(self) — обновление таблицы “Поставки”;
* updateTableOrder(self) — обновление таблицы “Заказ”;
* 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;
* new\_genre(self) — добавление жанра;
* delete\_genre(self) — удаление выбранного жанра;
* get\_from\_table\_genre(self) — получение данных из таблицы “Жанр” для записи в базу данных;
* save\_genre(self) — сохранение данных о жанрах в базу данных;
* new\_author(self) — добавление автора;
* delete\_author(self) — удаление выбранного автора;
* get\_from\_table\_author(self) — получение данных из таблицы “Автор” для записи в базу данных;
* save\_author(self) — сохранение данных об авторах в базу данных;
* new\_ph(self) — добавление издательства;
* delete\_ph(self) — удаление выбранного издательства;
* get\_from\_table\_ph(self) — получение данных из таблицы “Издательство” для записи в базу данных;
* save\_ph(self) — сохранение данных об издательствах в базу данных;
* new\_provider(self) — добавление поставщика;
* delete\_provider(self) — удаление выбранного поставщика;
* get\_from\_table\_provider(self) — получение данных из таблицы “Поставщик” для записи в базу данных.
* save\_provider(self) — сохранение данных о поставщике в базу данных;
* delete\_req(self) — удаление выбранного заказа;
* get\_from\_table\_req(self) — получение данных из таблицы “Заказы” для записи в базу данных;
* save\_req(self) — сохранение данных о заказе;
* delete\_pr\_od(self) — удаление выбранной поставки;
* get\_from\_table\_pr\_od(self) — получение данных из таблицы “Поставки” для записи в базу данных;
* save\_pr\_od(self) — сохранение данных о поставках в базу данных;
* build\_combobox\_clients(self) — добавление списка клиентов в ComboBox;
* build\_combobox\_books(self) — добавление списка книг в ComboBox;
* build\_combobox\_genre(self) — добавление списка жанров в ComboBox;
* build\_combobox\_author(self) — добавление списка авторов в ComboBox;
* build\_combobox\_ph(self) — добавление списка издательств в ComboBox;
* build\_combobox\_provider(self) — добавление поставщиков 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”;
* add\_new\_provider\_order(self) — оформление новой поставки и показ в ListWidget.
* add\_book\_to\_pr\_od(self) — добавление книги в заказ;
* save\_provider\_order(self) — сохранение заказа в базу данных;
* 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 какого времени она длится. При успешном входе передает в фасад время и логин успешного входа (для записи в базу данных), записывает индексы доступных страничек StackedWidget (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.
  + 1. **Организация взаимосвязи модулей программы**

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\_for\_authorization(self, login) — авторизация по логину;
* 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\_genres(self) — получение списка жанров;
* get\_authors(self) — получение списка авторов;
* get\_ph(self) — получение списка издательств;
* get\_providers(self) — получение списка поставщиков;
* get\_book\_name(self, id) — получение списка названий книг;
* get\_book\_cost(self, id) — получение стоимости книги;
* get\_provider\_name(self, id) — получение названия поставщика;
* get\_id\_client(self, surname, name, lastname) — получение кода клиента;
* get\_for\_authorization(self, login) — получение информации о сотруднике;
* get\_id\_emp(self, surname, name, lastname) —получение кода сотрудника;
* get\_genre\_id(self, name) — получение кода жанра;
* get\_author\_id(self, au) — получение кода автора;
* get\_ph\_id(self, name) — получение кода издательства;
* get\_provider\_id(self, name) — получение кода поставщика;
* 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, emp, book) — создание заказа;
* read\_clients(self) — просмотр 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;
* create\_delivery(self, provider, date, book, quantity, cost) — создание поставки;
* insert\_genre(self, genre) — добавление жанра;
* insert\_author(self, fio, date) — добавление автора;
* insert\_ph(self, name) — добавление издательства;
* insert\_provider(self, name, address, phone) — добавление поставщика;
* update\_genre(self, id, genre) — обновление списка жанров;
* update\_author(self, id, fio, date) — обновление списка авторов;
* update\_ph(self, id, ph) — обновление списка издательств;
* update\_provider(self, id, name, address, phone) — обновление списка поставщиков;
* update\_req(self, id, number, date, time, client, employee, book) — обновление списка заказов;
* update\_pr\_od(self, id, provider, date, book, quantity, cost) — обновление списка поставок;
* delete\_genre(self, id) — удаление жанра;
* delete\_author(self, id) — удаление автора;
* delete\_ph(self, id) — удаление издательства;
* delete\_book(self, id) — удаление книги;
* delete\_provider(self, id) — удаление поставщика;
* delete\_req(self, id) — удаление заказа;
* delete\_pr\_od(self, id) — удаление поставки;
* read\_genre(self) — просмотр списка жанров;
* read\_author(self) — просмотр списка авторов;
* read\_ph(self) — просмотр списка издательств;
* read\_provider(self) — просмотр списка поставщиков;
* read\_order(self) — просмотр списка заказов;
* read\_provider\_orders(self) — просмотр списка поставок.

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. **Схемы программы**
     1. **Схема структуры программы**

Для информационной системы книжного магазина структура состоит из следующих компонентов:

* сама информационная система, подразделяющаяся на несколько подсистем:

1. подсистема управления заказами. Данная подсистема отвечает за создание, изменение, удаление заказов;
2. подсистема управления закупками книг. Данная подсистема обеспечивает формирование закупок книг и поставок их на склад магазина.
3. подсистема учета книг на складе. Данная подсистема обеспечивает просмотр списка всех имеющихся книг.

Схематично структура проекта изображается как граф — в качестве основной вершины выступает сама информационная система, дочерними вершинами являются подсистемы. Представлена на рисунке 10.



Рисунок 10 — Структурная схема

* + 1. **Архитектура программы**

Архитектура программного обеспечения (Software Architecture) - набор структур, необходимый для рассуждения о системе, содержащий элементы ПО, связи между ними и их свойства.

Для информационной системы книжного магазина архитектура ПО должна отражать следующие элементы:

* непосредственно, каждый модуль системы, как самостоятельный объект, при этом объединить их в одну большую систему;
* объект базы данных, обеспечивающий хранение данных, получаемых от всех модулей системы ПО;
* объект графического интерфейса ПО;
* формирование отчетов в формате PDF как отдельная функция, но не являющаяся самостоятельным модулем.

Таким образом реализуется, представленная на рисунке 11, архитектура программного средства книжного магазина.

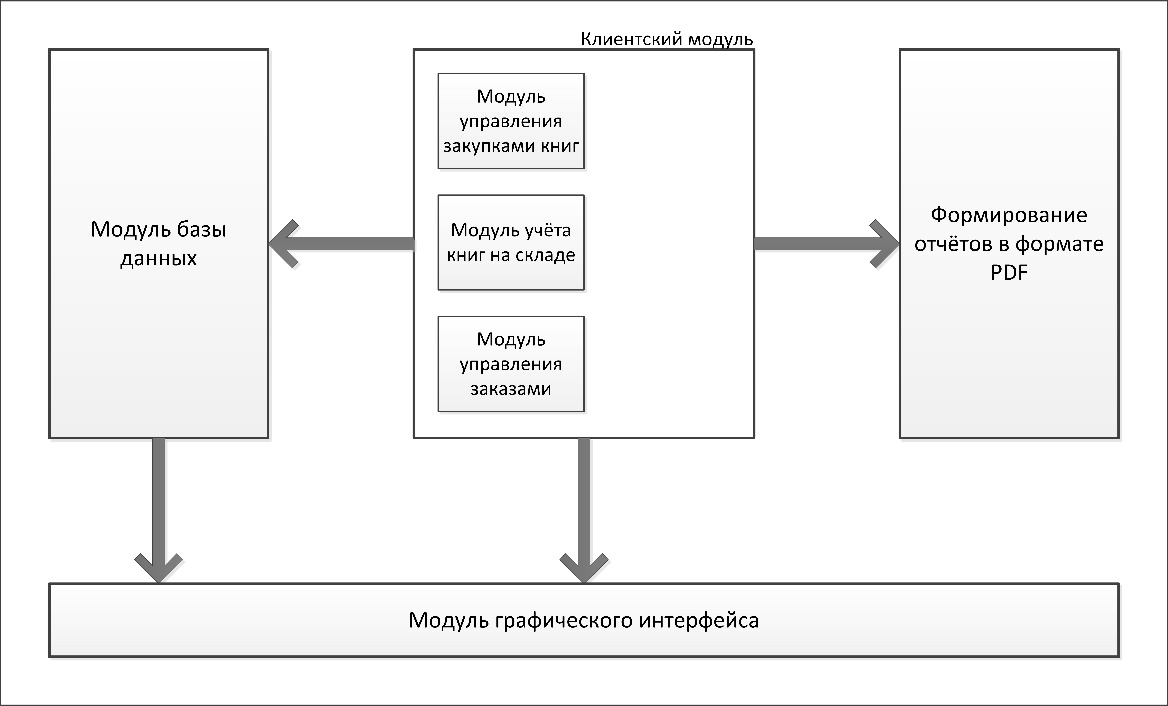


Рисунок 11 — Архитектура программного средства

* 1. **Инструкция по эксплуатации программы**
     1. **Общие сведения**
        1. **Назначение программы**

Информационной системой книжного магазина является 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.

Данная информационная система имеет в своем составе следующие подсистемы:

* подсистема управления заказами;
* подсистема управления закупками книг;
* подсистема учета книг на складе.

Каждая подсистема должна работать четко и слаженно. Для работы каждой из них вести отчетность по срокам выполнения работ. Подсистема управления заказами должна обеспечивать возможность создать новый заказ для покупателя, выбрав желаемую книгу из списка имеющихся, вносить данные о клиенте.

Подсистема управления закупками книг необходима для обеспечения книжного магазина товаром — должно обеспечиваться добавление новых книг, информацией о них: название, жанр, автор, издательство, а также работа с поставщиками — добавление или удаление поставщика, оформление закупки книг в определённом количестве.

Подсистема учета книг на складе должна предоставлять пользователю сводку данных по книгам — сколько книг всего на складе, список книг по жанрам, по авторам, по издательствам, а также отображает те книги, количество которых подходит к концу, для того чтобы восполнить запасы на складе.

* + - 1. **Начало работы**

Для работы с данной информационной системой необходимо установить приложение BookShop и MySQL Workbench.

В MySQL Workbench нужно добавить базу данных для работы приложения (если есть подключение к серверу):

1. Открыв подключение, во вкладке “Server” выбрать “Data Import”. Представлено на рисунке 12;

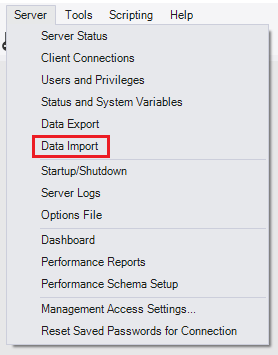


Рисунок 12 — Выбор импорта данных

1. выбрать пункт "Import from self-Contained File" и указать путь к файлу из проекта "DumpBookShop.sql". Представлено на рисунке 13;

image

Рисунок 13 — Импорт данных для размещения базы данных

1. нажать на кнопку «Start Import», после чего в списке доступных баз данных появится необходимая база данных под названием “bookshop”.
   * 1. **Установка и запуск программы**

Установка программы происходит следующим образом:

1. Запускается файл инсталлятора “BookShopSetup.exe”. На экран выводится окно инсталлятора. Представлено на рисунке 14;

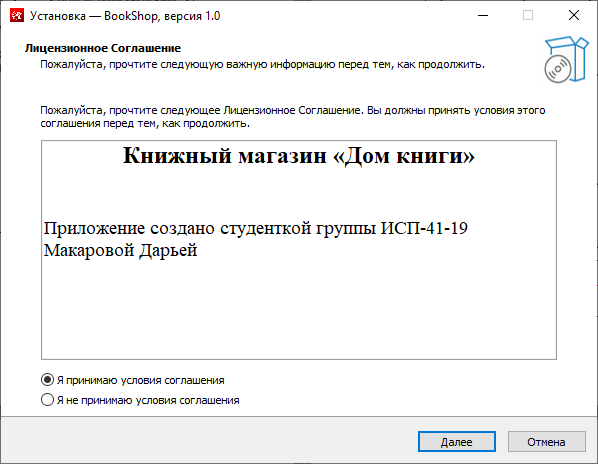


Рисунок 14 — Окно инсталлятора

1. необходимо выбрать пункт «Я принимаю условия соглашения» и нажать кнопку «Далее»;
2. следующее окно представляет собой выбор папки установки приложения. По умолчанию, приложение будет установлено по следующему пути: «C:\Program Files (x86)\BookShop». При необходимости можно изменить путь, нажав кнопку «Обзор». После выбора необходимо нажать кнопку «Далее»;
3. в следующем окне предоставляется возможность выбрать папку в меню «Пуск»;
4. следующее окно позволяет пользователю выбрать, создавать ли значок приложения на рабочем столе или нет. При нажатии кнопки «Далее» открывается последнее окно установки, в котором пользователь может проверить пути установки и прочие задачи. Для начала установки необходимо нажать кнопку «Установить»;
5. на рисунке 15 представлено окно, свидетельствующее о том, что установка успешно завершена и приложение готово к использованию.

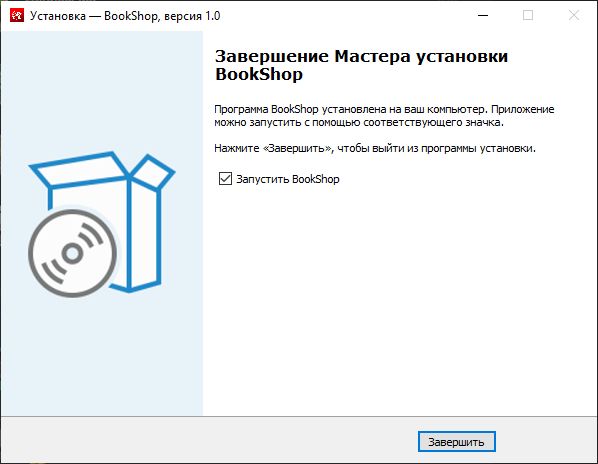


Рисунок 15 — Установка завершена

* + 1. **Работа с основными функциями**

При открытии программы появляется окно авторизации, представленным на рисунке 16. Пользователю необходимо ввести свои данные: логин и пароль.

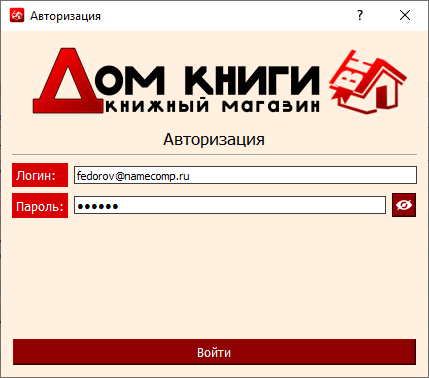


Рисунок 16 — Окно авторизации

При двух неудачных попытках ввода данных появляется капча, которую нужно ввести для дальнейшей авторизации. Корректно заполненные показаны на рисунке 17.

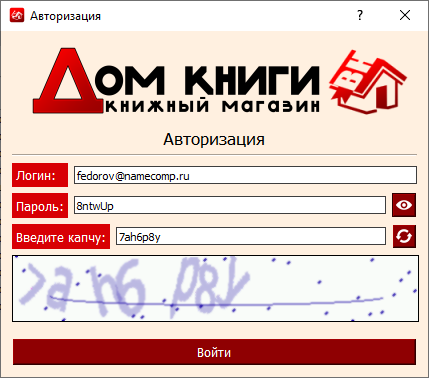


Рисунок 17 — Показ пароля и капчи в окне авторизации

После авторизации пользователь переходит на главное окно, где видит свои данные: ФИО, должность и фото. Представлено на рисунке 18.

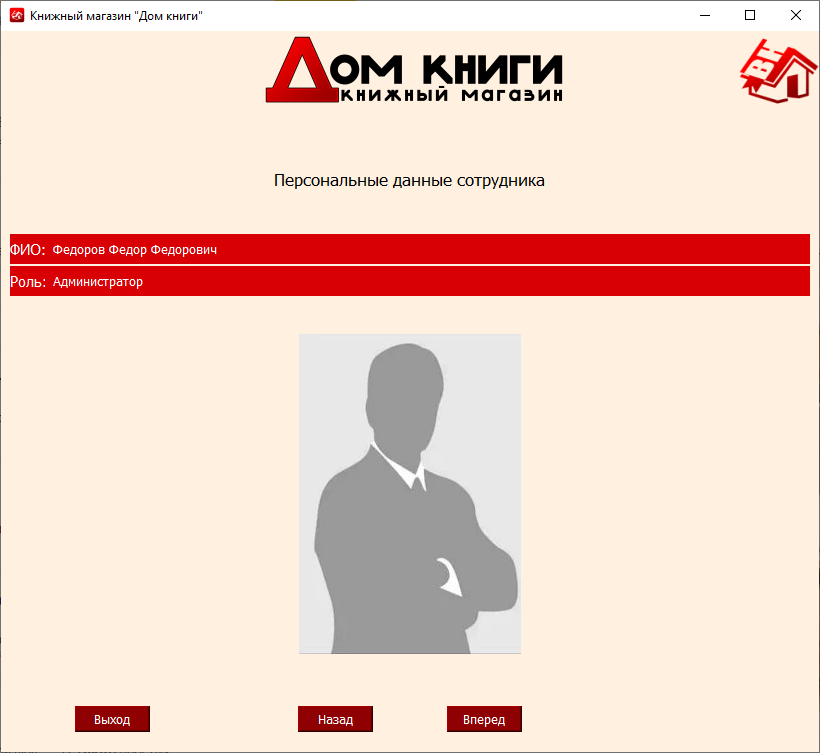


Рисунок 18 — Окно с персональными данными сотрудника

Перейдя к следующей странице по кнопке “Вперёд”, отобразятся основные функции пользователя в зависимости от его должности. Для пользователя с должностью “Администратор” выведется страница со следующими возможностями:

* во вкладке “Заказы” реализованы возможности создания заказа, выбрав клиента и книгу, просмотр всех заказов, просмотр всех клиентов и добавления нового клиента. Содержимое вкладки представлено на рисунке 19;

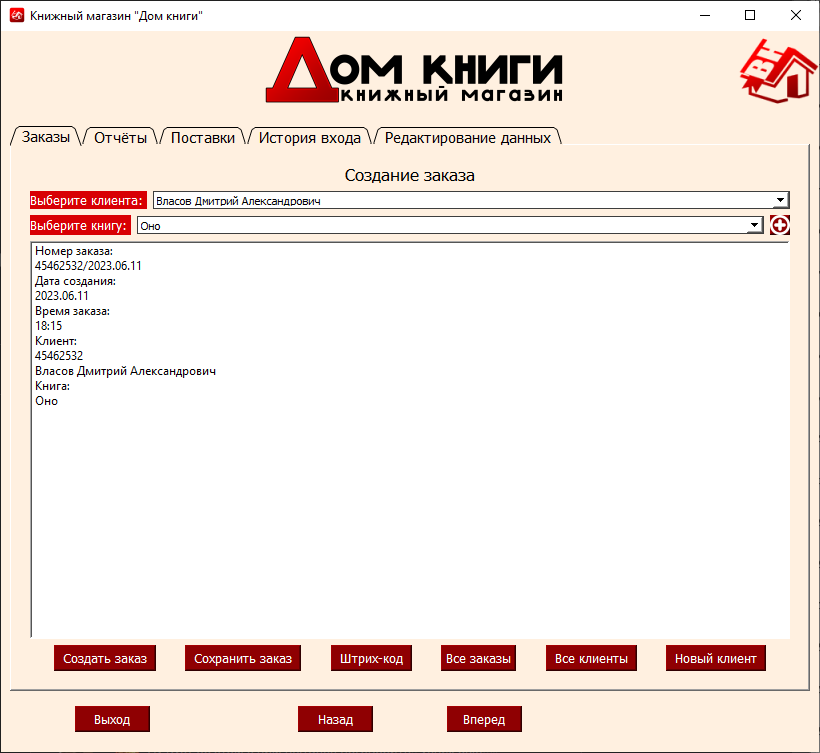


Рисунок 19 — Вкладка “Заказы”

* во вкладке “Отчёты” реализована возможность создания отчётов по “количеству книг по дням за период времени”, “количеству заказов по дням за период времени по каждой книге”, “количеству заказов по дням за период времени”. Содержимое вкладки представлено на рисунке 20;

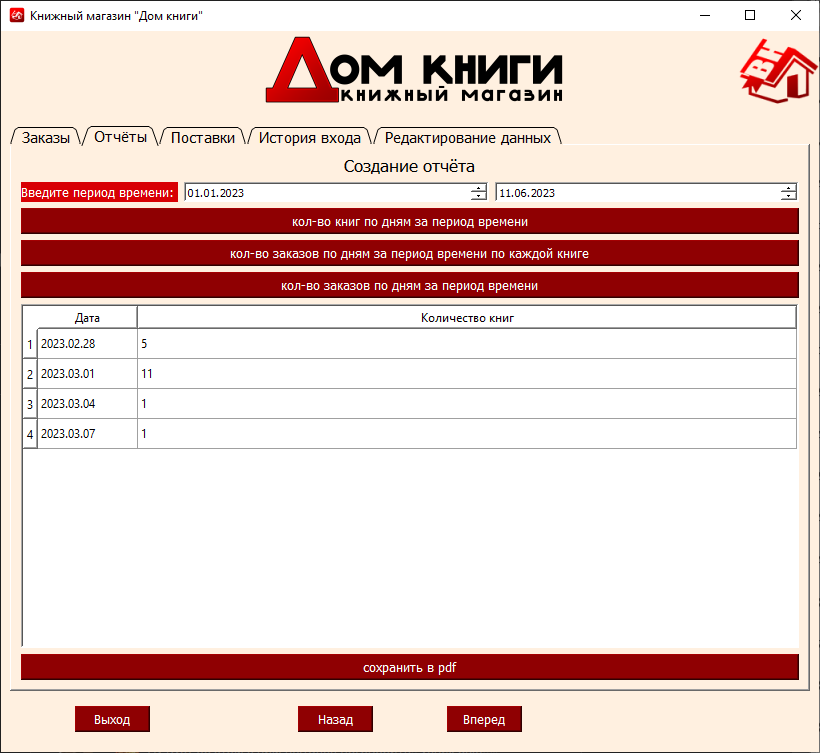


Рисунок 20 — Вкладка “Отчёты”

* во вкладке “Поставки” реализованы возможности создания заявки на поставку книг, выбрав поставщика, книгу и её количество, просмотр всех поставщиков и просмотр всех заявок. Содержимое вкладки представлено на рисунке 21;

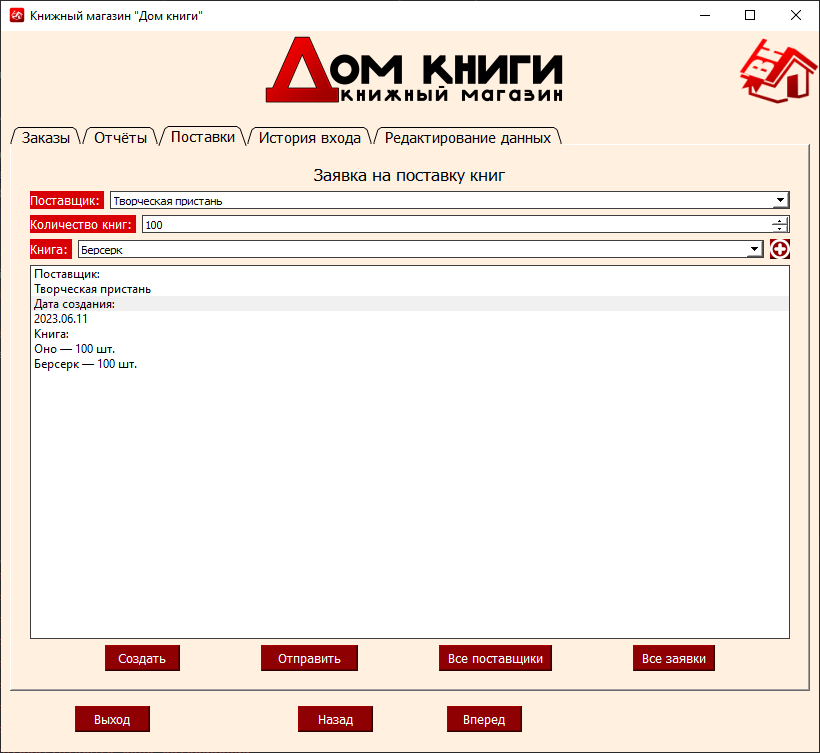


Рисунок 21 — Вкладка “Поставки”

* во вкладке “История входа” реализована возможность просмотра истории входа сотрудников. Содержимое вкладки представлено на рисунке 22;

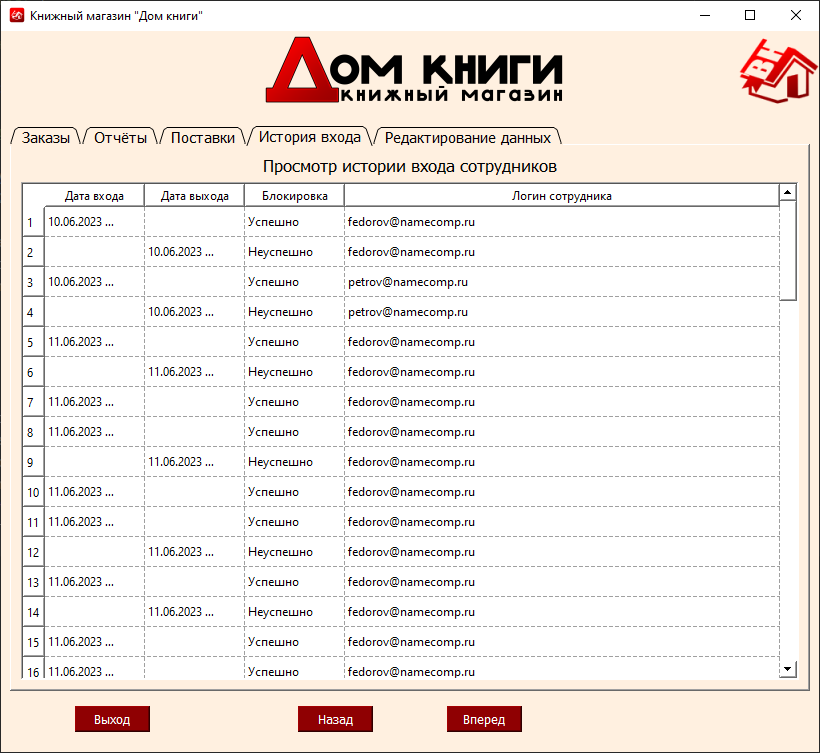


Рисунок 22 — Вкладка "История входа"

* во вкладке “Редактирование данных” реализованы возможности просмотра данных о “Книгах”, “Авторах”, “Жанрах”, “Издательствах”, их редактирование, добавление и удаление записей. Содержимое вкладки представлено на рисунках 23.

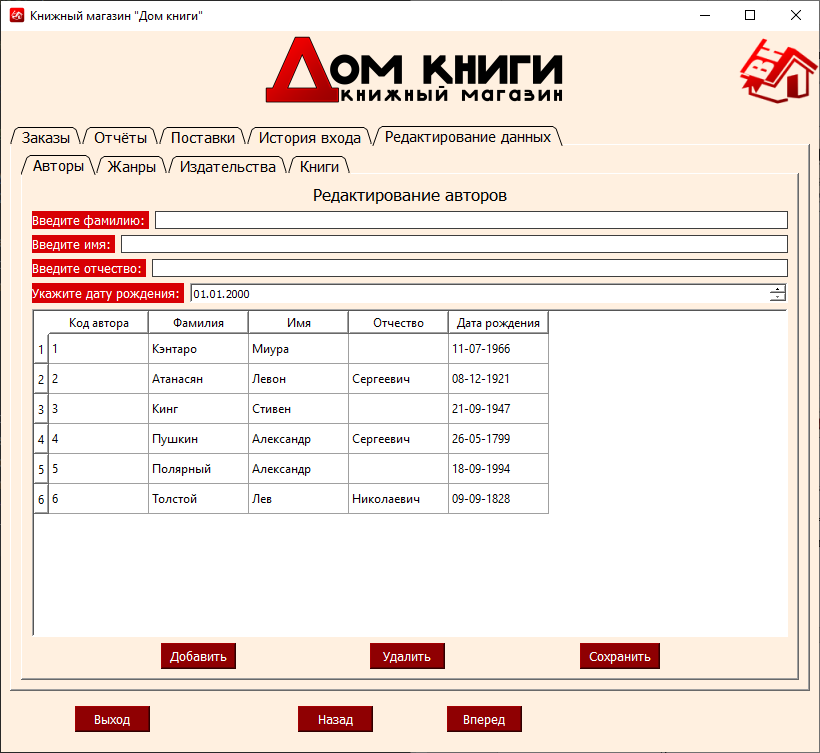


Рисунок 23 — Вкладка “Редактирование данных”

Для пользователя с должностью “Старший смены” и “Продавец” выведется страница с возможностью создания заказа, просмотра всех заказов, просмотра всех клиентов, добавления нового клиента. Представлена на рисунке 24.

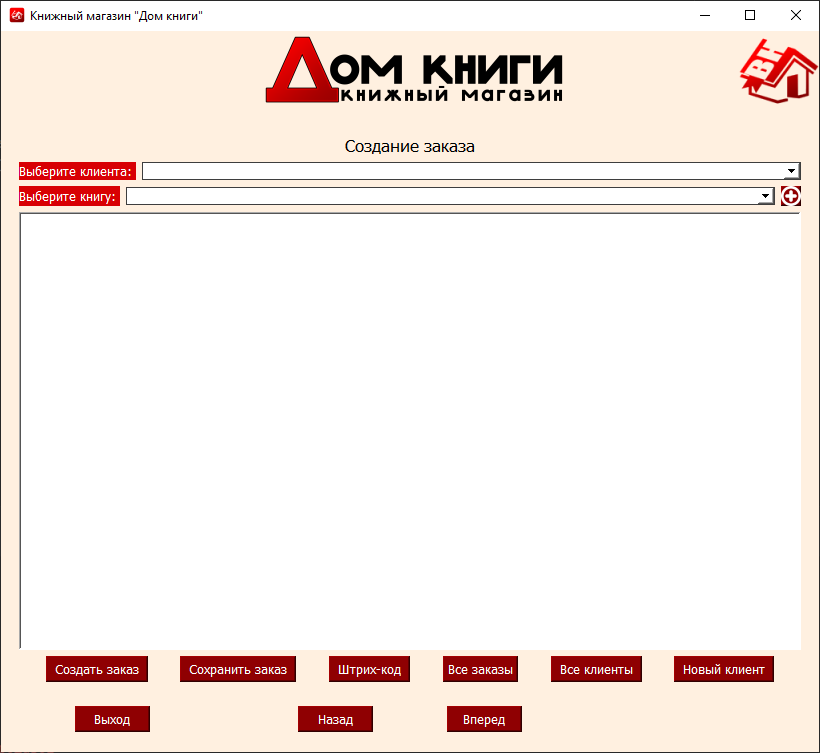


Рисунок 24 — Страница для пользователя с должностью “Старший смены” или “Продавец”

* 1. **Тестирование программных модулей**

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. |

Продолжение таблицы 4

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

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| Название теста | Входные данные | Результат |
| test\_delete\_author | Левый клик мыши по последней записи.  Левый клик мыши по кнопке btn\_delete\_author — “Сохранить”.  Левый клик мыши по кнопке btn\_save\_author — “Сохранить”. | Удаление записи об авторе из базы данных. |
| test\_delete\_ph | Левый клик мыши по последней записи.  Левый клик мыши по кнопке btn\_delete\_ph — “Сохранить”.  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 — “Сохранить”.  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. **Экономическая часть**
   1. **Расчет себестоимости разработки методики информационной системы книжного магазина**

В себестоимость разработки автоматизированной информационной системы входят следующие статьи затрат:

* основная заработная плата;
* дополнительная заработная плата;
* отчисления на социальное страхование;
* прочие расходы.

Разработку системы проводит один специалист: программист.

Зарплата программиста — 500 руб/час. При этом продолжительность рабочего дня каждого из них составляет 8 часов.

Расчет основной заработной платы представлен в таблице 5.

Таблица 5 — Расчет основной заработной платы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Этапы | Виды работ | Исполнитель | | Часовая тарифная ставка, руб. | Длительность выполнения, дней | Трудоемкость, нормо-час | Заработная плата, руб |
| Количество | Должность |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1. Начальный | Формулирование требований к программе, описание целей разработки | 1 | Программист | 500 | 1 | 8 | 4 000 |
| 2. Внешнее проектирование | Разработка архитектуры и структуры программы, алгоритма разработка интерфейса пользователя | 1 | Программист | 500 | 3 | 24 | 12 000 |

Продолжение таблицы 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Этапы | Виды работ | Исполнитель | | Часовая тарифная ставка, руб. | Длительность выполнения, дней | Трудоемкость, нормо-час | Заработная плата, руб |
| Количество | Должность |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3. Разработка и кодирование компонентов | Разработка каждого компонента и кодирование на языке программирования | 1 | Программист | 500 | 7 | 56 | 28 000 |
| 4. Основной этап | Отделка модулей | 1 | Программист | 500 | 1 | 8 | 4 000 |
| Тестирование компонентов | 1 | Программист | 500 | 1 | 8 | 4 000 |
| Комплексное тестирование программы | 1 | Программист | 500 | 1 | 8 | 4 000 |
| Оформление программной документации | 1 | Программист | 500 | 2 | 16 | 8 000 |
| 5. Заключительный этап | Коррекция программной документации | 1 | Программист | 500 | 1 | 8 | 4 000 |
| Итого | — | — | — | — | 17 | 136 | 68 000 |

К дополнительной заработной плате относятся: премии за выполнение основных этапов работ, выплата вознаграждения за выслугу лет и так далее. Дополнительная заработная плата является премиальной, по условиям премирования организации она составляет 20% от основной заработной платы.

Расчёт дополнительной заработной платы представлен в таблице 6.

Таблица 6 — Расчёт дополнительной заработной платы

|  |  |  |  |
| --- | --- | --- | --- |
| Основная  заработная плата, руб. | Дополнительная  заработная плата, % | Дополнительная  заработная плата, руб. | Итого  заработная плата, руб. |
| 68 000 | 20 | 13 600 | 81 600 |

К отчислениям на социальное страхование относятся отчисления в Пенсионный фонд Российской федерации, в Фонд социального страхования, в Фонд обязательного медицинского страхования и на социальное страхование от несчастных случаев и профессиональных заболеваний, если это прописано в договоре.

Отчисления на социальное страхование рассчитываются от суммы основной и дополнительной заработной платы по действующим ставкам.

Рассмотрим основные тарифы страховых взносов за работников в 2022 году. Они составляют:

1. на обязательное пенсионное страхование (ПФР)— 22 %;
2. на обязательное медицинское страхование (ФОМС)— 5,1 %;
3. на случай временной нетрудоспособности и материнства (ФСС)— 2,9 %;
4. на травматизм (ФСС тр)— от 0,2 % до 8,5 % в зависимости от класса профессионального риска, присвоенного основному осуществляемому виду деятельности.

Расчёт отчислений на социальное страхование представлен в таблице 7.

Таблица 7 — Расчёт отчислений на социальное страхование

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | База для расчёта отчислений на социальное страхование, руб. | Тариф страховых взносов, % | Отчисления на  социальное  страхование, руб. |
| 1 | 81 600 | 22 | 17 952 |
| 2 | 81 600 | 5,1 | 4 161,6 |
| 3 | 81 600 | 2,9 | 2 366,4 |
| 4 | 81 600 | 0,2 | 163,2 |
| Итого | | — | 24 643,2 |

К прочим расходам следует отнести расходы на обслуживание ЭВМ и плату за электроэнергию.

Затраты на электроэнергию рассчитываются исходя из потребляемой мощности устройства и тарифа на электроэнергию. В данном случае предполагается использование компьютера с мощностью 0,8 кВт час. Стоимость одного кВт часа электроэнергии равна 4,32 руб. Время использования электроэнергии в процессе разработки равно 136 часов.

Расчёт затрат на электроэнергию представлен в таблице 8.

Таблица 8 — Расчёт затрат на электроэнергию

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мощность устройства, кВт час | Время использования электроэнергии, час | Стоимость 1 кВт часа электроэнергии, руб. | Расчёт затрат на электроэнергию | Затраты на электроэнергию, руб. |
| 0,8 | 136 | 4,32 | 0,8\*136\*4,32 | 470 |

Расходы на обслуживание ЭВМ определяются из стоимости ЭВМ и срока ее полезного использования, по истечении которого, она подлежит замене. Примем это время равным 3 годам, в течение года ЭВМ используется 252 рабочих дня.

Расчёт расходов на обслуживание ЭВМ представлен в таблице 9.

Таблица 9 — Расчёт затрат на обслуживание ЭВМ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Стоимость ЭВМ, руб. | Срок полезного использования, лет | Время работы ЭВМ в год, дней | Продолжительность рабочего дня, час | Время использования электроэнергии, час | Расчёт затрат на обслуживание ЭВМ | Затраты на обслуживание ЭВМ, руб. |
| 60 000 | 3 | 252 | 8 | 136 | |  | | --- | | 60 000 \*136 | | 3\*252\*8 | | 1 349 |

Расчет себестоимости разработки системы представлен в таблице 10.

Таблица 10 — Расчет себестоимости разработки информационной системы книжного магазина

|  |  |  |
| --- | --- | --- |
| № п/п | Статьи затрат | Сумма, руб. |
| 1 | Основная заработная плата | 68 000 |
| 2 | Дополнительная заработная плата | 13 600 |
| 3 | Отчисления на социальное страхование | 24 643,2 |
| 4 | Затраты на электроэнергию | 470 |
| 5 | Затраты на обслуживание ЭВМ | 1 349 |
| Итого затрат | | 1. 062,2 |

2. **Мероприятия по технике безопасности**

При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора) должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м.

Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам и естественный свет падал преимущественно слева.

Рабочие места с ПЭВМ при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, рекомендуется изолировать друг от друга перегородками высотой 1,5–2,0 м.

Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600–700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Клавиатуру следует располагать на поверхности стола на расстоянии 100–300 мм от края, обращенного к пользователю, или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ.

Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья. При этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию. Поверхности сиденья, спинки и других элементов стула (кресла) должны быть полумягкими, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений. Конструкция рабочего стола должна обеспечивать возможность оптимального размещения на рабочей поверхности необходимого оборудования. Высота рабочей поверхности стола должна регулироваться в пределах от 680 до 760 мм, при отсутствии такой возможности ее высота должна составлять 270 мм.

Рабочий стол должен иметь свободное пространство для ног высотой не менее 600 мм, шириной не менее 500 мм, глубиной - не менее 650 мм. Рабочий стул должен обязательно иметь спинку. Рабочее кресло - спинку и подлокотники.

К работе с электронной аппаратурой допускаются лица, прошедшие инструктаж по правилам их безопасной эксплуатации. Работник должен знать инструкцию по эксплуатации каждого прибора.

Травмоопасность:

* при включении электронной аппаратуры в сеть;
* при выключении их из электросети;
* при работе с неисправными приборами;
* при несоблюдении инструкции по их эксплуатации.

Включать электронную аппаратуру в сеть в соответствии с потребляемым напряжением, согласно прилагаемым к приборам инструкций.

Необходимо соблюдать личную гигиену и чистоту рабочего места.

Перед началом работы:

* проверить исправность гибкого электрошнура, вилки, подводящих кабелей;
* проверить исправность электрической розетки, очистить прибор от пыли;
* во время работы аппаратуру необходимо установить на неподвижную подставку, проверить наличие напряжения;
* не допускать к работе с электронной аппаратурой посторонних лиц;
* в случае возгорания, короткого замыкания отключить сухими руками прибор из электросети, сообщить об этом администрации;
* о случаях травматизма сообщить администрации, при пожаре сообщить администрации и в пожарную службу, принять меры к его тушению.

По окончанию работы:

1. отключить прибор от электросети, не дергать за электрошнур;
2. произвести очистку прибора чистой тканью;
3. не допускать падения приборов электронной аппаратуры;
4. не допускать воздействия на подводящие кабели, электрошнуры горячих жидкостей, падения тяжёлых предметов;
5. обо всех недостатках, отмеченных в работе электронных приборов, сообщить администрации.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения выпускной квалификационной работы была разработана информационная система книжного магазина “Дом книги”. В процессе разработки были выполнены поставленные задачи по реализации программного обеспечения и модулей, входящих в него.

Программное обеспечение работает по четкому и слаженному алгоритму, способно 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.

Разработаны следующие модули системы:

* + подсистема управления заказами;
  + подсистема управления закупками книг;
  + подсистема учета книг на складе.

Были выполнены следующие задачи разработки программного обеспечения:

* + разработать базу данных, содержащую в себе все сведения необходимые для работы системы;
  + реализовать модули управления заказами, управления закупками книг, учета книг на складе;
  + сформировать отчеты по деятельности каждого модуля;
  + провести проверку корректности работы каждого модуля.

Результатом проведенной работы является чёткий алгоритм, по которому работает программа. Также в процессе работы была проведена оценка экономических затрат на разработку программного обеспечения и определена его полная себестоимость с учетом всех основных и косвенных затрат, возникших в процессе разработки.

Были даны основные рекомендации по технике безопасности при работе на производстве с ЭВМ.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Прохоренок Н.А., Дронов В.А. Python 3 и PyQt 5. Разработка приложений. — 2-е изд.— СПб.: БХВ-Петербург, 2018. — 832 с.
2. Лутц М. Изучаем Python / пер. с англ. — 3‑е изд. — СПб.: Символ‑Плюс, 2009. — 848 с.
3. Любанович Б. Простой Python. Современный стиль программирования. — 2-е изд. — СПб.: Питер, 2021. — 592 с.
4. Свейгарт Э. Python. Чистый код для продолжающих. — СПб.: Питер, 2022. — 384 с.
5. Хеллман Д. Стандартная библиотека Python 3: справочник с примерами / пер. с англ. — 2-е изд. — СПб.: ООО “Диалектика”, 2019. — 1376 с.
6. Свейгарт Э. Автоматизация рутинных задач с помощью Python / пер. с англ. — 2-е изд.— СПб.: ООО “Диалектика”, 2021. — 672 с.
7. Такфилд Б. Алгоритмы неформально. Инструкция для начинающих питонистов. — СПб.: Питер, 2022. — 272 с.
8. Ахмал И. 40 алгоритмов, которые должен знать каждый программист на Python. — СПб.: Питер, 2023. — 368 с.
9. Назина О. Что такое тестирование. Курс молодого бойца. — СПб.: БХВ-Петербург, 2022. - 592 с.: ил.
10. Хориков В. Принципы юнит-тестирования. — СПб.: Питер, 2021. — 320 с.
11. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001 — 368 с.
12. Персиваль Г, Грегори Б. Паттерны разработки на Python: TDD, DDD и событийно-ориентированная архитектура. — СПб.: Питер, 2022. — 336 с.
13. Федорова Г.Н. Основы проектирования баз данных. — М.: Академия, 2017. — 218 с.
14. Дюбуа П. MySQL. Сборник рецептов / пер. с англ. — СПб: Символ-Плюс, 2006. — 1056 с.
15. Шварц Б., Зайцев П., Ткаченко В. MySQL по максимуму. — 3-е изд. — СПб.: Питер, 2018. — 864 с.

# **ПРИЛОЖЕНИЕ А Иллюстрации форм**

Форма просмотра истории сотрудников представлена на рисунке А.1.

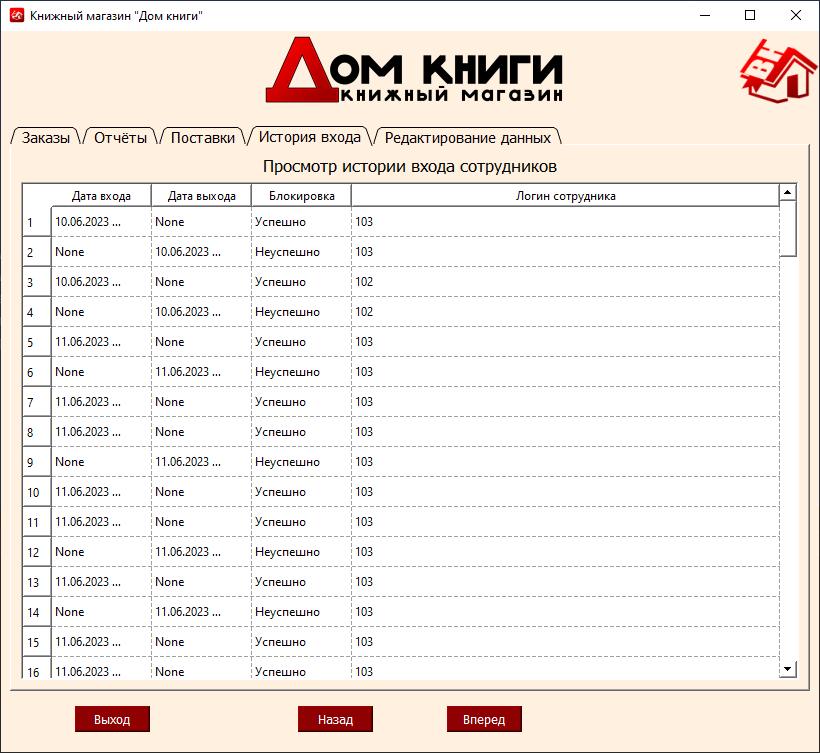


Рисунок А.1 — Форма просмотра истории входа сотрудников

Форма просмотра книг представлена на рисунке А.2.

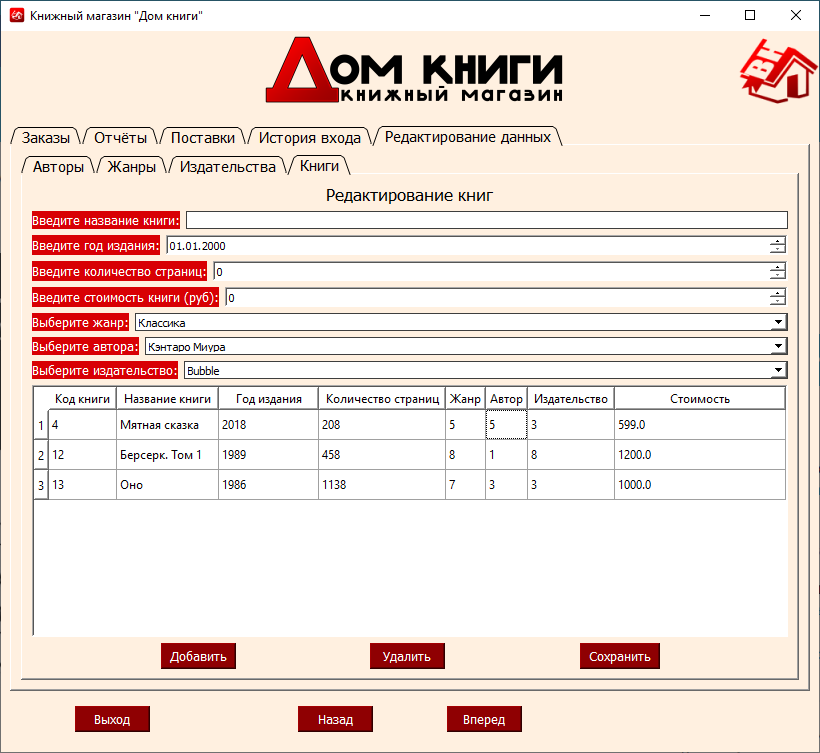


Рисунок А.2 — Форма просмотра книг

Форма просмотра жанров представлена на рисунке А.3.

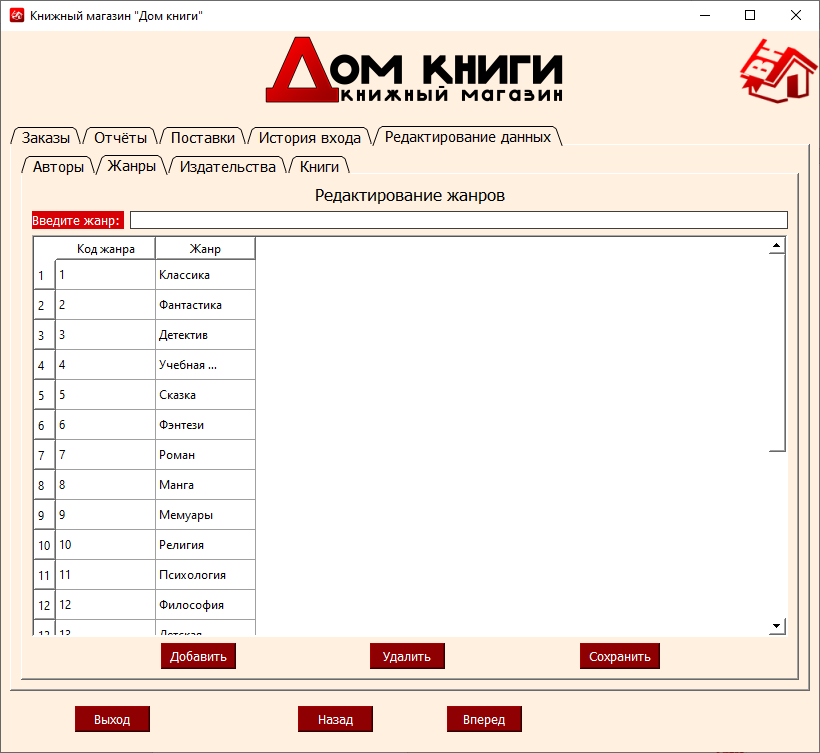


Рисунок А.3 — Форма просмотра жанров

Форма просмотра авторов представлена на рисунке А.4.

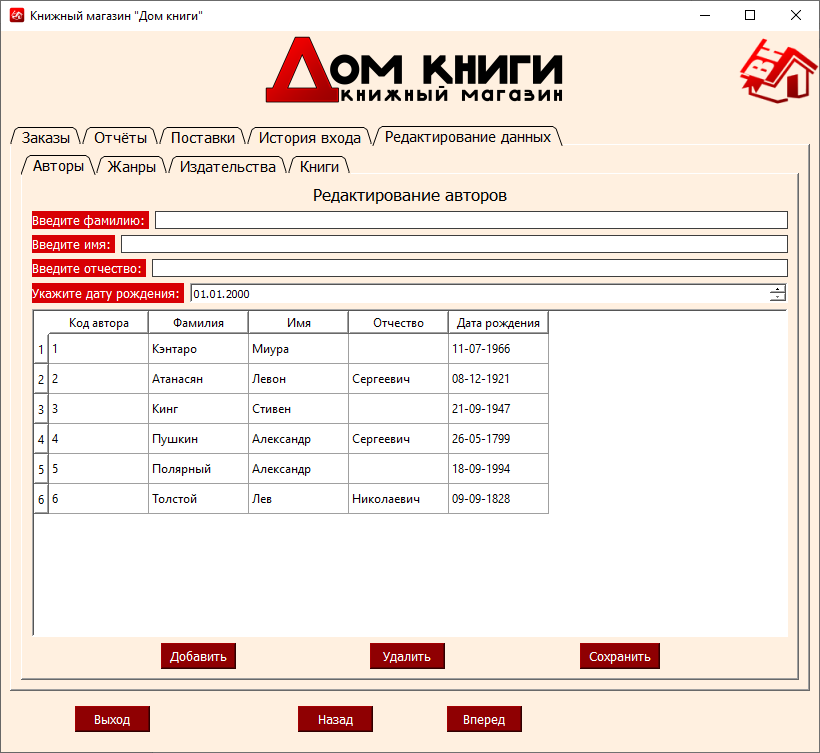


Рисунок А.4 — Форма просмотра авторов

Форма просмотра издательств представлена на рисунке А.5.

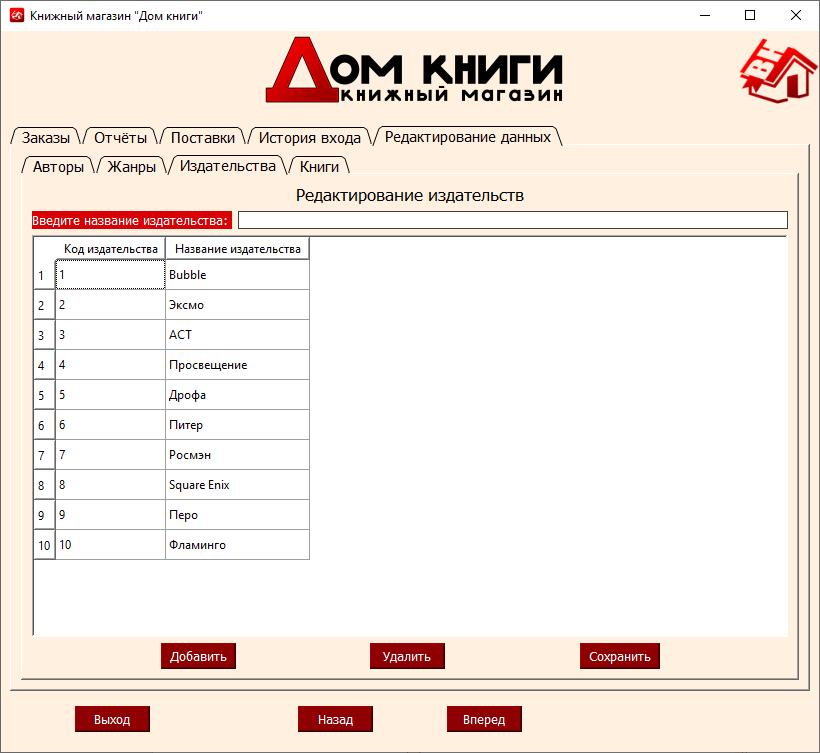


Рисунок А.5 — Форма просмотра издательств

Форма создания заказа представлена на рисунке А.6.

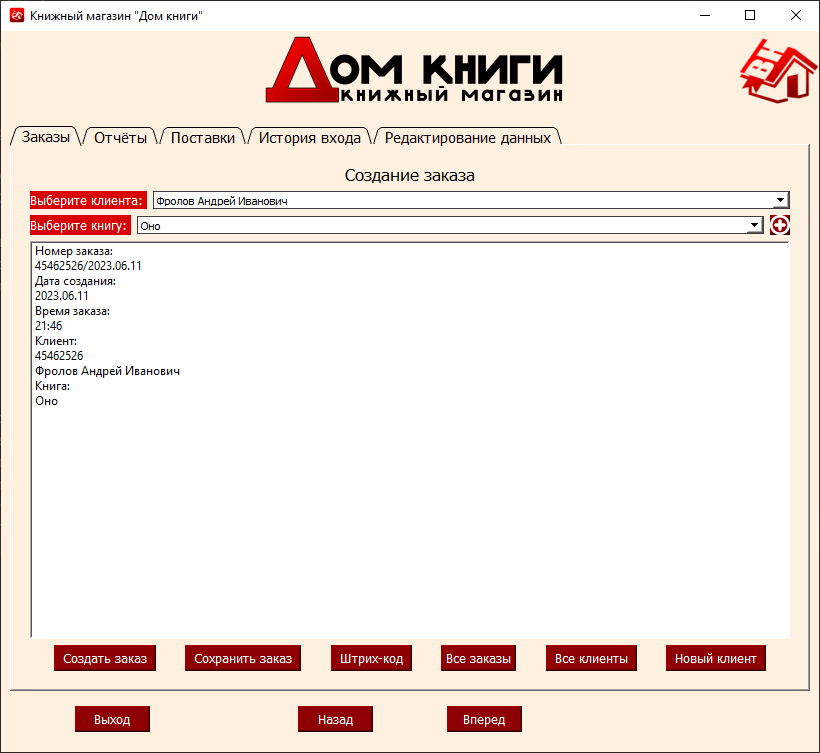


Рисунок А.6 — Форма создания заказа

Форма создания отчётов представлена на рисунке А.7.

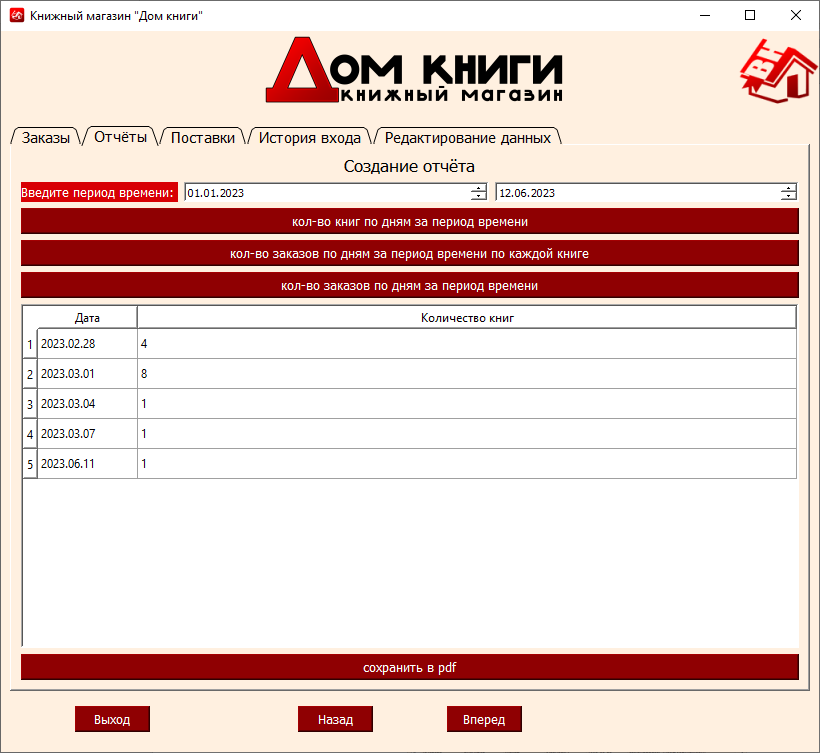


Рисунок А.7 — Форма создания отчётов

Форма просмотра клиентов представлена на рисунке А.8.

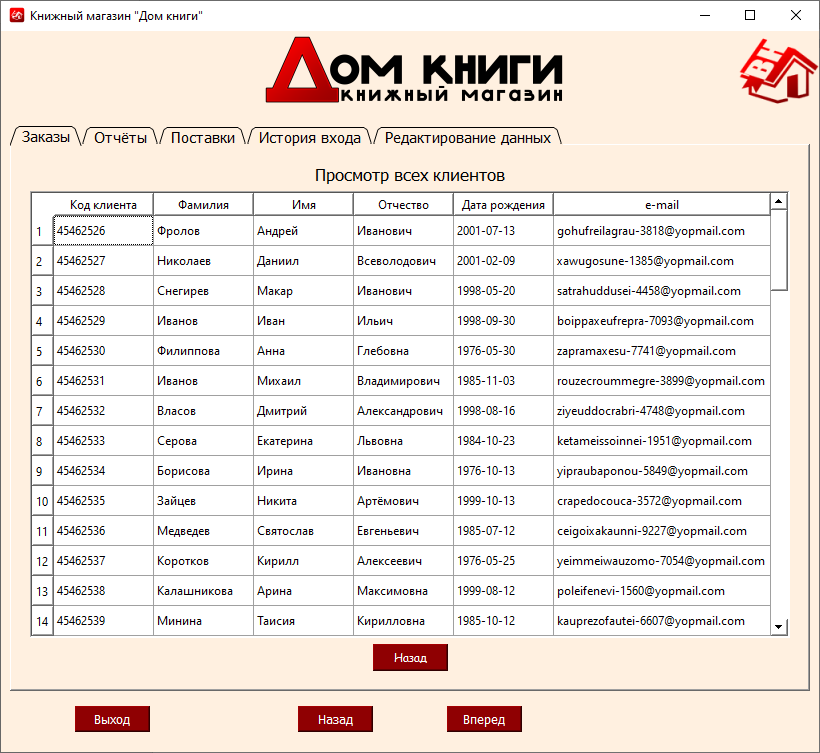


Рисунок А.8 — Форма просмотра клиентов

Форма добавления нового клиента представлена на рисунке А.9.

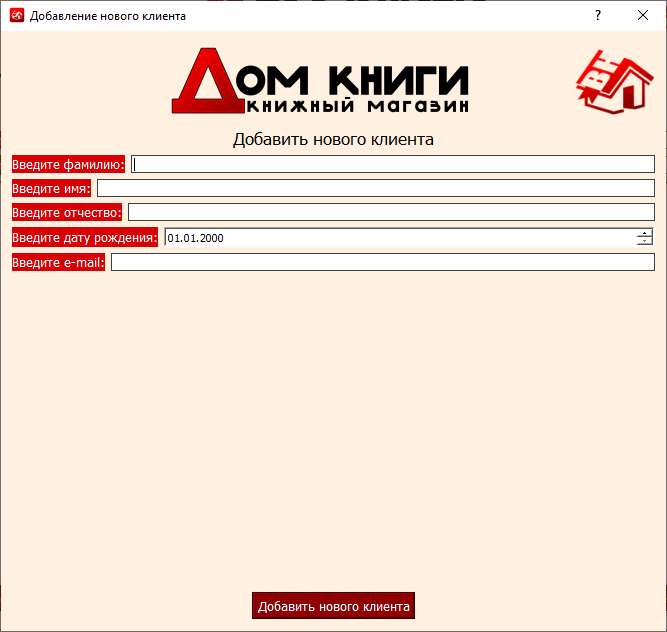


Рисунок А.9 — Форма добавления нового клиента

Форма добавления нового поставщика представлена на рисунке А.10.

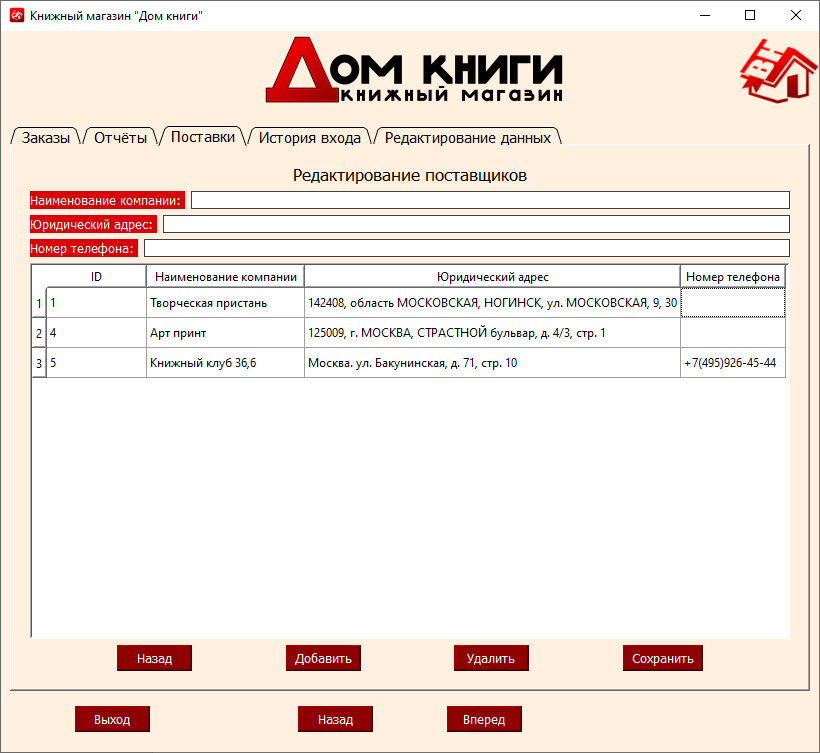


Рисунок А.10 — Форма добавления нового поставщика

Форма добавления новой поставки представлена на рисунке А.11.

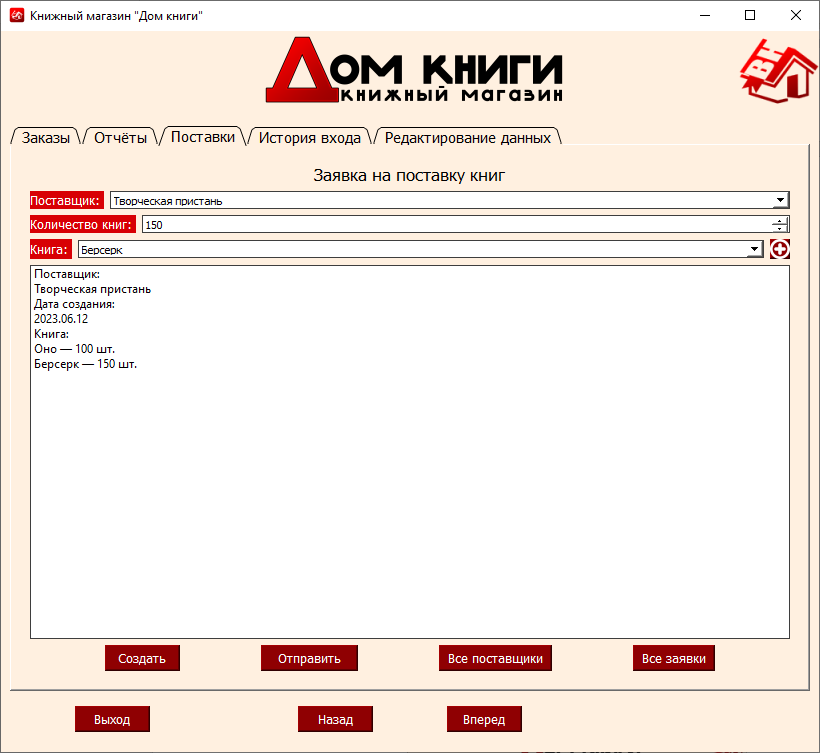


Рисунок А.11 — Форма добавления новой поставки

Форма просмотра заказов представлена на рисунке А.12.

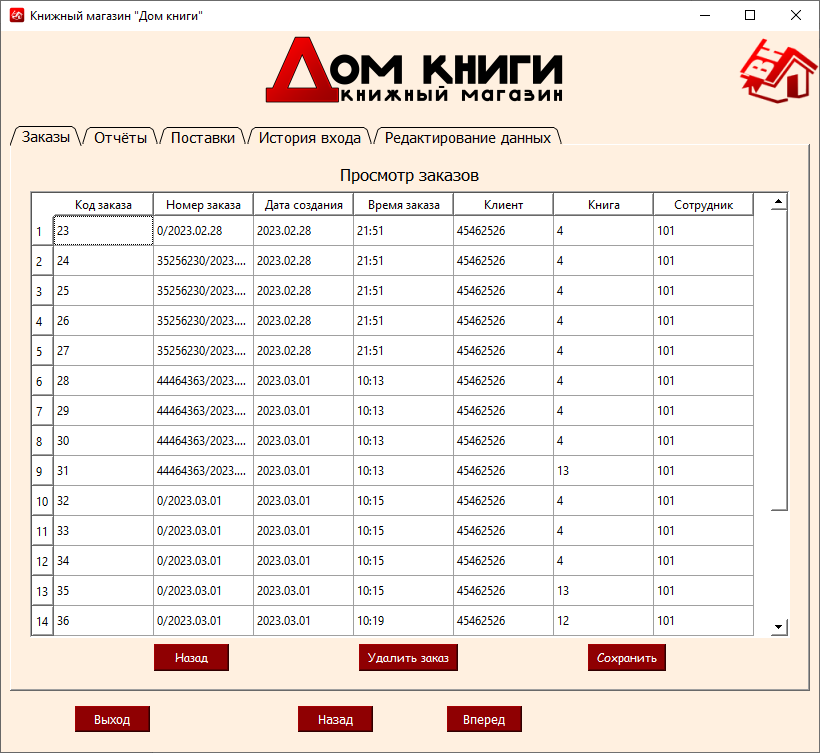


Рисунок А.12 — Форма просмотра заказов

Форма просмотра поставок представлена на рисунке А.13.

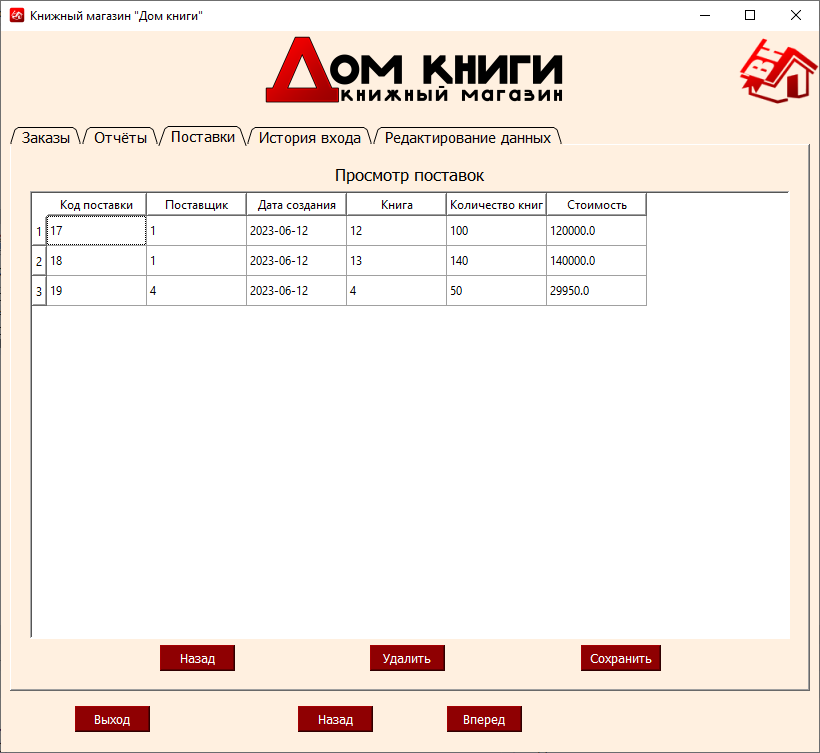


Рисунок А.13 — Форма просмотра поставок

# **ПРИЛОЖЕНИЕ Б Листинг форм в формате .py**

Auth.py

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_Form(object):

def setupUi(self, Form):

Form.setObjectName("Form")

Form.resize(427, 346)

Form.setMinimumSize(QtCore.QSize(427, 0))

Form.setMaximumSize(QtCore.QSize(427, 346))

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

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

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(Form)

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

self.label\_5.setMaximumSize(QtCore.QSize(290, 70))

font = QtGui.QFont()

font.setFamily("Rostov")

font.setPointSize(16)

self.label\_5.setFont(font)

self.label\_5.setText("")

self.label\_5.setTextFormat(QtCore.Qt.AutoText)

self.label\_5.setPixmap(QtGui.QPixmap("../лого.png"))

self.label\_5.setScaledContents(True)

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

self.label\_5.setWordWrap(False)

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(Form)

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)

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

self.horizontalLayout\_3.addItem(spacerItem2)

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

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

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

self.label = QtWidgets.QLabel(Form)

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

sizePolicy.setHorizontalStretch(43)

sizePolicy.setVerticalStretch(20)

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

self.label.setSizePolicy(sizePolicy)

self.label.setMaximumSize(QtCore.QSize(430, 20))

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(13)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label.setFont(font)

self.label.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

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

self.label.setObjectName("label")

self.verticalLayout.addWidget(self.label)

self.line = QtWidgets.QFrame(Form)

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(Form)

self.label\_2.setMinimumSize(QtCore.QSize(56, 24))

self.label\_2.setMaximumSize(QtCore.QSize(0, 24))

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

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

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

self.edit\_login.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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(Form)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

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

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

self.edit\_password.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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(Form)

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/eye2.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

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

self.btn\_hide\_password.setAutoRepeat(False)

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(Form)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

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

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

self.edit\_captcha.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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

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

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

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.captcha = QtWidgets.QLabel(Form)

self.captcha.setMinimumSize(QtCore.QSize(407, 67))

self.captcha.setMaximumSize(QtCore.QSize(407, 67))

self.captcha.setFrameShape(QtWidgets.QFrame.Box)

self.captcha.setScaledContents(True)

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

self.captcha.setObjectName("captcha")

self.verticalLayout.addWidget(self.captcha)

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

self.verticalLayout.addItem(spacerItem3)

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

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

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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(Form)

QtCore.QMetaObject.connectSlotsByName(Form)

def retranslateUi(self, Form):

\_translate = QtCore.QCoreApplication.translate

Form.setWindowTitle(\_translate("Form", "Авторизация"))

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

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

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

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

self.captcha.setText(\_translate("Form", "капча"))

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

main.py:

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_BookShop(object):

def setupUi(self, BookShop):

BookShop.setObjectName("BookShop")

BookShop.resize(818, 721)

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(300, 70))

self.label.setMaximumSize(QtCore.QSize(300, 70))

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(16)

self.label.setFont(font)

self.label.setText("")

self.label.setPixmap(QtGui.QPixmap("../лого.png"))

self.label.setScaledContents(True)

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("MS Shell Dlg 2")

font.setPointSize(13)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_3.setFont(font)

self.label\_3.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

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("MS Shell Dlg 2")

font.setPointSize(11)

font.setBold(False)

font.setItalic(False)

font.setWeight(9)

self.label\_4.setFont(font)

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

"font: 75 11pt \"MS Shell Dlg 2\";\n"

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

"")

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("MS Shell Dlg 2")

font.setPointSize(10)

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: 10pt \"MS Shell Dlg 2\";")

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("MS Shell Dlg 2")

font.setPointSize(11)

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: 11pt \"MS Shell Dlg 2\";")

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("MS Shell Dlg 2")

font.setPointSize(10)

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: 10pt \"MS Shell Dlg 2\";")

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.setMinimumSize(QtCore.QSize(393, 393))

self.lbl\_photo.setMaximumSize(QtCore.QSize(393, 393))

self.lbl\_photo.setText("")

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

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

self.lbl\_photo.setScaledContents(False)

self.lbl\_photo.setAlignment(QtCore.Qt.AlignCenter)

self.lbl\_photo.setWordWrap(False)

self.lbl\_photo.setTextInteractionFlags(QtCore.Qt.NoTextInteraction)

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(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 = QtWidgets.QWidget()

self.page.setObjectName("page")

self.verticalLayout\_6 = QtWidgets.QVBoxLayout(self.page)

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

self.tabWidget = QtWidgets.QTabWidget(self.page)

palette = QtGui.QPalette()

brush = QtGui.QBrush(QtGui.QColor(255, 240, 224))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 0, 0))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 240, 224))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 240, 224))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 240, 224))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 0, 0))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 240, 224))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 240, 224))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 240, 224))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 0, 0))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 240, 224))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 240, 224))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)

self.tabWidget.setPalette(palette)

font = QtGui.QFont()

font.setPointSize(11)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.tabWidget.setFont(font)

self.tabWidget.setMouseTracking(False)

self.tabWidget.setTabletTracking(False)

self.tabWidget.setFocusPolicy(QtCore.Qt.TabFocus)

self.tabWidget.setAcceptDrops(False)

self.tabWidget.setToolTipDuration(-1)

self.tabWidget.setAutoFillBackground(False)

self.tabWidget.setStyleSheet("")

self.tabWidget.setTabShape(QtWidgets.QTabWidget.Triangular)

self.tabWidget.setElideMode(QtCore.Qt.ElideNone)

self.tabWidget.setDocumentMode(False)

self.tabWidget.setTabsClosable(False)

self.tabWidget.setMovable(False)

self.tabWidget.setTabBarAutoHide(False)

self.tabWidget.setObjectName("tabWidget")

self.tab\_4 = QtWidgets.QWidget()

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

self.verticalLayout\_9 = QtWidgets.QVBoxLayout(self.tab\_4)

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

self.stackedWidget\_req = QtWidgets.QStackedWidget(self.tab\_4)

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

self.page\_6 = QtWidgets.QWidget()

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

self.verticalLayout\_23 = QtWidgets.QVBoxLayout(self.page\_6)

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

self.verticalLayout\_8 = QtWidgets.QVBoxLayout()

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

self.label\_10 = QtWidgets.QLabel(self.page\_6)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(13)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_10.setFont(font)

self.label\_10.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

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

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

self.verticalLayout\_8.addWidget(self.label\_10)

self.horizontalLayout\_13 = QtWidgets.QHBoxLayout()

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

self.label\_11 = QtWidgets.QLabel(self.page\_6)

self.label\_11.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_13.addWidget(self.label\_11)

self.comboBox\_clients = QtWidgets.QComboBox(self.page\_6)

self.comboBox\_clients.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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

self.horizontalLayout\_13.addWidget(self.comboBox\_clients)

self.horizontalLayout\_13.setStretch(0, 1)

self.horizontalLayout\_13.setStretch(1, 10)

self.verticalLayout\_8.addLayout(self.horizontalLayout\_13)

self.horizontalLayout\_14 = QtWidgets.QHBoxLayout()

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

self.label\_12 = QtWidgets.QLabel(self.page\_6)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_14.addWidget(self.label\_12)

self.comboBox\_book = QtWidgets.QComboBox(self.page\_6)

self.comboBox\_book.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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

self.horizontalLayout\_14.addWidget(self.comboBox\_book)

self.btn\_plus = QtWidgets.QPushButton(self.page\_6)

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\_14.addWidget(self.btn\_plus)

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

self.horizontalLayout\_14.setStretch(1, 100)

self.verticalLayout\_8.addLayout(self.horizontalLayout\_14)

self.add\_new\_field = QtWidgets.QListWidget(self.page\_6)

self.add\_new\_field.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"")

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\_8.addWidget(self.add\_new\_field)

self.horizontalLayout\_15 = QtWidgets.QHBoxLayout()

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

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

self.horizontalLayout\_15.addItem(spacerItem4)

self.btn\_new\_order = QtWidgets.QPushButton(self.page\_6)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_15.addWidget(self.btn\_new\_order)

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

self.horizontalLayout\_15.addItem(spacerItem5)

self.btn\_save\_request = QtWidgets.QPushButton(self.page\_6)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_15.addWidget(self.btn\_save\_request)

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

self.horizontalLayout\_15.addItem(spacerItem6)

self.btn\_code = QtWidgets.QPushButton(self.page\_6)

self.btn\_code.setEnabled(True)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_15.addWidget(self.btn\_code)

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

self.horizontalLayout\_15.addItem(spacerItem7)

self.btn\_all\_req = QtWidgets.QPushButton(self.page\_6)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_15.addWidget(self.btn\_all\_req)

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

self.horizontalLayout\_15.addItem(spacerItem8)

self.btn\_all\_clients = QtWidgets.QPushButton(self.page\_6)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_15.addWidget(self.btn\_all\_clients)

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

self.horizontalLayout\_15.addItem(spacerItem9)

self.btn\_new\_client = QtWidgets.QPushButton(self.page\_6)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_15.addWidget(self.btn\_new\_client)

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

self.horizontalLayout\_15.addItem(spacerItem10)

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

self.verticalLayout\_23.addLayout(self.verticalLayout\_8)

self.stackedWidget\_req.addWidget(self.page\_6)

self.page\_9 = QtWidgets.QWidget()

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

self.verticalLayout\_31 = QtWidgets.QVBoxLayout(self.page\_9)

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

self.verticalLayout\_30 = QtWidgets.QVBoxLayout()

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

self.label\_46 = QtWidgets.QLabel(self.page\_9)

self.label\_46.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_46.setAlignment(QtCore.Qt.AlignCenter)

self.label\_46.setObjectName("label\_46")

self.verticalLayout\_30.addWidget(self.label\_46)

self.verticalLayout\_31.addLayout(self.verticalLayout\_30)

self.table\_orders = QtWidgets.QTableWidget(self.page\_9)

self.table\_orders.setStyleSheet("background-color: rgb(255, 255, 255); ")

self.table\_orders.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

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

self.table\_orders.setColumnCount(0)

self.table\_orders.setRowCount(0)

self.verticalLayout\_31.addWidget(self.table\_orders)

self.horizontalLayout\_56 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_56.setObjectName("horizontalLayout\_56")

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

self.horizontalLayout\_56.addItem(spacerItem11)

self.btn\_back\_ord = QtWidgets.QPushButton(self.page\_9)

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

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

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

self.btn\_back\_ord.setObjectName("btn\_back\_ord")

self.horizontalLayout\_56.addWidget(self.btn\_back\_ord)

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

self.horizontalLayout\_56.addItem(spacerItem12)

self.btn\_del\_order = QtWidgets.QPushButton(self.page\_9)

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

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

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

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

self.horizontalLayout\_56.addWidget(self.btn\_del\_order)

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

self.horizontalLayout\_56.addItem(spacerItem13)

self.btn\_save\_order = QtWidgets.QPushButton(self.page\_9)

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

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

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

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

self.horizontalLayout\_56.addWidget(self.btn\_save\_order)

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

self.horizontalLayout\_56.addItem(spacerItem14)

self.verticalLayout\_31.addLayout(self.horizontalLayout\_56)

self.stackedWidget\_req.addWidget(self.page\_9)

self.page\_7 = QtWidgets.QWidget()

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

self.verticalLayout\_32 = QtWidgets.QVBoxLayout(self.page\_7)

self.verticalLayout\_32.setObjectName("verticalLayout\_32")

self.label\_47 = QtWidgets.QLabel(self.page\_7)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(13)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_47.setFont(font)

self.label\_47.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_47.setAlignment(QtCore.Qt.AlignCenter)

self.label\_47.setObjectName("label\_47")

self.verticalLayout\_32.addWidget(self.label\_47)

self.table\_clients = QtWidgets.QTableWidget(self.page\_7)

self.table\_clients.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"")

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\_32.addWidget(self.table\_clients)

self.horizontalLayout\_5 = QtWidgets.QHBoxLayout()

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

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

self.horizontalLayout\_5.addItem(spacerItem15)

self.btn\_back\_ord\_2 = QtWidgets.QPushButton(self.page\_7)

self.btn\_back\_ord\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

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

self.btn\_back\_ord\_2.setObjectName("btn\_back\_ord\_2")

self.horizontalLayout\_5.addWidget(self.btn\_back\_ord\_2)

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

self.horizontalLayout\_5.addItem(spacerItem16)

self.verticalLayout\_32.addLayout(self.horizontalLayout\_5)

self.stackedWidget\_req.addWidget(self.page\_7)

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

self.tabWidget.addTab(self.tab\_4, "")

self.tab\_5 = QtWidgets.QWidget()

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

self.verticalLayout\_10 = QtWidgets.QVBoxLayout(self.tab\_5)

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

self.label\_13 = QtWidgets.QLabel(self.tab\_5)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(13)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_13.setFont(font)

self.label\_13.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_13.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_10.addWidget(self.label\_13)

self.horizontalLayout\_19 = QtWidgets.QHBoxLayout()

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

self.label\_15 = QtWidgets.QLabel(self.tab\_5)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_19.addWidget(self.label\_15)

self.date\_start = QtWidgets.QDateTimeEdit(self.tab\_5)

font = QtGui.QFont()

font.setPointSize(8)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.date\_start.setFont(font)

self.date\_start.setStyleSheet("background-color: rgb(255, 255, 255);")

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

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

self.horizontalLayout\_19.addWidget(self.date\_start)

self.date\_end = QtWidgets.QDateTimeEdit(self.tab\_5)

self.date\_end.setStyleSheet("background-color: rgb(255, 255, 255);")

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

self.horizontalLayout\_19.addWidget(self.date\_end)

self.horizontalLayout\_19.setStretch(0, 1)

self.horizontalLayout\_19.setStretch(1, 2)

self.horizontalLayout\_19.setStretch(2, 2)

self.verticalLayout\_10.addLayout(self.horizontalLayout\_19)

self.btn\_count\_book = QtWidgets.QPushButton(self.tab\_5)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.verticalLayout\_10.addWidget(self.btn\_count\_book)

self.btn\_count\_order\_book = QtWidgets.QPushButton(self.tab\_5)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.verticalLayout\_10.addWidget(self.btn\_count\_order\_book)

self.btn\_count\_order = QtWidgets.QPushButton(self.tab\_5)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.verticalLayout\_10.addWidget(self.btn\_count\_order)

self.table\_graf = QtWidgets.QTableWidget(self.tab\_5)

self.table\_graf.setStyleSheet("background-color: rgb(255, 255, 255);")

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\_10.addWidget(self.table\_graf)

self.horizontalLayout\_18 = QtWidgets.QHBoxLayout()

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

self.btn\_pdf = QtWidgets.QPushButton(self.tab\_5)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_18.addWidget(self.btn\_pdf)

self.verticalLayout\_10.addLayout(self.horizontalLayout\_18)

self.tabWidget.addTab(self.tab\_5, "")

self.tab\_11 = QtWidgets.QWidget()

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

self.verticalLayout\_19 = QtWidgets.QVBoxLayout(self.tab\_11)

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

self.stackedWidget\_prov = QtWidgets.QStackedWidget(self.tab\_11)

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

self.page\_4 = QtWidgets.QWidget()

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

self.verticalLayout\_22 = QtWidgets.QVBoxLayout(self.page\_4)

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

self.verticalLayout\_21 = QtWidgets.QVBoxLayout()

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

self.horizontalLayout\_45 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_45.setObjectName("horizontalLayout\_45")

self.label\_37 = QtWidgets.QLabel(self.page\_4)

self.label\_37.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_37.setAlignment(QtCore.Qt.AlignCenter)

self.label\_37.setObjectName("label\_37")

self.horizontalLayout\_45.addWidget(self.label\_37)

self.verticalLayout\_21.addLayout(self.horizontalLayout\_45)

self.verticalLayout\_26 = QtWidgets.QVBoxLayout()

self.verticalLayout\_26.setObjectName("verticalLayout\_26")

self.horizontalLayout\_46 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_46.setObjectName("horizontalLayout\_46")

self.label\_38 = QtWidgets.QLabel(self.page\_4)

self.label\_38.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_38.setObjectName("label\_38")

self.horizontalLayout\_46.addWidget(self.label\_38)

self.comboProvider = QtWidgets.QComboBox(self.page\_4)

self.comboProvider.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

self.comboProvider.setObjectName("comboProvider")

self.horizontalLayout\_46.addWidget(self.comboProvider)

self.horizontalLayout\_46.setStretch(1, 100)

self.verticalLayout\_26.addLayout(self.horizontalLayout\_46)

self.horizontalLayout\_47 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_47.setObjectName("horizontalLayout\_47")

self.label\_39 = QtWidgets.QLabel(self.page\_4)

self.label\_39.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_39.setObjectName("label\_39")

self.horizontalLayout\_47.addWidget(self.label\_39)

self.spinBoxQuantityBook = QtWidgets.QSpinBox(self.page\_4)

self.spinBoxQuantityBook.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

self.spinBoxQuantityBook.setMaximum(999999999)

self.spinBoxQuantityBook.setObjectName("spinBoxQuantityBook")

self.horizontalLayout\_47.addWidget(self.spinBoxQuantityBook)

self.horizontalLayout\_47.setStretch(1, 100)

self.verticalLayout\_26.addLayout(self.horizontalLayout\_47)

self.verticalLayout\_21.addLayout(self.verticalLayout\_26)

self.horizontalLayout\_48 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_48.setObjectName("horizontalLayout\_48")

self.label\_40 = QtWidgets.QLabel(self.page\_4)

self.label\_40.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_40.setObjectName("label\_40")

self.horizontalLayout\_48.addWidget(self.label\_40)

self.comboBookProvider = QtWidgets.QComboBox(self.page\_4)

self.comboBookProvider.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

self.comboBookProvider.setObjectName("comboBookProvider")

self.horizontalLayout\_48.addWidget(self.comboBookProvider)

self.btn\_add\_book\_pr\_od = QtWidgets.QPushButton(self.page\_4)

self.btn\_add\_book\_pr\_od.setMaximumSize(QtCore.QSize(20, 20))

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

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

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

self.btn\_add\_book\_pr\_od.setText("")

self.btn\_add\_book\_pr\_od.setIcon(icon1)

self.btn\_add\_book\_pr\_od.setIconSize(QtCore.QSize(20, 20))

self.btn\_add\_book\_pr\_od.setObjectName("btn\_add\_book\_pr\_od")

self.horizontalLayout\_48.addWidget(self.btn\_add\_book\_pr\_od)

self.horizontalLayout\_48.setStretch(1, 100)

self.verticalLayout\_21.addLayout(self.horizontalLayout\_48)

self.listWidgetDelivery = QtWidgets.QListWidget(self.page\_4)

self.listWidgetDelivery.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

self.listWidgetDelivery.setObjectName("listWidgetDelivery")

self.verticalLayout\_21.addWidget(self.listWidgetDelivery)

self.horizontalLayout\_49 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_49.setObjectName("horizontalLayout\_49")

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

self.horizontalLayout\_49.addItem(spacerItem17)

self.btn\_create\_ord = QtWidgets.QPushButton(self.page\_4)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_create\_ord.setObjectName("btn\_create\_ord")

self.horizontalLayout\_49.addWidget(self.btn\_create\_ord)

spacerItem18 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_49.addItem(spacerItem18)

self.btn\_send = QtWidgets.QPushButton(self.page\_4)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_49.addWidget(self.btn\_send)

spacerItem19 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_49.addItem(spacerItem19)

self.btn\_all\_prov = QtWidgets.QPushButton(self.page\_4)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_49.addWidget(self.btn\_all\_prov)

spacerItem20 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_49.addItem(spacerItem20)

self.btn\_all\_pr\_od = QtWidgets.QPushButton(self.page\_4)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_all\_pr\_od.setObjectName("btn\_all\_pr\_od")

self.horizontalLayout\_49.addWidget(self.btn\_all\_pr\_od)

spacerItem21 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_49.addItem(spacerItem21)

self.verticalLayout\_21.addLayout(self.horizontalLayout\_49)

self.verticalLayout\_22.addLayout(self.verticalLayout\_21)

self.stackedWidget\_prov.addWidget(self.page\_4)

self.page\_5 = QtWidgets.QWidget()

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

self.verticalLayout\_24 = QtWidgets.QVBoxLayout(self.page\_5)

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

self.verticalLayout\_27 = QtWidgets.QVBoxLayout()

self.verticalLayout\_27.setObjectName("verticalLayout\_27")

self.label\_44 = QtWidgets.QLabel(self.page\_5)

self.label\_44.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_44.setAlignment(QtCore.Qt.AlignCenter)

self.label\_44.setObjectName("label\_44")

self.verticalLayout\_27.addWidget(self.label\_44)

self.verticalLayout\_24.addLayout(self.verticalLayout\_27)

self.verticalLayout\_28 = QtWidgets.QVBoxLayout()

self.verticalLayout\_28.setObjectName("verticalLayout\_28")

self.horizontalLayout\_51 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_51.setObjectName("horizontalLayout\_51")

self.label\_41 = QtWidgets.QLabel(self.page\_5)

self.label\_41.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_41.setObjectName("label\_41")

self.horizontalLayout\_51.addWidget(self.label\_41)

self.edit\_name\_pr = QtWidgets.QLineEdit(self.page\_5)

self.edit\_name\_pr.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

self.edit\_name\_pr.setObjectName("edit\_name\_pr")

self.horizontalLayout\_51.addWidget(self.edit\_name\_pr)

self.verticalLayout\_28.addLayout(self.horizontalLayout\_51)

self.horizontalLayout\_52 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_52.setObjectName("horizontalLayout\_52")

self.label\_42 = QtWidgets.QLabel(self.page\_5)

self.label\_42.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_42.setObjectName("label\_42")

self.horizontalLayout\_52.addWidget(self.label\_42)

self.edit\_address\_pr = QtWidgets.QLineEdit(self.page\_5)

self.edit\_address\_pr.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

self.edit\_address\_pr.setObjectName("edit\_address\_pr")

self.horizontalLayout\_52.addWidget(self.edit\_address\_pr)

self.verticalLayout\_28.addLayout(self.horizontalLayout\_52)

self.horizontalLayout\_53 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_53.setObjectName("horizontalLayout\_53")

self.label\_43 = QtWidgets.QLabel(self.page\_5)

self.label\_43.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_43.setObjectName("label\_43")

self.horizontalLayout\_53.addWidget(self.label\_43)

self.edit\_phone\_pr = QtWidgets.QLineEdit(self.page\_5)

self.edit\_phone\_pr.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

self.edit\_phone\_pr.setObjectName("edit\_phone\_pr")

self.horizontalLayout\_53.addWidget(self.edit\_phone\_pr)

self.verticalLayout\_28.addLayout(self.horizontalLayout\_53)

self.verticalLayout\_24.addLayout(self.verticalLayout\_28)

self.table\_provider = QtWidgets.QTableWidget(self.page\_5)

self.table\_provider.setStyleSheet("background-color: rgb(255, 255, 255); ")

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

self.table\_provider.setColumnCount(0)

self.table\_provider.setRowCount(0)

self.verticalLayout\_24.addWidget(self.table\_provider)

self.horizontalLayout\_50 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_50.setObjectName("horizontalLayout\_50")

spacerItem22 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_50.addItem(spacerItem22)

self.btn\_back\_prov = QtWidgets.QPushButton(self.page\_5)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_back\_prov.setObjectName("btn\_back\_prov")

self.horizontalLayout\_50.addWidget(self.btn\_back\_prov)

spacerItem23 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_50.addItem(spacerItem23)

self.btn\_add\_prov = QtWidgets.QPushButton(self.page\_5)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_add\_prov.setObjectName("btn\_add\_prov")

self.horizontalLayout\_50.addWidget(self.btn\_add\_prov)

spacerItem24 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_50.addItem(spacerItem24)

self.btn\_del\_prov = QtWidgets.QPushButton(self.page\_5)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_del\_prov.setObjectName("btn\_del\_prov")

self.horizontalLayout\_50.addWidget(self.btn\_del\_prov)

spacerItem25 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_50.addItem(spacerItem25)

self.btn\_save\_prov = QtWidgets.QPushButton(self.page\_5)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_50.addWidget(self.btn\_save\_prov)

spacerItem26 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_50.addItem(spacerItem26)

self.verticalLayout\_24.addLayout(self.horizontalLayout\_50)

self.stackedWidget\_prov.addWidget(self.page\_5)

self.page\_8 = QtWidgets.QWidget()

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

self.verticalLayout\_29 = QtWidgets.QVBoxLayout(self.page\_8)

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

self.horizontalLayout\_55 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_55.setObjectName("horizontalLayout\_55")

self.label\_45 = QtWidgets.QLabel(self.page\_8)

self.label\_45.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_45.setAlignment(QtCore.Qt.AlignCenter)

self.label\_45.setObjectName("label\_45")

self.horizontalLayout\_55.addWidget(self.label\_45)

self.verticalLayout\_29.addLayout(self.horizontalLayout\_55)

self.table\_pr\_od = QtWidgets.QTableWidget(self.page\_8)

self.table\_pr\_od.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"")

self.table\_pr\_od.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.table\_pr\_od.setObjectName("table\_pr\_od")

self.table\_pr\_od.setColumnCount(0)

self.table\_pr\_od.setRowCount(0)

self.verticalLayout\_29.addWidget(self.table\_pr\_od)

self.horizontalLayout\_54 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_54.setObjectName("horizontalLayout\_54")

spacerItem27 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_54.addItem(spacerItem27)

self.btn\_back\_prov\_2 = QtWidgets.QPushButton(self.page\_8)

self.btn\_back\_prov\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_back\_prov\_2.setObjectName("btn\_back\_prov\_2")

self.horizontalLayout\_54.addWidget(self.btn\_back\_prov\_2)

spacerItem28 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_54.addItem(spacerItem28)

self.btn\_delete\_pr\_od = QtWidgets.QPushButton(self.page\_8)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_delete\_pr\_od.setObjectName("btn\_delete\_pr\_od")

self.horizontalLayout\_54.addWidget(self.btn\_delete\_pr\_od)

spacerItem29 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_54.addItem(spacerItem29)

self.btn\_save\_pr\_od = QtWidgets.QPushButton(self.page\_8)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_save\_pr\_od.setObjectName("btn\_save\_pr\_od")

self.horizontalLayout\_54.addWidget(self.btn\_save\_pr\_od)

spacerItem30 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_54.addItem(spacerItem30)

self.verticalLayout\_29.addLayout(self.horizontalLayout\_54)

self.stackedWidget\_prov.addWidget(self.page\_8)

self.verticalLayout\_19.addWidget(self.stackedWidget\_prov)

self.tabWidget.addTab(self.tab\_11, "")

self.tab\_3 = QtWidgets.QWidget()

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

self.verticalLayout\_7 = QtWidgets.QVBoxLayout(self.tab\_3)

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

self.horizontalLayout\_12 = QtWidgets.QHBoxLayout()

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

self.label\_9 = QtWidgets.QLabel(self.tab\_3)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(13)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_9.setFont(font)

self.label\_9.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_9.setAlignment(QtCore.Qt.AlignCenter)

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

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

self.verticalLayout\_7.addLayout(self.horizontalLayout\_12)

self.table\_entry = QtWidgets.QTableWidget(self.tab\_3)

self.table\_entry.setStatusTip("")

self.table\_entry.setStyleSheet("background-color: rgb(255, 255, 255);")

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.tabWidget.addTab(self.tab\_3, "")

self.tab\_6 = QtWidgets.QWidget()

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

self.verticalLayout\_11 = QtWidgets.QVBoxLayout(self.tab\_6)

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

self.tabWidget\_2 = QtWidgets.QTabWidget(self.tab\_6)

font = QtGui.QFont()

font.setPointSize(11)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.tabWidget\_2.setFont(font)

self.tabWidget\_2.setMouseTracking(False)

self.tabWidget\_2.setStyleSheet("")

self.tabWidget\_2.setInputMethodHints(QtCore.Qt.ImhNone)

self.tabWidget\_2.setTabPosition(QtWidgets.QTabWidget.North)

self.tabWidget\_2.setTabShape(QtWidgets.QTabWidget.Triangular)

self.tabWidget\_2.setElideMode(QtCore.Qt.ElideNone)

self.tabWidget\_2.setUsesScrollButtons(True)

self.tabWidget\_2.setDocumentMode(False)

self.tabWidget\_2.setTabsClosable(False)

self.tabWidget\_2.setMovable(False)

self.tabWidget\_2.setTabBarAutoHide(False)

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

self.tab\_9 = QtWidgets.QWidget()

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

self.verticalLayout\_15 = QtWidgets.QVBoxLayout(self.tab\_9)

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

self.horizontalLayout\_33 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_33.setObjectName("horizontalLayout\_33")

self.label\_25 = QtWidgets.QLabel(self.tab\_9)

self.label\_25.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_25.setAlignment(QtCore.Qt.AlignCenter)

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

self.horizontalLayout\_33.addWidget(self.label\_25)

self.verticalLayout\_15.addLayout(self.horizontalLayout\_33)

self.verticalLayout\_17 = QtWidgets.QVBoxLayout()

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

self.horizontalLayout\_34 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_34.setObjectName("horizontalLayout\_34")

self.label\_26 = QtWidgets.QLabel(self.tab\_9)

self.label\_26.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_26.setObjectName("label\_26")

self.horizontalLayout\_34.addWidget(self.label\_26)

self.edit\_f = QtWidgets.QLineEdit(self.tab\_9)

self.edit\_f.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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

self.horizontalLayout\_34.addWidget(self.edit\_f)

self.horizontalLayout\_34.setStretch(1, 100)

self.verticalLayout\_17.addLayout(self.horizontalLayout\_34)

self.horizontalLayout\_35 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_35.setObjectName("horizontalLayout\_35")

self.label\_27 = QtWidgets.QLabel(self.tab\_9)

self.label\_27.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_27.setObjectName("label\_27")

self.horizontalLayout\_35.addWidget(self.label\_27)

self.edit\_i = QtWidgets.QLineEdit(self.tab\_9)

self.edit\_i.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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

self.horizontalLayout\_35.addWidget(self.edit\_i)

self.horizontalLayout\_35.setStretch(1, 100)

self.verticalLayout\_17.addLayout(self.horizontalLayout\_35)

self.horizontalLayout\_36 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_36.setObjectName("horizontalLayout\_36")

self.label\_28 = QtWidgets.QLabel(self.tab\_9)

self.label\_28.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_28.setObjectName("label\_28")

self.horizontalLayout\_36.addWidget(self.label\_28)

self.edit\_o = QtWidgets.QLineEdit(self.tab\_9)

self.edit\_o.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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

self.horizontalLayout\_36.addWidget(self.edit\_o)

self.horizontalLayout\_36.setStretch(1, 100)

self.verticalLayout\_17.addLayout(self.horizontalLayout\_36)

self.horizontalLayout\_37 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_37.setObjectName("horizontalLayout\_37")

self.label\_30 = QtWidgets.QLabel(self.tab\_9)

self.label\_30.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_37.addWidget(self.label\_30)

self.date\_author = QtWidgets.QDateEdit(self.tab\_9)

self.date\_author.setStyleSheet("background-color: rgb(255, 255, 255);")

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

self.horizontalLayout\_37.addWidget(self.date\_author)

self.horizontalLayout\_37.setStretch(1, 100)

self.verticalLayout\_17.addLayout(self.horizontalLayout\_37)

self.verticalLayout\_15.addLayout(self.verticalLayout\_17)

self.table\_author = QtWidgets.QTableWidget(self.tab\_9)

self.table\_author.setStyleSheet("background-color: rgb(255, 255, 255);")

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

self.table\_author.setColumnCount(0)

self.table\_author.setRowCount(0)

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

self.horizontalLayout\_32 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_32.setObjectName("horizontalLayout\_32")

spacerItem31 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_32.addItem(spacerItem31)

self.btn\_new\_author = QtWidgets.QPushButton(self.tab\_9)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_32.addWidget(self.btn\_new\_author)

spacerItem32 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_32.addItem(spacerItem32)

self.btn\_delete\_author = QtWidgets.QPushButton(self.tab\_9)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_32.addWidget(self.btn\_delete\_author)

spacerItem33 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_32.addItem(spacerItem33)

self.btn\_save\_author = QtWidgets.QPushButton(self.tab\_9)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_32.addWidget(self.btn\_save\_author)

spacerItem34 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_32.addItem(spacerItem34)

self.verticalLayout\_15.addLayout(self.horizontalLayout\_32)

self.tabWidget\_2.addTab(self.tab\_9, "")

self.tab\_8 = QtWidgets.QWidget()

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

self.verticalLayout\_14 = QtWidgets.QVBoxLayout(self.tab\_8)

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

self.horizontalLayout\_30 = QtWidgets.QHBoxLayout()

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

self.label\_23 = QtWidgets.QLabel(self.tab\_8)

self.label\_23.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_23.setAlignment(QtCore.Qt.AlignCenter)

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

self.horizontalLayout\_30.addWidget(self.label\_23)

self.verticalLayout\_14.addLayout(self.horizontalLayout\_30)

self.horizontalLayout\_31 = QtWidgets.QHBoxLayout()

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

self.label\_24 = QtWidgets.QLabel(self.tab\_8)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_31.addWidget(self.label\_24)

self.edit\_genre = QtWidgets.QLineEdit(self.tab\_8)

self.edit\_genre.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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

self.horizontalLayout\_31.addWidget(self.edit\_genre)

self.verticalLayout\_14.addLayout(self.horizontalLayout\_31)

self.table\_genre = QtWidgets.QTableWidget(self.tab\_8)

self.table\_genre.setStyleSheet("background-color: rgb(255, 255, 255);")

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

self.table\_genre.setColumnCount(0)

self.table\_genre.setRowCount(0)

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

self.horizontalLayout\_29 = QtWidgets.QHBoxLayout()

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

spacerItem35 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_29.addItem(spacerItem35)

self.btn\_new\_genre = QtWidgets.QPushButton(self.tab\_8)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_29.addWidget(self.btn\_new\_genre)

spacerItem36 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_29.addItem(spacerItem36)

self.btn\_delete\_genre = QtWidgets.QPushButton(self.tab\_8)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_29.addWidget(self.btn\_delete\_genre)

spacerItem37 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_29.addItem(spacerItem37)

self.btn\_save\_genre = QtWidgets.QPushButton(self.tab\_8)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_29.addWidget(self.btn\_save\_genre)

spacerItem38 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_29.addItem(spacerItem38)

self.verticalLayout\_14.addLayout(self.horizontalLayout\_29)

self.tabWidget\_2.addTab(self.tab\_8, "")

self.tab\_10 = QtWidgets.QWidget()

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

self.verticalLayout\_16 = QtWidgets.QVBoxLayout(self.tab\_10)

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

self.verticalLayout\_18 = QtWidgets.QVBoxLayout()

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

self.label\_32 = QtWidgets.QLabel(self.tab\_10)

self.label\_32.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_32.setAlignment(QtCore.Qt.AlignCenter)

self.label\_32.setObjectName("label\_32")

self.verticalLayout\_18.addWidget(self.label\_32)

self.verticalLayout\_16.addLayout(self.verticalLayout\_18)

self.horizontalLayout\_39 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_39.setObjectName("horizontalLayout\_39")

self.label\_31 = QtWidgets.QLabel(self.tab\_10)

self.label\_31.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_39.addWidget(self.label\_31)

self.edit\_ph = QtWidgets.QLineEdit(self.tab\_10)

self.edit\_ph.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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

self.horizontalLayout\_39.addWidget(self.edit\_ph)

self.verticalLayout\_16.addLayout(self.horizontalLayout\_39)

self.table\_ph = QtWidgets.QTableWidget(self.tab\_10)

self.table\_ph.setStyleSheet("background-color: rgb(255, 255, 255);")

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

self.table\_ph.setColumnCount(0)

self.table\_ph.setRowCount(0)

self.verticalLayout\_16.addWidget(self.table\_ph)

self.horizontalLayout\_38 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_38.setObjectName("horizontalLayout\_38")

spacerItem39 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_38.addItem(spacerItem39)

self.btn\_new\_ph = QtWidgets.QPushButton(self.tab\_10)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_38.addWidget(self.btn\_new\_ph)

spacerItem40 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_38.addItem(spacerItem40)

self.btn\_delete\_ph = QtWidgets.QPushButton(self.tab\_10)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_38.addWidget(self.btn\_delete\_ph)

spacerItem41 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_38.addItem(spacerItem41)

self.btn\_save\_ph = QtWidgets.QPushButton(self.tab\_10)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_38.addWidget(self.btn\_save\_ph)

spacerItem42 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_38.addItem(spacerItem42)

self.verticalLayout\_16.addLayout(self.horizontalLayout\_38)

self.tabWidget\_2.addTab(self.tab\_10, "")

self.tab\_7 = QtWidgets.QWidget()

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

self.verticalLayout\_13 = QtWidgets.QVBoxLayout(self.tab\_7)

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

self.horizontalLayout\_20 = QtWidgets.QHBoxLayout()

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

self.label\_16 = QtWidgets.QLabel(self.tab\_7)

self.label\_16.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_16.setAlignment(QtCore.Qt.AlignCenter)

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

self.horizontalLayout\_20.addWidget(self.label\_16)

self.verticalLayout\_13.addLayout(self.horizontalLayout\_20)

self.verticalLayout\_12 = QtWidgets.QVBoxLayout()

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

self.horizontalLayout\_22 = QtWidgets.QHBoxLayout()

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

self.label\_17 = QtWidgets.QLabel(self.tab\_7)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_22.addWidget(self.label\_17)

self.edit\_title\_book = QtWidgets.QLineEdit(self.tab\_7)

self.edit\_title\_book.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;background-color: rgb(255, 255, 255);")

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

self.horizontalLayout\_22.addWidget(self.edit\_title\_book)

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

self.verticalLayout\_12.addLayout(self.horizontalLayout\_22)

self.horizontalLayout\_23 = QtWidgets.QHBoxLayout()

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

self.label\_18 = QtWidgets.QLabel(self.tab\_7)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_23.addWidget(self.label\_18)

self.year = QtWidgets.QDateEdit(self.tab\_7)

self.year.setStyleSheet("background-color: rgb(255, 255, 255);")

self.year.setObjectName("year")

self.horizontalLayout\_23.addWidget(self.year)

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

self.verticalLayout\_12.addLayout(self.horizontalLayout\_23)

self.horizontalLayout\_28 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_28.setObjectName("horizontalLayout\_28")

self.label\_29 = QtWidgets.QLabel(self.tab\_7)

self.label\_29.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_28.addWidget(self.label\_29)

self.spin\_lists = QtWidgets.QSpinBox(self.tab\_7)

self.spin\_lists.setStyleSheet("background-color: rgb(255, 255, 255);")

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

self.horizontalLayout\_28.addWidget(self.spin\_lists)

self.horizontalLayout\_28.setStretch(1, 100)

self.verticalLayout\_12.addLayout(self.horizontalLayout\_28)

self.horizontalLayout\_24 = QtWidgets.QHBoxLayout()

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

self.label\_19 = QtWidgets.QLabel(self.tab\_7)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

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

self.spin\_cost = QtWidgets.QSpinBox(self.tab\_7)

self.spin\_cost.setStyleSheet("background-color: rgb(255, 255, 255);")

self.spin\_cost.setMaximum(100000)

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

self.horizontalLayout\_24.addWidget(self.spin\_cost)

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

self.verticalLayout\_12.addLayout(self.horizontalLayout\_24)

self.horizontalLayout\_25 = QtWidgets.QHBoxLayout()

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

self.label\_20 = QtWidgets.QLabel(self.tab\_7)

self.label\_20.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_25.addWidget(self.label\_20)

self.comboBox\_genre = QtWidgets.QComboBox(self.tab\_7)

self.comboBox\_genre.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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

self.horizontalLayout\_25.addWidget(self.comboBox\_genre)

self.horizontalLayout\_25.setStretch(1, 100)

self.verticalLayout\_12.addLayout(self.horizontalLayout\_25)

self.horizontalLayout\_26 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_26.setObjectName("horizontalLayout\_26")

self.label\_21 = QtWidgets.QLabel(self.tab\_7)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_26.addWidget(self.label\_21)

self.comboBox\_author = QtWidgets.QComboBox(self.tab\_7)

self.comboBox\_author.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;background-color: rgb(255, 255, 255);")

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

self.horizontalLayout\_26.addWidget(self.comboBox\_author)

self.horizontalLayout\_26.setStretch(1, 100)

self.verticalLayout\_12.addLayout(self.horizontalLayout\_26)

self.horizontalLayout\_27 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_27.setObjectName("horizontalLayout\_27")

self.label\_22 = QtWidgets.QLabel(self.tab\_7)

self.label\_22.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_27.addWidget(self.label\_22)

self.comboBox\_ph = QtWidgets.QComboBox(self.tab\_7)

self.comboBox\_ph.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;background-color: rgb(255, 255, 255);")

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

self.horizontalLayout\_27.addWidget(self.comboBox\_ph)

self.horizontalLayout\_27.setStretch(1, 100)

self.verticalLayout\_12.addLayout(self.horizontalLayout\_27)

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

self.table\_book = QtWidgets.QTableWidget(self.tab\_7)

self.table\_book.setStyleSheet("background-color: rgb(255, 255, 255);")

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\_13.addWidget(self.table\_book)

self.horizontalLayout\_21 = QtWidgets.QHBoxLayout()

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

spacerItem43 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_21.addItem(spacerItem43)

self.btn\_new\_book = QtWidgets.QPushButton(self.tab\_7)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

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

spacerItem44 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_21.addItem(spacerItem44)

self.btn\_delete\_book = QtWidgets.QPushButton(self.tab\_7)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

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

spacerItem45 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_21.addItem(spacerItem45)

self.btn\_save\_book = QtWidgets.QPushButton(self.tab\_7)

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

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

"font: 10pt \"MS Shell Dlg 2\";")

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

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

spacerItem46 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_21.addItem(spacerItem46)

self.verticalLayout\_13.addLayout(self.horizontalLayout\_21)

self.tabWidget\_2.addTab(self.tab\_7, "")

self.verticalLayout\_11.addWidget(self.tabWidget\_2)

self.tabWidget.addTab(self.tab\_6, "")

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

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

self.page\_2 = QtWidgets.QWidget()

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

self.verticalLayout\_5 = QtWidgets.QVBoxLayout(self.page\_2)

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

self.stackedWidget\_req\_2 = QtWidgets.QStackedWidget(self.page\_2)

self.stackedWidget\_req\_2.setObjectName("stackedWidget\_req\_2")

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

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

self.verticalLayout\_25 = QtWidgets.QVBoxLayout(self.page\_req\_2)

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

self.verticalLayout\_20 = QtWidgets.QVBoxLayout()

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

self.label\_14 = QtWidgets.QLabel(self.page\_req\_2)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(13)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_14.setFont(font)

self.label\_14.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

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

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

self.verticalLayout\_20.addWidget(self.label\_14)

self.horizontalLayout\_16 = QtWidgets.QHBoxLayout()

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

self.label\_33 = QtWidgets.QLabel(self.page\_req\_2)

self.label\_33.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_33.setObjectName("label\_33")

self.horizontalLayout\_16.addWidget(self.label\_33)

self.comboBox\_clients\_2 = QtWidgets.QComboBox(self.page\_req\_2)

self.comboBox\_clients\_2.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

self.comboBox\_clients\_2.setObjectName("comboBox\_clients\_2")

self.horizontalLayout\_16.addWidget(self.comboBox\_clients\_2)

self.horizontalLayout\_16.setStretch(0, 1)

self.horizontalLayout\_16.setStretch(1, 10)

self.verticalLayout\_20.addLayout(self.horizontalLayout\_16)

self.horizontalLayout\_17 = QtWidgets.QHBoxLayout()

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

self.label\_34 = QtWidgets.QLabel(self.page\_req\_2)

self.label\_34.setStyleSheet("background-color: rgb(216, 0, 4);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.label\_34.setObjectName("label\_34")

self.horizontalLayout\_17.addWidget(self.label\_34)

self.comboBox\_book\_2 = QtWidgets.QComboBox(self.page\_req\_2)

self.comboBox\_book\_2.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

self.comboBox\_book\_2.setObjectName("comboBox\_book\_2")

self.horizontalLayout\_17.addWidget(self.comboBox\_book\_2)

self.btn\_plus\_2 = QtWidgets.QPushButton(self.page\_req\_2)

self.btn\_plus\_2.setMaximumSize(QtCore.QSize(20, 20))

self.btn\_plus\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

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

self.btn\_plus\_2.setText("")

self.btn\_plus\_2.setIcon(icon1)

self.btn\_plus\_2.setIconSize(QtCore.QSize(20, 20))

self.btn\_plus\_2.setCheckable(False)

self.btn\_plus\_2.setChecked(False)

self.btn\_plus\_2.setAutoRepeat(False)

self.btn\_plus\_2.setAutoExclusive(False)

self.btn\_plus\_2.setObjectName("btn\_plus\_2")

self.horizontalLayout\_17.addWidget(self.btn\_plus\_2)

self.horizontalLayout\_17.setStretch(0, 1)

self.horizontalLayout\_17.setStretch(1, 100)

self.verticalLayout\_20.addLayout(self.horizontalLayout\_17)

self.add\_new\_field\_2 = QtWidgets.QListWidget(self.page\_req\_2)

self.add\_new\_field\_2.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"")

self.add\_new\_field\_2.setIconSize(QtCore.QSize(500, 151))

self.add\_new\_field\_2.setSelectionRectVisible(True)

self.add\_new\_field\_2.setObjectName("add\_new\_field\_2")

self.verticalLayout\_20.addWidget(self.add\_new\_field\_2)

self.horizontalLayout\_40 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_40.setObjectName("horizontalLayout\_40")

spacerItem47 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_40.addItem(spacerItem47)

self.btn\_new\_order\_2 = QtWidgets.QPushButton(self.page\_req\_2)

self.btn\_new\_order\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_new\_order\_2.setObjectName("btn\_new\_order\_2")

self.horizontalLayout\_40.addWidget(self.btn\_new\_order\_2)

spacerItem48 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_40.addItem(spacerItem48)

self.btn\_save\_request\_2 = QtWidgets.QPushButton(self.page\_req\_2)

self.btn\_save\_request\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_save\_request\_2.setObjectName("btn\_save\_request\_2")

self.horizontalLayout\_40.addWidget(self.btn\_save\_request\_2)

spacerItem49 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_40.addItem(spacerItem49)

self.btn\_code\_2 = QtWidgets.QPushButton(self.page\_req\_2)

self.btn\_code\_2.setEnabled(True)

self.btn\_code\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_code\_2.setObjectName("btn\_code\_2")

self.horizontalLayout\_40.addWidget(self.btn\_code\_2)

spacerItem50 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_40.addItem(spacerItem50)

self.btn\_all\_req\_2 = QtWidgets.QPushButton(self.page\_req\_2)

self.btn\_all\_req\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_all\_req\_2.setObjectName("btn\_all\_req\_2")

self.horizontalLayout\_40.addWidget(self.btn\_all\_req\_2)

spacerItem51 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_40.addItem(spacerItem51)

self.btn\_all\_clients\_2 = QtWidgets.QPushButton(self.page\_req\_2)

self.btn\_all\_clients\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_all\_clients\_2.setObjectName("btn\_all\_clients\_2")

self.horizontalLayout\_40.addWidget(self.btn\_all\_clients\_2)

spacerItem52 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_40.addItem(spacerItem52)

self.btn\_new\_client\_2 = QtWidgets.QPushButton(self.page\_req\_2)

self.btn\_new\_client\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

"font: 10pt \"MS Shell Dlg 2\";")

self.btn\_new\_client\_2.setObjectName("btn\_new\_client\_2")

self.horizontalLayout\_40.addWidget(self.btn\_new\_client\_2)

spacerItem53 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_40.addItem(spacerItem53)

self.verticalLayout\_20.addLayout(self.horizontalLayout\_40)

self.verticalLayout\_25.addLayout(self.verticalLayout\_20)

self.stackedWidget\_req\_2.addWidget(self.page\_req\_2)

self.page2 = QtWidgets.QWidget()

self.page2.setObjectName("page2")

self.verticalLayout\_33 = QtWidgets.QVBoxLayout(self.page2)

self.verticalLayout\_33.setObjectName("verticalLayout\_33")

self.verticalLayout\_34 = QtWidgets.QVBoxLayout()

self.verticalLayout\_34.setObjectName("verticalLayout\_34")

self.label\_48 = QtWidgets.QLabel(self.page2)

self.label\_48.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_48.setAlignment(QtCore.Qt.AlignCenter)

self.label\_48.setObjectName("label\_48")

self.verticalLayout\_34.addWidget(self.label\_48)

self.verticalLayout\_33.addLayout(self.verticalLayout\_34)

self.table\_orders\_2 = QtWidgets.QTableWidget(self.page2)

self.table\_orders\_2.setStyleSheet("background-color: rgb(255, 255, 255); ")

self.table\_orders\_2.setFrameShape(QtWidgets.QFrame.Box)

self.table\_orders\_2.setFrameShadow(QtWidgets.QFrame.Plain)

self.table\_orders\_2.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.table\_orders\_2.setObjectName("table\_orders\_2")

self.table\_orders\_2.setColumnCount(0)

self.table\_orders\_2.setRowCount(0)

self.verticalLayout\_33.addWidget(self.table\_orders\_2)

self.horizontalLayout\_57 = QtWidgets.QHBoxLayout()

self.horizontalLayout\_57.setObjectName("horizontalLayout\_57")

spacerItem54 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_57.addItem(spacerItem54)

self.btn\_back\_ord\_3 = QtWidgets.QPushButton(self.page2)

self.btn\_back\_ord\_3.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

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

self.btn\_back\_ord\_3.setObjectName("btn\_back\_ord\_3")

self.horizontalLayout\_57.addWidget(self.btn\_back\_ord\_3)

spacerItem55 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_57.addItem(spacerItem55)

self.btn\_del\_order\_2 = QtWidgets.QPushButton(self.page2)

self.btn\_del\_order\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

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

self.btn\_del\_order\_2.setObjectName("btn\_del\_order\_2")

self.horizontalLayout\_57.addWidget(self.btn\_del\_order\_2)

spacerItem56 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_57.addItem(spacerItem56)

self.btn\_save\_order\_2 = QtWidgets.QPushButton(self.page2)

self.btn\_save\_order\_2.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

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

self.btn\_save\_order\_2.setObjectName("btn\_save\_order\_2")

self.horizontalLayout\_57.addWidget(self.btn\_save\_order\_2)

spacerItem57 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_57.addItem(spacerItem57)

self.verticalLayout\_33.addLayout(self.horizontalLayout\_57)

self.stackedWidget\_req\_2.addWidget(self.page2)

self.page3 = QtWidgets.QWidget()

self.page3.setObjectName("page3")

self.verticalLayout\_35 = QtWidgets.QVBoxLayout(self.page3)

self.verticalLayout\_35.setObjectName("verticalLayout\_35")

self.label\_49 = QtWidgets.QLabel(self.page3)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(13)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.label\_49.setFont(font)

self.label\_49.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")

self.label\_49.setAlignment(QtCore.Qt.AlignCenter)

self.label\_49.setObjectName("label\_49")

self.verticalLayout\_35.addWidget(self.label\_49)

self.table\_clients\_2 = QtWidgets.QTableWidget(self.page3)

self.table\_clients\_2.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"")

self.table\_clients\_2.setFrameShape(QtWidgets.QFrame.Box)

self.table\_clients\_2.setFrameShadow(QtWidgets.QFrame.Plain)

self.table\_clients\_2.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.table\_clients\_2.setDragDropOverwriteMode(True)

self.table\_clients\_2.setGridStyle(QtCore.Qt.SolidLine)

self.table\_clients\_2.setObjectName("table\_clients\_2")

self.table\_clients\_2.setColumnCount(7)

self.table\_clients\_2.setRowCount(0)

item = QtWidgets.QTableWidgetItem()

self.table\_clients\_2.setHorizontalHeaderItem(0, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients\_2.setHorizontalHeaderItem(1, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients\_2.setHorizontalHeaderItem(2, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients\_2.setHorizontalHeaderItem(3, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients\_2.setHorizontalHeaderItem(4, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients\_2.setHorizontalHeaderItem(5, item)

item = QtWidgets.QTableWidgetItem()

self.table\_clients\_2.setHorizontalHeaderItem(6, item)

self.table\_clients\_2.horizontalHeader().setStretchLastSection(True)

self.verticalLayout\_35.addWidget(self.table\_clients\_2)

self.horizontalLayout\_8 = QtWidgets.QHBoxLayout()

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

spacerItem58 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_8.addItem(spacerItem58)

self.btn\_back\_ord\_4 = QtWidgets.QPushButton(self.page3)

self.btn\_back\_ord\_4.setStyleSheet("background-color: rgb(143, 0, 2);\n"

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

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

self.btn\_back\_ord\_4.setObjectName("btn\_back\_ord\_4")

self.horizontalLayout\_8.addWidget(self.btn\_back\_ord\_4)

spacerItem59 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout\_8.addItem(spacerItem59)

self.verticalLayout\_35.addLayout(self.horizontalLayout\_8)

self.stackedWidget\_req\_2.addWidget(self.page3)

self.verticalLayout\_5.addWidget(self.stackedWidget\_req\_2)

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

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

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

self.horizontalLayout = QtWidgets.QHBoxLayout()

self.horizontalLayout.setSpacing(0)

self.horizontalLayout.setObjectName("horizontalLayout")

spacerItem60 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout.addItem(spacerItem60)

self.btn\_exit = QtWidgets.QPushButton(self.centralwidget)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(10)

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"

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout.addWidget(self.btn\_exit)

spacerItem61 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout.addItem(spacerItem61)

self.btn\_back = QtWidgets.QPushButton(self.centralwidget)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(10)

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"

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout.addWidget(self.btn\_back)

spacerItem62 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout.addItem(spacerItem62)

self.btn\_next = QtWidgets.QPushButton(self.centralwidget)

font = QtGui.QFont()

font.setFamily("MS Shell Dlg 2")

font.setPointSize(10)

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"

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

"font: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout.addWidget(self.btn\_next)

spacerItem63 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout.addItem(spacerItem63)

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(0)

self.tabWidget.setCurrentIndex(0)

self.stackedWidget\_req.setCurrentIndex(0)

self.add\_new\_field.setCurrentRow(-1)

self.stackedWidget\_prov.setCurrentIndex(0)

self.tabWidget\_2.setCurrentIndex(0)

self.stackedWidget\_req\_2.setCurrentIndex(0)

self.add\_new\_field\_2.setCurrentRow(-1)

QtCore.QMetaObject.connectSlotsByName(BookShop)

def retranslateUi(self, BookShop):

\_translate = QtCore.QCoreApplication.translate

BookShop.setWindowTitle(\_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\_10.setText(\_translate("BookShop", "Создание заказа"))

self.label\_11.setText(\_translate("BookShop", "Выберите клиента: "))

self.label\_12.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\_req.setText(\_translate("BookShop", "Все заказы"))

self.btn\_all\_clients.setText(\_translate("BookShop", " Все клиенты "))

self.btn\_new\_client.setText(\_translate("BookShop", " Новый клиент "))

self.label\_46.setText(\_translate("BookShop", "Просмотр заказов"))

self.btn\_back\_ord.setText(\_translate("BookShop", " Назад "))

self.btn\_del\_order.setText(\_translate("BookShop", " Удалить заказ "))

self.btn\_save\_order.setText(\_translate("BookShop", " Сохранить "))

self.label\_47.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.btn\_back\_ord\_2.setText(\_translate("BookShop", " Назад "))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_4), \_translate("BookShop", "Заказы"))

self.label\_13.setText(\_translate("BookShop", "Создание отчёта"))

self.label\_15.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.btn\_pdf.setText(\_translate("BookShop", "сохранить в pdf"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_5), \_translate("BookShop", "Отчёты"))

self.label\_37.setText(\_translate("BookShop", "Заявка на поставку книг"))

self.label\_38.setText(\_translate("BookShop", "Поставщик: "))

self.label\_39.setText(\_translate("BookShop", "Количество книг: "))

self.label\_40.setText(\_translate("BookShop", "Книга: "))

self.btn\_create\_ord.setText(\_translate("BookShop", " Создать "))

self.btn\_send.setText(\_translate("BookShop", " Отправить "))

self.btn\_all\_prov.setText(\_translate("BookShop", " Все поставщики "))

self.btn\_all\_pr\_od.setText(\_translate("BookShop", " Все заявки "))

self.label\_44.setText(\_translate("BookShop", "Редактирование поставщиков"))

self.label\_41.setText(\_translate("BookShop", "Наименование компании: "))

self.label\_42.setText(\_translate("BookShop", "Юридический адрес: "))

self.label\_43.setText(\_translate("BookShop", "Номер телефона: "))

self.btn\_back\_prov.setText(\_translate("BookShop", " Назад "))

self.btn\_add\_prov.setText(\_translate("BookShop", " Добавить "))

self.btn\_del\_prov.setText(\_translate("BookShop", " Удалить "))

self.btn\_save\_prov.setText(\_translate("BookShop", " Сохранить "))

self.label\_45.setText(\_translate("BookShop", "Просмотр поставок"))

self.btn\_back\_prov\_2.setText(\_translate("BookShop", " Назад "))

self.btn\_delete\_pr\_od.setText(\_translate("BookShop", " Удалить "))

self.btn\_save\_pr\_od.setText(\_translate("BookShop", " Сохранить "))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_11), \_translate("BookShop", "Поставки"))

self.label\_9.setText(\_translate("BookShop", "Просмотр истории входа сотрудников"))

self.table\_entry.setSortingEnabled(True)

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_3), \_translate("BookShop", "История входа"))

self.label\_25.setText(\_translate("BookShop", "Редактирование авторов"))

self.label\_26.setText(\_translate("BookShop", "Введите фамилию: "))

self.label\_27.setText(\_translate("BookShop", "Введите имя: "))

self.label\_28.setText(\_translate("BookShop", "Введите отчество: "))

self.label\_30.setText(\_translate("BookShop", "Укажите дату рождения: "))

self.btn\_new\_author.setText(\_translate("BookShop", " Добавить "))

self.btn\_delete\_author.setText(\_translate("BookShop", " Удалить "))

self.btn\_save\_author.setText(\_translate("BookShop", " Сохранить "))

self.tabWidget\_2.setTabText(self.tabWidget\_2.indexOf(self.tab\_9), \_translate("BookShop", "Авторы"))

self.label\_23.setText(\_translate("BookShop", "Редактирование жанров"))

self.label\_24.setText(\_translate("BookShop", "Введите жанр: "))

self.btn\_new\_genre.setText(\_translate("BookShop", " Добавить "))

self.btn\_delete\_genre.setText(\_translate("BookShop", " Удалить "))

self.btn\_save\_genre.setText(\_translate("BookShop", " Сохранить "))

self.tabWidget\_2.setTabText(self.tabWidget\_2.indexOf(self.tab\_8), \_translate("BookShop", "Жанры"))

self.label\_32.setText(\_translate("BookShop", "Редактирование издательств"))

self.label\_31.setText(\_translate("BookShop", "Введите название издательства: "))

self.btn\_new\_ph.setText(\_translate("BookShop", " Добавить "))

self.btn\_delete\_ph.setText(\_translate("BookShop", " Удалить "))

self.btn\_save\_ph.setText(\_translate("BookShop", " Сохранить "))

self.tabWidget\_2.setTabText(self.tabWidget\_2.indexOf(self.tab\_10), \_translate("BookShop", "Издательства"))

self.label\_16.setText(\_translate("BookShop", "Редактирование книг"))

self.label\_17.setText(\_translate("BookShop", "Введите название книги:"))

self.label\_18.setText(\_translate("BookShop", "Введите год издания:"))

self.label\_29.setText(\_translate("BookShop", "Введите количество страниц:"))

self.label\_19.setText(\_translate("BookShop", "Введите стоимость книги (руб):"))

self.label\_20.setText(\_translate("BookShop", "Выберите жанр:"))

self.label\_21.setText(\_translate("BookShop", "Выберите автора:"))

self.label\_22.setText(\_translate("BookShop", "Выберите издательство:"))

self.btn\_new\_book.setText(\_translate("BookShop", " Добавить "))

self.btn\_delete\_book.setText(\_translate("BookShop", " Удалить "))

self.btn\_save\_book.setText(\_translate("BookShop", " Сохранить "))

self.tabWidget\_2.setTabText(self.tabWidget\_2.indexOf(self.tab\_7), \_translate("BookShop", "Книги"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_6), \_translate("BookShop", "Редактирование данных"))

self.label\_14.setText(\_translate("BookShop", "Создание заказа"))

self.label\_33.setText(\_translate("BookShop", "Выберите клиента: "))

self.label\_34.setText(\_translate("BookShop", "Выберите книгу: "))

self.btn\_new\_order\_2.setText(\_translate("BookShop", " Создать заказ "))

self.btn\_save\_request\_2.setText(\_translate("BookShop", " Сохранить заказ "))

self.btn\_code\_2.setText(\_translate("BookShop", " Штрих-код "))

self.btn\_all\_req\_2.setText(\_translate("BookShop", "Все заказы"))

self.btn\_all\_clients\_2.setText(\_translate("BookShop", " Все клиенты "))

self.btn\_new\_client\_2.setText(\_translate("BookShop", " Новый клиент "))

self.label\_48.setText(\_translate("BookShop", "Просмотр заказов"))

self.btn\_back\_ord\_3.setText(\_translate("BookShop", " Назад "))

self.btn\_del\_order\_2.setText(\_translate("BookShop", " Удалить заказ "))

self.btn\_save\_order\_2.setText(\_translate("BookShop", " Сохранить "))

self.label\_49.setText(\_translate("BookShop", "Просмотр всех клиентов"))

self.table\_clients\_2.setSortingEnabled(False)

item = self.table\_clients\_2.horizontalHeaderItem(0)

item.setText(\_translate("BookShop", "id"))

item = self.table\_clients\_2.horizontalHeaderItem(1)

item.setText(\_translate("BookShop", "ФИО"))

item = self.table\_clients\_2.horizontalHeaderItem(2)

item.setText(\_translate("BookShop", "Код клиента"))

item = self.table\_clients\_2.horizontalHeaderItem(3)

item.setText(\_translate("BookShop", "Паспортные данные"))

item = self.table\_clients\_2.horizontalHeaderItem(4)

item.setText(\_translate("BookShop", "Дата рождения"))

item = self.table\_clients\_2.horizontalHeaderItem(5)

item.setText(\_translate("BookShop", "Адрес"))

item = self.table\_clients\_2.horizontalHeaderItem(6)

item.setText(\_translate("BookShop", "email"))

self.btn\_back\_ord\_4.setText(\_translate("BookShop", " Назад "))

self.btn\_exit.setText(\_translate("BookShop", " Выход "))

self.btn\_back.setText(\_translate("BookShop", " Назад "))

self.btn\_next.setText(\_translate("BookShop", " Вперед "))

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)

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

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

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

self.label.setSizePolicy(sizePolicy)

self.label.setMinimumSize(QtCore.QSize(300, 70))

self.label.setMaximumSize(QtCore.QSize(300, 70))

font = QtGui.QFont()

font.setFamily("Comic Sans MS")

font.setPointSize(13)

self.label.setFont(font)

self.label.setText("")

self.label.setPixmap(QtGui.QPixmap("../лого.png"))

self.label.setScaledContents(True)

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: 13pt \"MS Shell Dlg 2\";")

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: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_4.addWidget(self.label\_5)

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

self.edit\_surname.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_20.addWidget(self.label\_3)

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

self.edit\_name.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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: 10pt \"MS Shell Dlg 2\";")

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

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

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

self.edit\_lastname.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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: 10pt \"MS Shell Dlg 2\";")

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

self.horizontalLayout\_5.addWidget(self.label\_7)

self.date\_birth = QtWidgets.QDateEdit(Dialog)

self.date\_birth.setStyleSheet("background-color: rgb(255, 255, 255); ")

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: 10pt \"MS Shell Dlg 2\";")

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

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

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

self.edit\_email.setStyleSheet("background-color: rgb(255, 255, 255); \n"

"border-top: 1px solid #3e3e3e;\n"

"border-left: 1px solid #3e3e3e;\n"

"border-right: 1px solid #3e3e3e;\n"

"border-bottom: 1px solid #3e3e3e;")

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"

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

"font: 10pt \"MS Shell Dlg 2\";")

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\_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"))

# **ПРИЛОЖЕНИЕ В Исходный код программы**

gui.py:

import datetime

import logging

import os

import random

import sys

import time

from io import BytesIO

import barcode

import img2pdf

from PyQt5 import QtWidgets

from PyQt5 import uic, QtGui

from PyQt5.QtCore import Qt

from PyQt5.QtGui import QPixmap

from PyQt5.QtWidgets import QListWidgetItem, QAction

from PyQt5.QtWidgets import QMainWindow, QApplication, QTableWidgetItem, QDialog

from PyQt5.QtWidgets import QMessageBox

from barcode.writer import ImageWriter

from captcha.image import ImageCaptcha

from facade import Facade

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\_req = self.ui.stackedWidget\_req

self.page\_req\_2 = self.ui.stackedWidget\_req\_2

self.page\_prov = self.ui.stackedWidget\_prov

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\_all\_clients\_2.clicked.connect(self.page\_all\_clients\_2)

self.ui.btn\_all\_req.clicked.connect(self.page\_all\_requests)

self.ui.btn\_all\_pr\_od.clicked.connect(self.page\_all\_pr\_od)

self.ui.btn\_all\_req\_2.clicked.connect(self.page\_all\_requests\_2)

self.ui.btn\_all\_prov.clicked.connect(self.page\_all\_providers)

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.ui.btn\_add\_prov.clicked.connect(self.new\_provider)

self.ui.btn\_del\_prov.clicked.connect(self.delete\_provider)

self.ui.btn\_save\_prov.clicked.connect(self.save\_provider)

self.ui.btn\_delete\_pr\_od.clicked.connect(self.delete\_pr\_od)

self.ui.btn\_save\_pr\_od.clicked.connect(self.save\_pr\_od)

self.ui.btn\_del\_order.clicked.connect(self.delete\_req)

self.ui.btn\_save\_order.clicked.connect(self.save\_req)

self.build\_combobox\_clients()

self.build\_combobox\_books()

self.build\_combobox\_genre()

self.build\_combobox\_author()

self.build\_combobox\_ph()

self.build\_combobox\_provider()

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\_back\_ord.clicked.connect(self.page\_ord\_back)

self.ui.btn\_back\_ord\_2.clicked.connect(self.page\_ord\_back)

self.ui.btn\_back\_ord\_3.clicked.connect(self.page\_ord\_back\_2)

self.ui.btn\_back\_ord\_4.clicked.connect(self.page\_ord\_back\_2)

self.ui.btn\_create\_ord.clicked.connect(self.add\_new\_provider\_order)

self.ui.btn\_send.clicked.connect(self.save\_provider\_order)

self.ui.btn\_add\_book\_pr\_od.clicked.connect(self.add\_book\_to\_pr\_od)

self.ui.btn\_back\_prov.clicked.connect(self.page\_prov\_back)

self.ui.btn\_back\_prov\_2.clicked.connect(self.page\_prov\_back)

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()

self.updateTableProvider()

self.updateTableProviderOrder()

self.updateTableOrder()

self.updateTableOrder\_2()

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)

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)

self.hide()

self.open\_auth()

event.ignore()

logging.log(logging.INFO, 'Произведён выход через кнопку "Х"')

def page\_ord\_back(self):

self.page\_req.setCurrentIndex(0)

logging.log(logging.INFO, 'Переход к таблице "Клиент"')

def page\_ord\_back\_2(self):

self.page\_req\_2.setCurrentIndex(0)

logging.log(logging.INFO, 'Переход к таблице "Клиент"')

def page\_prov\_back(self):

self.page\_prov.setCurrentIndex(0)

logging.log(logging.INFO, 'Переход к таблице "Клиент"')

def page\_all\_clients(self):

"""

Переход к странице с таблицей клиентов.

"""

self.updateTableClient()

self.page\_req.setCurrentIndex(2)

logging.log(logging.INFO, 'Переход к таблице "Клиент"')

def page\_all\_clients\_2(self):

"""

Переход к странице с таблицей клиентов.

"""

self.updateTableClient\_2()

self.page\_req\_2.setCurrentIndex(2)

logging.log(logging.INFO, 'Переход к таблице "Клиент"')

def page\_all\_pr\_od(self):

"""

Переход к странице с таблицей заказов.

"""

self.updateTableProviderOrder()

self.page\_prov.setCurrentIndex(2)

logging.log(logging.INFO, 'Переход к таблице "Поставки"')

def page\_all\_requests(self):

"""

Переход к странице с таблицей заказов.

"""

self.updateTableOrder()

self.page\_req.setCurrentIndex(1)

logging.log(logging.INFO, 'Переход к таблице "Заказ"')

def page\_all\_requests\_2(self):

"""

Переход к странице с таблицей заказов.

"""

self.updateTableOrder()

self.page\_req\_2.setCurrentIndex(1)

logging.log(logging.INFO, 'Переход к таблице "Заказ"')

def page\_all\_providers(self):

"""

Переход к странице с таблицей заказов.

"""

self.updateTableProvider()

self.page\_prov.setCurrentIndex(1)

logging.log(logging.INFO, 'Переход к таблице "Поставщик"')

def updateTableClient(self):

"""

Обновление таблицы `Клиент`.

"""

self.table\_clients.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 updateTableClient\_2(self):

"""

Обновление таблицы `Клиент`.

"""

self.table\_clients\_2.clear()

rec = self.facade.read\_clients()

self.ui.table\_clients\_2.setColumnCount(6)

self.ui.table\_clients\_2.setRowCount(len(rec))

self.ui.table\_clients\_2.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\_2.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(8)

self.ui.table\_book.setRowCount(len(rec))

self.ui.table\_book.setHorizontalHeaderLabels(

['Код книги', 'Название книги', 'Год издания', 'Количество страниц', 'Жанр', 'Автор',

'Издательство', 'Стоимость'])

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(4)

self.table\_entry.setRowCount(len(rec))

self.table\_entry.setHorizontalHeaderLabels(['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 updateTableProvider(self):

"""

Обновление таблицы `Поставщик`.

"""

self.table\_provider.clear()

rec = self.facade.read\_provider()

self.table\_provider.setColumnCount(4)

self.table\_provider.setRowCount(len(rec))

self.table\_provider.setHorizontalHeaderLabels(

['ID', 'Наименование компании', 'Юридический адрес', 'Номер телефона'])

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\_provider.setItem(i, x, item)

logging.log(logging.INFO, 'Таблица "Поставщик" обновлена')

def updateTableProviderOrder(self):

"""

Обновление таблицы `Поставки`.

"""

self.table\_pr\_od.clear()

rec = self.facade.read\_provider\_orders()

self.table\_pr\_od.setColumnCount(6)

self.table\_pr\_od.setRowCount(len(rec))

self.table\_pr\_od.setHorizontalHeaderLabels(['Код поставки', 'Поставщик', 'Дата создания', 'Книга', 'Количество книг', 'Стоимость'])

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\_pr\_od.setItem(i, x, item)

logging.log(logging.INFO, 'Таблица "Поставка" обновлена')

def updateTableOrder(self):

"""

Обновление таблицы `Заказ`.

"""

self.table\_orders.clear()

rec = self.facade.read\_order()

self.table\_orders.setColumnCount(7)

self.table\_orders.setRowCount(len(rec))

self.table\_orders.setHorizontalHeaderLabels(

['Код заказа', 'Номер заказа', 'Дата создания', 'Время заказа', 'Клиент', 'Книга', 'Сотрудник'])

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\_orders.setItem(i, x, item)

logging.log(logging.INFO, 'Таблица "Заказ" обновлена')

def updateTableOrder\_2(self):

"""

Обновление таблицы `Заказ`.

"""

self.table\_orders\_2.clear()

rec = self.facade.read\_order()

self.table\_orders\_2.setColumnCount(7)

self.table\_orders\_2.setRowCount(len(rec))

self.table\_orders\_2.setHorizontalHeaderLabels(

['Код заказа', 'Номер заказа', 'Дата создания', 'Время заказа', 'Клиент', 'Книга', 'Сотрудник'])

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\_orders\_2.setItem(i, x, item)

logging.log(logging.INFO, 'Таблица "Заказ" обновлена')

def new\_book(self):

"""

Добавление книги.

"""

title\_book = self.ui.edit\_title\_book.text()

year = self.ui.year.dateTime().toString('yyyy')

lists = self.ui.spin\_lists.value()

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, year, lists, genre, author, ph, cost\_book)

self.updateTableBook()

self.ui.edit\_title\_book.clear()

self.ui.spin\_lists.clear()

self.ui.spin\_cost.clear()

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],

string[6], string[7])

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()

self.ui.edit\_genre.clear()

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.copy())

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()

self.ui.edit\_f.clear()

self.ui.edit\_i.clear()

self.ui.edit\_o.clear()

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()

self.ui.edit\_ph.clear()

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 new\_provider(self):

"""

Добавление поставщика.

"""

name\_pr = self.ui.edit\_name\_pr.text()

address\_pr = self.ui.edit\_address\_pr.text()

phone\_pr = self.ui.edit\_phone\_pr.text()

if name\_pr != '' and address\_pr != '':

if phone\_pr != '':

self.facade.insert\_provider(name\_pr, address\_pr, phone\_pr)

if phone\_pr == '':

self.facade.insert\_provider(name\_pr, address\_pr, '')

self.updateTableProvider()

self.ui.edit\_name\_pr.clear()

self.ui.edit\_address\_pr.clear()

self.ui.edit\_phone\_pr.clear()

logging.log(logging.INFO, 'Добавлены данные в таблицу "Поставщик"')

def delete\_provider(self):

"""

Удаление выбранного поставщика.

"""

SelectedRow = self.table\_provider.currentRow()

rowcount = self.table\_provider.rowCount()

colcount = self.table\_provider.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\_provider.setItem(SelectedRow, col, QTableWidgetItem(''))

ix = self.table\_provider.model().index(-1, -1)

self.table\_provider.setCurrentIndex(ix)

logging.log(logging.INFO, 'Удалены данные из таблицы "Поставщик"')

def get\_from\_table\_provider(self):

"""

Получение данных из таблицы `Поставщик` для записи в базу данных.

:return: data

"""

rows = self.table\_provider.rowCount()

cols = self.table\_provider.columnCount()

data = []

for row in range(rows):

tmp = []

for col in range(cols):

tmp.append(self.table\_provider.item(row, col).text())

data.append(tmp)

return data

logging.log(logging.INFO, 'Получены данные для сохранения в таблицу "Поставщик"')

def save\_provider(self):

"""

Сохранение данных о поставщиках в базу данных

"""

data = self.get\_from\_table\_provider()

for string in data:

if string[1] != '':

self.facade.update\_provider(int(string[0]), string[1], string[2], string[3])

else:

self.facade.delete\_provider(int(string[0]))

self.updateTableProvider()

logging.log(logging.INFO, 'Данные сохранены в таблицу "Поставщик"')

def delete\_req(self):

"""

Удаление выбранного поставщика.

"""

SelectedRow = self.table\_orders.currentRow()

rowcount = self.table\_orders.rowCount()

colcount = self.table\_orders.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\_orders.setItem(SelectedRow, col, QTableWidgetItem(''))

ix = self.table\_orders.model().index(-1, -1)

self.table\_orders.setCurrentIndex(ix)

logging.log(logging.INFO, 'Удалены данные из таблицы "Заказ"')

def get\_from\_table\_req(self):

"""

Получение данных из таблицы `Заказ` для записи в базу данных.

:return: data

"""

rows = self.table\_orders.rowCount()

cols = self.table\_orders.columnCount()

data = []

for row in range(rows):

tmp = []

for col in range(cols):

tmp.append(self.table\_orders.item(row, col).text())

data.append(tmp)

return data

logging.log(logging.INFO, 'Получены данные для сохранения в таблицу "Заказ"')

def save\_req(self):

"""

Сохранение данных о заказах в базу данных

"""

data = self.get\_from\_table\_req()

for string in data:

if string[1] != '':

self.facade.update\_req(int(string[0]), string[1], string[2], string[3], string[4], string[5], string[6])

else:

self.facade.delete\_req(int(string[0]))

self.updateTableOrder()

logging.log(logging.INFO, 'Данные сохранены в таблицу "Заказ"')

def delete\_pr\_od(self):

"""

Удаление выбранной поставки.

"""

SelectedRow = self.table\_pr\_od.currentRow()

rowcount = self.table\_pr\_od.rowCount()

colcount = self.table\_pr\_od.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\_pr\_od.setItem(SelectedRow, col, QTableWidgetItem(''))

ix = self.table\_pr\_od.model().index(-1, -1)

self.table\_pr\_od.setCurrentIndex(ix)

logging.log(logging.INFO, 'Удалены данные из таблицы "Поставка"')

def get\_from\_table\_pr\_od(self):

"""

Получение данных из таблицы `Поставка` для записи в базу данных.

:return: data

"""

rows = self.table\_pr\_od.rowCount()

cols = self.table\_pr\_od.columnCount()

data = []

for row in range(rows):

tmp = []

for col in range(cols):

tmp.append(self.table\_pr\_od.item(row, col).text())

data.append(tmp)

return data

logging.log(logging.INFO, 'Получены данные для сохранения в таблицу "Поставка"')

def save\_pr\_od(self):

"""

Сохранение данных о поставках в базу данных

"""

data = self.get\_from\_table\_pr\_od()

for string in data:

if string[1] != '':

self.facade.update\_pr\_od(int(string[0]), string[1], string[2], string[3], string[4], string[5])

else:

self.facade.delete\_pr\_od(int(string[0]))

self.updateTableProviderOrder()

logging.log(logging.INFO, 'Данные сохранены в таблицу "Поставка"')

def build\_combobox\_clients(self):

"""

Добавление списка клиентов в ComboBox.

"""

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, 'ComboBox "Клиенты" обновлён')

def build\_combobox\_books(self):

"""

Добавление списка книг в ComboBox.

"""

books = self.facade.get\_books()

self.comboBox\_book.clear()

self.comboBookProvider.clear()

if self.comboBox\_book is not None:

self.comboBox\_book.addItems(books)

self.comboBookProvider.addItems(books)

logging.log(logging.INFO, 'ComboBox "Книги" обновлён')

def build\_combobox\_genre(self):

"""

Добавление списка жанров в ComboBox.

"""

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, 'ComboBox "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 ComboBox.

"""

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, 'ComboBox "Издательства" обновлён')

def build\_combobox\_provider(self):

"""

Добавление списка поставщиков в ComboBox.

"""

provider = self.facade.get\_providers()

self.comboProvider.clear()

if self.comboProvider is not None:

self.comboProvider.addItems(provider)

logging.log(logging.INFO, 'ComboBox "Поставщики" 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()

self.cl2 = self.facade.get\_id\_client(cl1)

client\_code = QListWidgetItem(self.cl2)

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()

self.date\_req = str(date\_now.strftime("%Y.%m.%d"))

self.time\_req = str(date\_now.strftime("%H:%M"))

number\_title = QListWidgetItem("Номер заказа:")

number = str(self.cl2) + '/' + str(self.date\_req)

self.add\_new\_field.clear()

if client != '' and 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(self.date\_req)

self.add\_new\_field.addItem(time\_req\_title)

self.add\_new\_field.addItem(self.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

self.add\_new\_field.clear()

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.cl2) + 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 add\_new\_provider\_order(self):

"""

Оформление новой поставки и его показ в ListWidget.

"""

provider\_title = QListWidgetItem("Поставщик:")

provider = QListWidgetItem(self.comboProvider.currentText())

book\_title = QListWidgetItem("Книга:")

info\_book = str(self.comboBookProvider.currentText()) + ' — ' + str(self.spinBoxQuantityBook.value()) + ' шт.'

book = QListWidgetItem(info\_book)

q = self.spinBoxQuantityBook.value()

date\_ord\_title = QListWidgetItem("Дата создания:")

date\_now = datetime.datetime.now()

self.date\_prd = str(date\_now.strftime("%Y.%m.%d"))

self.listWidgetDelivery.clear()

if provider != '' and book != '' and q >= 1:

self.listWidgetDelivery.addItem(provider\_title)

self.listWidgetDelivery.addItem(provider)

self.listWidgetDelivery.addItem(date\_ord\_title)

self.listWidgetDelivery.addItem(self.date\_prd)

self.listWidgetDelivery.addItem(book\_title)

self.listWidgetDelivery.addItem(book)

else:

self.mes\_box('Ошибка!')

logging.log(logging.INFO, 'Заявка сформирована в ListWidget')

def add\_book\_to\_pr\_od(self):

"""

Добавление книги в поставку.

"""

if self.listWidgetDelivery.count() != 0:

quantity = self.spinBoxQuantityBook.value()

if quantity >= 1:

info\_book = str(self.comboBookProvider.currentText()) + ' — ' + str(

self.spinBoxQuantityBook.value()) + ' шт.'

self.listWidgetDelivery.addItem(info\_book)

logging.log(logging.INFO, 'Добавлена книга в заявку (ListWidget)')

else:

self.mes\_box('Ошибка!')

else:

self.mes\_box('Заявка не создана!')

def save\_provider\_order(self):

"""

Сохранение поставки в базу данных.

"""

if self.listWidgetDelivery.count() != 0:

ignore\_serv = [0, 1, 2, 3, 4]

ignore2 = [1, 3]

count = self.listWidgetDelivery.count()

list\_book = [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 = []

books = []

costs = []

quantity\_list = []

for i in list\_book:

book = self.listWidgetDelivery.item(i).text()

book\_info = book.split(' — ')

id\_book = self.facade.get\_book\_id(book\_info[0])

quantity = book\_info[1][0:-4]

cost1 = float(self.facade.get\_book\_cost(id\_book))

cost = cost1 \* int(quantity)

books.append(id\_book)

quantity\_list.append(quantity)

costs.append(cost)

for j in list\_req:

if j == 1:

provider = self.listWidgetDelivery.item(j).text()

id\_provider = self.facade.get\_provider\_id(provider)

request.append(id\_provider)

else:

request.append(self.listWidgetDelivery.item(j).text())

for x, a in enumerate(books):

if a != 'o':

self.facade.create\_delivery(request[0], request[1], a, quantity\_list[x], costs[x])

else:

continue

logging.log(logging.INFO, 'Заказ добавлен в базу данных')

else:

self.mes\_box('Заявка не создана!')

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.set\_font('Calibri Regular', size=14)

pdf.cell(200, -20, txt="Отчёт по количеству заказов по дням за период времени по каждой книге", ln=1,

align="C")

pdf.set\_font('Calibri Regular', 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

pdf.set\_font('Calibri Regular', size=14)

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.

"""

super(DialogAuth, self).\_\_init\_\_(parent)

self.ui = uic.loadUi("auth.ui", self)

self.setWindowFlags(self.windowFlags() | Qt.WindowStaysOnTopHint)

self.facade = Facade()

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)

self.ui.btn\_hide\_password.setIcon(QtGui.QIcon('img/eye2.png'))

logging.log(logging.INFO, 'Пароль скрыт')

else:

self.vis\_p = True

ed.setEchoMode(QtWidgets.QLineEdit.Normal)

self.ui.btn\_hide\_password.setIcon(QtGui.QIcon('img/eye.png'))

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.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 «label» и возвращает значение капчи в переменной self.now\_captcha.

"""

syms = 'qwertyuiopasdfghjklzxcvbnm1234567890'

count\_syms = random.randint(5, 10)

now\_syms = [''] \* count\_syms

for i in range(count\_syms):

now\_syms[i] = syms[random.randint(0, 35)]

self.now\_captcha = ''.join(now\_syms)

imagea = ImageCaptcha(width = 407, height = 67)

data = imagea.generate(self.now\_captcha)

imagea.write(self.now\_captcha, 'CAPTCHA.png')

self.ui.captcha.setPixmap(QPixmap('CAPTCHA.png'))

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 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 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:

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

else:

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(f' {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]

logging.log(logging.INFO, 'Авторизован пользователь "Администратор"')

self.parent().show()

self.parent().ui.lbl\_photo.setPixmap(pix)

self.parent().ui.lbl\_fio.setText(f' {fio}')

self.parent().now\_login = auth\_log

self.parent().emp = id\_emp

self.close()

if os.path.isfile('CAPTCHA.png'):

os.remove('CAPTCHA.png')

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', 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, year, lists, genre, author, ph, cost):

"""

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, %s, %s)", (name, year, lists, genre, author, ph, cost))

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()

def insert\_provider(self, name, address, phone):

"""

Добавление поставщика

:param name: наименование поставщика

:param address: юридический адрес

:param phone: номер телефона

"""

cursor = self.conn.cursor()

cursor.execute(f"INSERT INTO provider VALUES (NULL, %s, %s, %s)", (name, address, phone))

self.conn.commit()

def insert\_delivery(self, provider, date, book, quantity, cost):

"""

Добавление поставки

:param provider: поставщик

:param date: дата создания

:param book: книга

:param quantity: количество книг

:param cost: стоимость

"""

cursor = self.conn.cursor()

cursor.execute(f"INSERT INTO provider\_order VALUES (NULL, %s, %s, %s, %s, %s)", (provider, date, book, quantity, cost))

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 издательства`='{id}'")

self.conn.commit()

def update\_book(self, id, name, year, lists, genre, author, ph, cost):

"""

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}', `Год издания`='{year}', `Количество страниц`='{lists}', `Стоимость`='{cost}', `Жанр`='{genre}', `Автор`='{author}', `Издательство`='{ph}' WHERE `Код книги`='{id}'")

self.conn.commit()

def update\_provider(self, id, name, address, phone):

cursor = self.conn.cursor()

cursor.execute(f"UPDATE provider set `Наименование компании`='{name}', `Юридический адрес`='{address}', "

f"`Номер телефона`='{phone}' WHERE `Код поставщика`='{id}'")

self.conn.commit()

def update\_req(self, id, number, date, time, client, book, employee):

"""

Обновление заказов

: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 requests set `Код заказа` = '{number}', `Дата создания` = '{date}', `Время заказа` = '{time}', "

f"`Клиент` = '{client}', `Книга` = '{book}', `Сотрудник` = '{employee}' WHERE `ID`='{id}'")

self.conn.commit()

def update\_pr\_od(self, id, provider, date, book, quantity, cost):

"""

Обновление поставки

:param id: код поставки

:param provider: код поставщика

:param date: дата создания

:param book: код книги

:param quantity: количество книг

:param cost: стоимость

"""

cursor = self.conn.cursor()

cursor.execute(

f"UPDATE provider\_order set `Поставщик`='{provider}', `Дата создания`='{date}', `Книга`='{book}', "

f"`Количество`='{quantity}', `Стоимость`='{cost}' WHERE `Код поставки`='{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()

def delete\_provider(self, id):

"""

Удаление поставщика

:param id: код поставщика

"""

cursor = self.conn.cursor()

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=0")

cursor.execute(f"DELETE FROM provider WHERE `Код поставщика`='{id}'")

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=1")

self.conn.commit()

def delete\_req(self, id):

"""

Удаление заказа

:param id: код заказа

"""

cursor = self.conn.cursor()

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=0")

cursor.execute(f"DELETE FROM requests WHERE `ID`='{id}'")

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=1")

self.conn.commit()

def delete\_pr\_od(self, id):

"""

Удаление поставки

:param id: код поставки

"""

cursor = self.conn.cursor()

cursor.execute(f"SET FOREIGN\_KEY\_CHECKS=0")

cursor.execute(f"DELETE FROM provider\_order WHERE `Код поставки`='{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\_providers(self):

"""

Получение списка поставщиков

:return: providers

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT \* FROM provider")

providers = cursor.fetchall()

return providers

def select\_provider\_order(self):

"""

Получение списка поставок

:return: provider\_orders

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT \* FROM provider\_order")

provider\_orders = cursor.fetchall()

return provider\_orders

def select\_orders(self):

"""

Получение списка заказов

:return: requests

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT \* FROM requests")

requests = cursor.fetchall()

return requests

def select\_history(self):

"""

Получение истории входа сотрудников

:return: history

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT `Дата входа`, `Дата выхода`, `Блокировка`, `Логин сотрудника` FROM history")

history = cursor.fetchall()

return history

def get\_emp\_log(self, id):

"""

Получение логина сотрудника

:param id: код сотрудника

:return: login

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT `Логин` FROM employees WHERE `Код сотрудника`='{id}'")

login = str(cursor.fetchone())

return login[2:-3]

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] + ' ' + i[2]))

return clients

def get\_genres(self):

"""

Получение наименований жанров

:return: genres

"""

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\_providers(self):

"""

Получение названий поставщиков

:return: pr

"""

pr = []

cursor = self.conn.cursor()

cursor.execute(f"SELECT `Наименование компании` FROM provider")

rows = cursor.fetchall()

for i in rows:

pr.append(str(i)[2:-3])

return pr

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]

def get\_book\_cost(self, code):

"""

Получение стоимости книги

:param code: код книги

:return: name

"""

cursor = self.conn.cursor()

cursor.execute(f"SELECT `Стоимость` FROM books WHERE `Код книги`='{code}'")

name = str(cursor.fetchone())

return name[1:-2]

def get\_provider\_id(self, name):

"""

Получение кода поставщика

:param name: наименование поставщика

:return: code

"""

cursor = self.conn.cursor(buffered=True)

cursor.execute(f"SELECT `Код поставщика` FROM provider WHERE `Наименование компании`='{name}'")

code = str(cursor.fetchone())

return code[1:-2]

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

db = Database()

db.get\_client\_name(45462526)

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 create\_delivery(self, provider, date, book, quantity, cost):

self.db.insert\_delivery(provider, date, book, quantity, cost)

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, year, lists, cost, genre, author, ph):

self.db.insert\_book(name, year, lists, cost, genre, author, ph)

def insert\_provider(self, name, address, phone):

self.db.insert\_provider(name, address, phone)

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, year, lists, genre, author, ph, cost):

self.db.update\_book(id, name, year, lists, genre, author, ph, cost)

def update\_provider(self, id, name, address, phone):

self.db.update\_provider(id, name, address, phone)

def update\_req(self, id, number, date, time, client, book, employee):

self.db.update\_req(id, number, date, time, client, book, employee)

def update\_pr\_od(self, id, provider, date, book, quantity, cost):

self.db.update\_pr\_od(id, provider, date, book, quantity, cost)

# 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)

def delete\_provider(self, id):

self.db.delete\_provider(id)

def delete\_req(self, id):

self.db.delete\_req(id)

def delete\_pr\_od(self, id):

self.db.delete\_pr\_od(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 read\_provider(self):

return self.db.select\_providers()

def read\_order(self):

return self.db.select\_orders()

def read\_provider\_orders(self):

return self.db.select\_provider\_order()

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\_providers(self):

return self.db.get\_providers()

def get\_book\_name(self, id):

return self.db.get\_book\_name(id)

def get\_book\_cost(self, id):

return self.db.get\_book\_cost(id)

def get\_provider\_name(self, id):

return self.db.get\_provider\_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)

def get\_provider\_id(self, name):

return self.db.get\_provider\_id(name)

# **ПРИЛОЖЕНИЕ Г 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)

def test\_push\_provider(self):

btn\_add = self.window.ui.btn\_add\_prov

self.window.ui.edit\_name\_pr.setText("test\_provider")

self.window.ui.edit\_address\_pr.setText("test\_provider")

self.window.ui.edit\_phone\_pr.setText("test\_provider")

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)

def test\_delete\_provider(self):

rowcount = self.window.table\_provider.rowCount()

self.window.table\_provider.setCurrentCell(rowcount-1, 1, QItemSelectionModel.SelectionFlag.Select)

btn\_del = self.window.ui.btn\_del\_prov

QTest.mouseClick(btn\_del, QtCore.Qt.MouseButton.LeftButton)

QTest.mouseClick(self.window.ui.btn\_save\_prov, QtCore.Qt.MouseButton.LeftButton)

# **ПРИЛОЖЕНИЕ Д Скрипт для создания инсталлятора приложения**

; Script generated by the Inno Setup Script Wizard.

; SEE THE DOCUMENTATION FOR DETAILS ON CREATING INNO SETUP SCRIPT FILES!

[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={{E582337F-EAAF-4BB6-814A-B67E511FCF0E}

AppName=BookShop

AppVersion=1.0

;AppVerName=BookShop 1.0

AppPublisher=Makarova Darya

AppPublisherURL=https://github.com/ssmlnsk/BookShop

AppSupportURL=https://github.com/ssmlnsk/BookShop

AppUpdatesURL=https://github.com/ssmlnsk/BookShop

DefaultDirName={autopf}\BookShop

DefaultGroupName=BookShop

LicenseFile=C:\Users\ssmlnsk\Desktop\4 курс\BookShop\BookShop.rtf

; Uncomment the following line to run in non administrative install mode (install for current user only.)

;PrivilegesRequired=lowest

OutputBaseFilename=BookShopSetup

SetupIconFile=C:\Users\ssmlnsk\Desktop\4 курс\BookShop\img\logo.ico

Compression=lzma

SolidCompression=yes

WizardStyle=modern

UsePreviousAppDir=no

[Languages]

Name: "english"; MessagesFile: "compiler:Default.isl"

Name: "russian"; MessagesFile: "compiler:Languages\Russian.isl"

[Tasks]

Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked

[Files]

Source: "C:\Users\ssmlnsk\Desktop\4 курс\BookShop\dist\gui\gui.exe"; DestDir: "{app}"; Flags: ignoreversion

Source: "C:\Users\ssmlnsk\Desktop\4 курс\BookShop\dist\gui\\*"; DestDir: "{app}"; Flags: ignoreversion recursesubdirs createallsubdirs

; NOTE: Don't use "Flags: ignoreversion" on any shared system files

[Icons]

Name: "{group}\BookShop"; Filename: "{app}\gui.exe"

Name: "{group}\{cm:UninstallProgram,BookShop}"; Filename: "{uninstallexe}"

Name: "{autodesktop}\BookShop"; Filename: "{app}\gui.exe"; Tasks: desktopicon

[Run]

Filename: "{app}\gui.exe"; Description: "{cm:LaunchProgram,BookShop}"; Flags: nowait postinstall skipifsilent

**ПРИЛОЖЕНИЕ Е  
Содержимое базы данных**

CREATE DATABASE IF NOT EXISTS `bookshop` /\*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;

USE `bookshop`;

-- MySQL dump 10.13 Distrib 8.0.29, for Win64 (x86\_64)

--

-- Host: localhost Database: bookshop

-- ------------------------------------------------------

-- Server version 8.0.25

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `author`

--

DROP TABLE IF EXISTS `author`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

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` varchar(45) DEFAULT NULL,

PRIMARY KEY (`Код автора`)

) ENGINE=InnoDB AUTO\_INCREMENT=18 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `author`

--

LOCK TABLES `author` WRITE;

/\*!40000 ALTER TABLE `author` DISABLE KEYS \*/;

INSERT INTO `author` VALUES (1,'Кэнтаро','Миура','','11-07-1966'),(2,'Атанасян','Левон','Сергеевич','08-12-1921'),(3,'Кинг','Стивен','','21-09-1947'),(4,'Пушкин','Александр','Сергеевич','26-05-1799'),(5,'Полярный','Александр','','18-09-1994'),(6,'Толстой','Лев','Николаевич','09-09-1828');

/\*!40000 ALTER TABLE `author` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `books`

--

DROP TABLE IF EXISTS `books`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `books` (

`Код книги` int NOT NULL AUTO\_INCREMENT,

`Наименование книги` varchar(45) DEFAULT NULL,

`Год издания` int DEFAULT NULL,

`Количество страниц` int DEFAULT NULL,

`Жанр` int DEFAULT NULL,

`Автор` int DEFAULT NULL,

`Издательство` int DEFAULT NULL,

`Стоимость` float DEFAULT NULL,

PRIMARY KEY (`Код книги`),

KEY `books\_FK` (`Жанр`),

KEY `books\_FK\_1` (`Автор`),

KEY `books\_FK\_2` (`Издательство`)

) ENGINE=InnoDB AUTO\_INCREMENT=35 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `books`

--

LOCK TABLES `books` WRITE;

/\*!40000 ALTER TABLE `books` DISABLE KEYS \*/;

INSERT INTO `books` VALUES (4,'Мятная сказка',2018,208,5,5,3,599),(12,'Берсерк. Том 1',1989,458,8,1,8,1200),(13,'Оно',1986,1138,7,3,3,1000);

/\*!40000 ALTER TABLE `books` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `clients`

--

DROP TABLE IF EXISTS `clients`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `clients` (

`Код клиента` int NOT NULL AUTO\_INCREMENT,

`Фамилия` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Имя` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Отчество` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Дата рождения` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`e-mail` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

PRIMARY KEY (`Код клиента`)

) ENGINE=InnoDB AUTO\_INCREMENT=45462614 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `clients`

--

LOCK TABLES `clients` WRITE;

/\*!40000 ALTER TABLE `clients` DISABLE KEYS \*/;

INSERT INTO `clients` VALUES (45462526,'Фролов','Андрей','Иванович','2001-07-13','gohufreilagrau-3818@yopmail.com'),(45462527,'Николаев','Даниил','Всеволодович','2001-02-09','xawugosune-1385@yopmail.com'),(45462528,'Снегирев','Макар','Иванович','1998-05-20','satrahuddusei-4458@yopmail.com'),(45462529,'Иванов','Иван','Ильич','1998-09-30','boippaxeufrepra-7093@yopmail.com'),(45462530,'Филиппова','Анна','Глебовна','1976-05-30','zapramaxesu-7741@yopmail.com'),(45462531,'Иванов','Михаил','Владимирович','1985-11-03','rouzecroummegre-3899@yopmail.com'),(45462532,'Власов','Дмитрий','Александрович','1998-08-16','ziyeuddocrabri-4748@yopmail.com'),(45462533,'Серова','Екатерина','Львовна','1984-10-23','ketameissoinnei-1951@yopmail.com'),(45462534,'Борисова','Ирина','Ивановна','1976-10-13','yipraubaponou-5849@yopmail.com'),(45462535,'Зайцев','Никита','Артёмович','1999-10-13','crapedocouca-3572@yopmail.com'),(45462536,'Медведев','Святослав','Евгеньевич','1985-07-12','ceigoixakaunni-9227@yopmail.com'),(45462537,'Коротков','Кирилл','Алексеевич','1976-05-25','yeimmeiwauzomo-7054@yopmail.com'),(45462538,'Калашникова','Арина','Максимовна','1999-08-12','poleifenevi-1560@yopmail.com'),(45462539,'Минина','Таисия','Кирилловна','1985-10-12','kauprezofautei-6607@yopmail.com'),(45462540,'Наумов','Серафим','Романович','1999-04-14','quaffaullelourei-1667@yopmail.com'),(45462541,'Воробьева','Василиса','Евгеньевна','1999-01-12','jsteele@rojas-robinson.net'),(45462542,'Калинин','Александр','Андреевич','1999-01-06','vhopkins@lewis-mullen.com'),(45462543,'Кузнецова','Милана','Владиславовна','1999-01-23','nlewis@yahoo.com'),(45462544,'Фирсов','Егор','Романович','1993-09-01','garciadavid@mckinney-mcbride.com'),(45462545,'Зимина','Агния','Александровна','1998-09-02','cbradley@castro.com'),(45462546,'Титов','Андрей','Глебович','1985-10-22','cuevascatherine@carlson.biz'),(45462547,'Орлов','Николай','Егорович','1985-07-26','thomasmoore@wilson-singh.net'),(45462548,'Кузнецова','Аиша','Михайловна','1998-10-03','jessica84@hotmail.com'),(45462549,'Куликов','Никита','Георгиевич','1999-04-22','jessicapark@hotmail.com'),(45462550,'Карпова','София','Егоровна','1993-09-30','ginaritter@schneider-buchanan.com'),(45462551,'Смирнова','Дарья','Макаровна','1976-03-21','stephen99@yahoo.com'),(45462552,'Абрамова','Александра','Мироновна','1999-03-25','lopezlisa@hotmail.com'),(45462553,'Наумов','Руслан','Михайлович','1999-10-10','lori17@hotmail.com'),(45462554,'Бочаров','Никита','Матвеевич','1997-06-28','campbellkevin@gardner.com'),(45462555,'Соловьев','Давид','Ильич','1984-03-05','morganhoward@clark.com'),(45462556,'Васильева','Валерия','Дмитриевна','1999-09-29','carsontamara@gmail.com'),(45462557,'Макарова','Василиса','Андреевна','1999-04-07','kevinpatel@gmail.com'),(45462558,'Алексеев','Матвей','Викторович','1998-08-01','sethbishop@yahoo.com'),(45462559,'Никитина','Полина','Александровна','1976-09-18','drollins@schultz-soto.net'),(45462560,'Окулова','Олеся','Алексеевна','1999-04-02','pblack@copeland-winters.org'),(45462561,'Захарова','Полина','Яновна','1976-04-20','johnathon.oberbrunner@yahoo.com'),(45462562,'Зайцев','Владимир','Давидович','1998-01-25','bradly29@gmail.com'),(45462563,'Иванов','Виталий','Даниилович','1976-08-10','stark.cristina@hilpert.biz'),(45462564,'Захаров','Матвей','Романович','1993-07-11','bruen.eleanore@yahoo.com'),(45462565,'Иванов','Степан','Степанович','1998-09-18','percival.halvorson@yahoo.com'),(45462566,'Ткачева','Милана','Тимуровна','1998-05-23','javonte71@kuhlman.biz'),(45462567,'Семенов','Даниил','Иванович','1976-01-03','vconnelly@kautzer.com'),(45462568,'Виноградов','Вячеслав','Дмитриевич','1976-07-11','anabelle07@schultz.info'),(45462569,'Соболева','Николь','Фёдоровна','1976-05-01','nitzsche.katlynn@yahoo.com'),(45462570,'Тихонова','Анна','Львовна','1985-03-22','corine16@von.com'),(45462571,'Кузнецова','Ульяна','Савельевна','1999-06-02','otha.wisozk@lubowitz.org'),(45462572,'Смирнова','Анна','Германовна','1997-07-17','may.kirlin@hotmail.com'),(45462573,'Черепанова','Анна','Давидовна','1985-11-05','bryana.kautzer@yahoo.com'),(45462574,'Григорьев','Максим','Кириллович','1999-05-25','deborah.christiansen@quigley.biz'),(45462575,'Голубев','Даниэль','Александрович','1999-06-13','connelly.makayla@yahoo.com'),(45462576,'Миронов','Юрий','Денисович','1985-01-25','tatum.collins@fay.org'),(45462577,'Терехов','Михаил','Андреевич','1976-07-05','itzel73@anderson.com'),(45462578,'Орлова','Алиса','Михайловна','1997-02-23','arjun39@hotmail.com'),(45462579,'Кулаков','Константин','Даниилович','1993-06-19','ohara.rebeka@yahoo.com'),(45462580,'Кудрявцев','Максим','Романович','1998-05-09','danika58@rath.com'),(45462581,'Соболева','Кира','Фёдоровна','1998-03-13','janae.bogan@gmail.com'),(45462582,'Коновалов','Арсений','Максимович','1985-02-17','vern91@yahoo.com'),(45462583,'Гусев','Михаил','Дмитриевич','1999-11-22','mariana.leannon@larkin.net'),(45462584,'Суханова','Варвара','Матвеевна','1993-09-12','vmoore@gmail.com'),(45462585,'Орлова','Ясмина','Васильевна','1984-06-23','damon.mcclure@mills.com'),(45462586,'Васильева','Ксения','Константиновна','1999-07-31','grady.reilly@block.com'),(45462587,'Борисова','Тамара','Данииловна','1993-05-28','boyd.koss@yahoo.com'),(45462588,'Дмитриев','Мирон','Ильич','1985-04-12','obartell@franecki.info'),(45462589,'Лебедева','Анна','Александровна','1985-03-29','reina75@ferry.net'),(45462590,'Пономарев','Артём','Маркович','1984-06-01','karson28@hotmail.com'),(45462591,'Борисова','Елена','Михайловна','1976-05-22','damaris61@okon.com'),(45462592,'Моисеев','Камиль','Максимович','1999-06-16','carroll.jerod@hotmail.com'),(45462593,'Герасимова','Дарья','Константиновна','1984-10-12','ron.treutel@quitzon.com'),(45462594,'Михайлова','Мария','Марковна','1976-12-01','olen79@yahoo.com'),(45462595,'Коршунов','Кирилл','Максимович','1985-05-21','pacocha.robbie@yahoo.com');

/\*!40000 ALTER TABLE `clients` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `employees`

--

DROP TABLE IF EXISTS `employees`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `employees` (

`Код сотрудника` int NOT NULL AUTO\_INCREMENT,

`Фамилия` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Имя` varchar(100) DEFAULT NULL,

`Отчество` varchar(100) DEFAULT NULL,

`Должность` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Логин` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Пароль` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Последний вход` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Тип входа` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Фото` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

PRIMARY KEY (`Код сотрудника`)

) ENGINE=InnoDB AUTO\_INCREMENT=113 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `employees`

--

LOCK TABLES `employees` WRITE;

/\*!40000 ALTER TABLE `employees` DISABLE KEYS \*/;

INSERT INTO `employees` VALUES (101,'Иванов','Иван','Иванович','Продавец','Ivanov@namecomp.ru','2L6KZG','03.03.2023 23:39:56','Успешно','Иванов.jpeg'),(102,'Петров','Петр','Петрович','Продавец','petrov@namecomp.ru','uzWC67','03.03.2023 23:39:56','Успешно','Петров.jpeg'),(103,'Федоров','Федор','Федорович','Администратор','fedorov@namecomp.ru','8ntwUp','11.06.2023 21:07:29','Успешно','Федоров.jpeg'),(104,'Миронов','Вениамин','Куприянович','Старший смены','mironov@namecomp.ru','YOyhfR','10.06.2023 19:44:32','Успешно','Миронов.jpeg'),(105,'Ширяев','Ермолай','Вениаминович','Старший смены','shiryev@namecomp.ru','RSbvHv','03.03.2023 23:39:56','Неуспешно','Ширяев.jpeg'),(106,'Игнатов','Кассиан','Васильевич','Старший смены','ignatov@namecomp.ru','rwVDh9','03.03.2023 23:39:56','Успешно','Игнатов.jpg'),(107,'Хохлов','Владимир','Мэлсович','Продавец','hohlov@namecomp.ru','LdNyos','03.03.2023 23:39:56','Успешно','Хохлов.jpeg'),(108,'Стрелков','Мстислав','Георгьевич','Продавец','strelkov@namecomp.ru','gynQMT','03.03.2023 23:39:56','Успешно','Стрелков.jpeg'),(109,'Беляева','Лилия','Наумовна','Продавец','belyeva@namecomp.ru','AtnDjr','03.03.2023 23:39:56','Успешно','Беляева.jpeg'),(110,'Смирнова','Ульяна','Гордеевна','Продавец','smirnova@namecomp.ru','JlFRCZ','03.03.2023 23:39:56','Успешно','Смирнова.jpeg'),(111,'test','test','test','Администратор','1','1','12.06.2023 00:26:42','Успешно','Смирнова.jpeg'),(112,'test2','test2','test2','Продавец','2','2','11.06.2023 17:43:30','Успешно','Беляева.jpeg');

/\*!40000 ALTER TABLE `employees` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `genre`

--

DROP TABLE IF EXISTS `genre`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `genre` (

`Код жанра` int NOT NULL AUTO\_INCREMENT,

`Жанр` varchar(45) DEFAULT NULL,

PRIMARY KEY (`Код жанра`)

) ENGINE=InnoDB AUTO\_INCREMENT=39 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `genre`

--

LOCK TABLES `genre` WRITE;

/\*!40000 ALTER TABLE `genre` DISABLE KEYS \*/;

INSERT INTO `genre` VALUES (1,'Классика'),(2,'Фантастика'),(3,'Детектив'),(4,'Учебная литература'),(5,'Сказка'),(6,'Фэнтези'),(7,'Роман'),(8,'Манга'),(9,'Мемуары'),(10,'Религия'),(11,'Психология'),(12,'Философия'),(13,'Детская литература'),(14,'Комиксы'),(15,'Проза'),(16,'Поэзия'),(17,'Драматургия'),(18,'Художественная литература'),(19,'Экономия'),(20,'Право'),(21,'Справочник'),(22,'Медицина');

/\*!40000 ALTER TABLE `genre` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `history`

--

DROP TABLE IF EXISTS `history`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `history` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Дата входа` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Дата выхода` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Блокировка` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT 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`)

) ENGINE=InnoDB AUTO\_INCREMENT=797 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `history`

--

LOCK TABLES `history` WRITE;

/\*!40000 ALTER TABLE `history` DISABLE KEYS \*/;

INSERT INTO `history` VALUES (453,'10.06.2023 18:33:06',NULL,'Успешно',103),(454,NULL,'10.06.2023 18:35:48','Неуспешно',103),(455,'10.06.2023 18:39:13',NULL,'Успешно',102),(456,NULL,'10.06.2023 18:39:22','Неуспешно',102),(631,'11.06.2023 15:16:47',NULL,'Успешно',103),(632,NULL,'11.06.2023 15:16:51','Неуспешно',103),(674,'11.06.2023 15:32:20',NULL,'Успешно',103),(675,'11.06.2023 15:32:29',NULL,'Успешно',103),(676,NULL,'11.06.2023 15:32:30','Неуспешно',103),(677,'11.06.2023 15:32:57',NULL,'Успешно',103),(678,'11.06.2023 15:33:08',NULL,'Успешно',103),(679,NULL,'11.06.2023 15:33:10','Неуспешно',103),(680,'11.06.2023 15:33:51',NULL,'Успешно',103),(681,NULL,'11.06.2023 15:33:58','Неуспешно',103),(682,'11.06.2023 15:34:12',NULL,'Успешно',103),(683,'11.06.2023 15:34:36',NULL,'Успешно',103),(684,NULL,'11.06.2023 15:34:37','Неуспешно',103),(685,'11.06.2023 15:34:39',NULL,'Успешно',103),(686,'11.06.2023 15:38:48',NULL,'Успешно',103),(687,NULL,'11.06.2023 15:38:49','Неуспешно',103),(688,'11.06.2023 15:38:51',NULL,'Успешно',103),(689,'11.06.2023 15:39:28',NULL,'Успешно',103),(690,NULL,'11.06.2023 15:39:29','Неуспешно',103);

/\*!40000 ALTER TABLE `history` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `provider`

--

DROP TABLE IF EXISTS `provider`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `provider` (

`Код поставщика` int NOT NULL AUTO\_INCREMENT,

`Наименование компании` varchar(100) NOT NULL,

`Юридический адрес` varchar(100) NOT NULL,

`Номер телефона` varchar(100) NOT NULL,

PRIMARY KEY (`Код поставщика`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `provider`

--

LOCK TABLES `provider` WRITE;

/\*!40000 ALTER TABLE `provider` DISABLE KEYS \*/;

INSERT INTO `provider` VALUES (1,'Творческая пристань','142408, область МОСКОВСКАЯ, НОГИНСК, ул. МОСКОВСКАЯ, 9, 30',''),(4,'Арт принт','125009, г. МОСКВА, СТРАСТНОЙ бульвар, д. 4/3, стр. 1',''),(5,'Книжный клуб 36,6','Москва. ул. Бакунинская, д. 71, стр. 10','+7(495)926-45-44');

/\*!40000 ALTER TABLE `provider` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `provider\_order`

--

DROP TABLE IF EXISTS `provider\_order`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `provider\_order` (

`Код поставки` int NOT NULL AUTO\_INCREMENT,

`Поставщик` int NOT NULL,

`Дата создания` date NOT NULL,

`Книга` int NOT NULL,

`Количество` int NOT NULL,

`Стоимость` float NOT NULL,

PRIMARY KEY (`Код поставки`),

KEY `book\_idx` (`Книга`),

KEY `provider\_idx` (`Поставщик`),

CONSTRAINT `book` FOREIGN KEY (`Книга`) REFERENCES `books` (`Код книги`),

CONSTRAINT `provider` FOREIGN KEY (`Поставщик`) REFERENCES `provider` (`Код поставщика`)

) ENGINE=InnoDB AUTO\_INCREMENT=20 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `provider\_order`

--

LOCK TABLES `provider\_order` WRITE;

/\*!40000 ALTER TABLE `provider\_order` DISABLE KEYS \*/;

INSERT INTO `provider\_order` VALUES (17,1,'2023-06-12',12,100,120000),(18,1,'2023-06-12',13,140,140000),(19,4,'2023-06-12',4,50,29950);

/\*!40000 ALTER TABLE `provider\_order` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `publishing\_house`

--

DROP TABLE IF EXISTS `publishing\_house`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `publishing\_house` (

`Код издательства` int NOT NULL AUTO\_INCREMENT,

`Название издательства` varchar(100) DEFAULT NULL,

PRIMARY KEY (`Код издательства`)

) ENGINE=InnoDB AUTO\_INCREMENT=16 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `publishing\_house`

--

LOCK TABLES `publishing\_house` WRITE;

/\*!40000 ALTER TABLE `publishing\_house` DISABLE KEYS \*/;

INSERT INTO `publishing\_house` VALUES (1,'Bubble'),(2,'Эксмо'),(3,'ACT'),(4,'Просвещение'),(5,'Дрофа'),(6,'Питер'),(7,'Росмэн'),(8,'Square Enix'),(9,'Перо'),(10,'Фламинго');

/\*!40000 ALTER TABLE `publishing\_house` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `requests`

--

DROP TABLE IF EXISTS `requests`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `requests` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Код заказа` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Дата создания` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Время заказа` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`Клиент` int DEFAULT NULL,

`Книга` int DEFAULT NULL,

`Сотрудник` int DEFAULT 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`)

) ENGINE=InnoDB AUTO\_INCREMENT=45 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `requests`

--

LOCK TABLES `requests` WRITE;

/\*!40000 ALTER TABLE `requests` DISABLE KEYS \*/;

INSERT INTO `requests` VALUES (24,'35256230/2023.02.28','2023.02.28','21:51',45462526,4,101),(25,'35256230/2023.02.28','2023.02.28','21:51',45462526,4,101),(26,'35256230/2023.02.28','2023.02.28','21:51',45462526,4,101),(27,'35256230/2023.02.28','2023.02.28','21:51',45462526,4,101),(28,'44464363/2023.03.01','2023.03.01','10:13',45462526,4,101),(29,'44464363/2023.03.01','2023.03.01','10:13',45462526,4,101),(30,'44464363/2023.03.01','2023.03.01','10:13',45462526,4,101),(31,'44464363/2023.03.01','2023.03.01','10:13',45462526,13,101),(32,'0/2023.03.01','2023.03.01','10:15',45462526,4,101),(33,'0/2023.03.01','2023.03.01','10:15',45462526,4,101),(36,'0/2023.03.01','2023.03.01','10:19',45462526,12,101),(37,'0/2023.03.01','2023.03.01','10:19',45462526,13,101),(41,'45462526/2023.03.04','2023.03.04','10:19',45462526,13,101),(42,'45462526/2023.03.07','2023.03.07','12:49',45462526,4,102),(44,'45462526/2023.06.11','2023.06.11','18:28',45462526,13,111);

/\*!40000 ALTER TABLE `requests` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2023-06-12 0:52:20