**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное  
образовательное учреждение высшего образования**

**«Тульский государственный университет»**

Кафедра информационной безопасности

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе по дисциплине

**«БАЗЫ ДАННЫХ»**

на тему

Прикладное программное обеспечение «Бюро технической инвентаризации»

Автор работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ студент гр. \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись) (фамилия и инициалы)

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись) (должность) (фамилия и инициалы)

Работа защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись) (должность) (фамилия и инициалы)

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

(дата, подпись) (должность) (фамилия и инициалы)

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

(дата, подпись) (должность) (фамилия и инициалы)

Тула 20\_\_\_

Утверждаю

Зав. кафедрой ПМиИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Сычугов

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.

**ЗАДАНИЕ**

на курсовую работу по дисциплине

**«БАЗЫ ДАННЫХ»**

студенту гр. 221081 Чичикину Ивану Сергеевичу

(фамилия, имя, отчество)

Тема работы

Прикладное программное обеспечение «Бюро технической инвентаризации»

Входные данные \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

Задание получил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (дата)

График выполнения работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

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

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

Замечания консультанта

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

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

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

К защите. Консультант работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (дата)

**РЕЦЕНЗИЯ**

*на курсовую работу (курсовой проект) студента Тульского государственного университета,*

гр. 221081 Чичикина Ивана Сергеевича

(фамилия, имя, отчество)

по дисциплине Базы данных

на тему: Прикладное программное обеспечение «Бюро технической инвентаризации».

Тематика работы (задание) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ профилю (направленности)

(соответствует/не соответствует)

образовательной программы, а также в полной мере способствует формированию необходимых компетенций (установленных в рабочей программе) у обучающихся.

Содержание работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ заданной тематике.

(соответствует/не соответствует)

Объем работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ для раскрытия темы.

(достаточный/недостаточный)

Оформление работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ установленным требованиям.

(соответствует/не соответствует)

Тема работы раскрыта в \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ мере.

(полной/неполной)

Использованная при выполнении работы (проекта) литература \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

(актуальна/не актуальна)

Качество приложений (при наличии)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ установленным

(соответствует/не соответствует)

требованиям.

**Замечания:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(отсутствуют или перечисляются замечания рецензента к работе)

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

Качество выполнения работы свидетельствует о \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ уровне (недостаточном/пороговом/повышенном)

сформированности необходимых компетенций (установленных в рабочей программе).

Работа выполнена на \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ уровне и, при соответствующей

(высоком/среднем/низком)

защите может быть оценена «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_».

**Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /**

**подпись должность, звание, Ф.И.О.**

**Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Содержание**

[Введение 4](#_Toc104835136)

[Раздел 1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 7](#_Toc104835137)

[1.1 Назначение разработки 7](#_Toc104835138)

[1.2 Требования к программе 8](#_Toc104835139)

[1.3 Стадии и этапы разработки 9](#_Toc104835140)

[Раздел 2 ОПИСАНИЕ ПРОГРАММЫ 10](#_Toc104835141)

[2.1 Функциональное назначение 10](#_Toc104835142)

[2.2 Описание логической структуры 11](#_Toc104835143)

[Раздел 3 ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ ПРОГРАММЫ 29](#_Toc104835144)

[3.1 Назначение программы 29](#_Toc104835145)

[3.2 Выполнение программы 29](#_Toc104835146)

[ЗАКЛЮЧЕНИЕ 39](#_Toc104835147)

[Список использованных источников 40](#_Toc104835148)

[ПРИЛОЖЕНИЕ 41](#_Toc104835149)

Введение

Объектно-ориентированное программирование — это методология программирования, которая основана на представлении программы в виде совокупности объектов, каждый из которых является реализацией определенного класса.

JavaFX — платформа на основе [Java](https://ru.wikipedia.org/wiki/Java) для создания [приложений](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) с насыщенным [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F). Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под [операционных систем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), так и для интернет-приложений ([RIA](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%8B%D1%89%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)), работающих в [браузерах](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80), и для приложений на мобильных устройствах. JavaFX призвана заменить использовавшуюся ранее библиотеку [Swing](https://ru.wikipedia.org/wiki/Swing" \o "Swing). Платформа JavaFX конкурирует с [Microsoft Silverlight](https://ru.wikipedia.org/wiki/Microsoft_Silverlight), [Adobe Flash](https://ru.wikipedia.org/wiki/Adobe_Flash) и аналогичными системами.

Начиная с версии Java 11 больше не входит в Java SE и не разрабатывается компанией Oracle (как отдельный модуль поддерживается компанией Gluon). Oracle будет поддерживать JavaFX в составе Java JDK 8 до марта 2025.

Scene Builder — это инструмент, с помощью которого можно конструировать окна в виде графического интерфейса и после их сохранять, и эта программа на основании результата будет конструировать fxml-файлы, которые можно использовать в JavaFX-приложении.

IntelliJ IDEA — [интегрированная среда разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) для многих [языков программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), в частности [Java](https://ru.wikipedia.org/wiki/Java), [JavaScript](https://ru.wikipedia.org/wiki/JavaScript), [Python](https://ru.wikipedia.org/wiki/Python), разработанная компанией [JetBrains](https://ru.wikipedia.org/wiki/JetBrains" \o "JetBrains).

Первая версия появилась в январе [2001 года](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для [рефакторинга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3), которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с шестой версии продукта IntelliJ IDEA предоставляет интегрированный инструментарий для разработки [графического пользовательского интерфейса](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F). Среди прочих возможностей, среда хорошо совместима со многими популярными [свободными](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) инструментами разработчиков, такими как [CVS](https://ru.wikipedia.org/wiki/Concurrent_Versions_System), [Subversion](https://ru.wikipedia.org/wiki/Subversion" \o "Subversion), [Apache Ant](https://ru.wikipedia.org/wiki/Apache_Ant), [Maven](https://ru.wikipedia.org/wiki/Maven" \o "Maven) и [JUnit](https://ru.wikipedia.org/wiki/JUnit" \o "JUnit). В феврале 2007 года разработчики IntelliJ анонсировали раннюю версию плагина для поддержки программирования на языке [Ruby](https://ru.wikipedia.org/wiki/Ruby).

Начиная с версии 9.0 среда доступна в двух редакциях: Community Edition и Ultimate Edition. Community Edition является полностью [свободной](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) версией, доступной под лицензией [Apache 2.0](https://ru.wikipedia.org/wiki/Apache_License), в ней реализована полная поддержка [Java SE](https://ru.wikipedia.org/wiki/Java_SE), [Kotlin](https://ru.wikipedia.org/wiki/Kotlin" \o "Kotlin), [Groovy](https://ru.wikipedia.org/wiki/Groovy" \o "Groovy), [Scala](https://ru.wikipedia.org/wiki/Scala_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), а также интеграция с наиболее популярными [системами управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8). В редакции Ultimate Edition, доступной под коммерческой лицензией, реализована поддержка [Java EE](https://ru.wikipedia.org/wiki/Java_EE), [UML](https://ru.wikipedia.org/wiki/UML)-диаграмм, подсчёт [покрытия кода](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0), а также поддержка других систем управления версиями, языков и [фреймворков](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA).

SQL ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) structured query language — «язык структурированных запросов») — [декларативный](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), применяемый для создания, модификации и управления данными в [реляционной базе данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), управляемой соответствующей [системой управления базами данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Является, прежде всего, [информационно-логическим языком](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), предназначенным для описания, изменения и извлечения данных, хранимых в [реляционных базах данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). В общем случае SQL (без ряда современных расширений) считается [языком программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) не [полным по Тьюрингу](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B0_%D0%BF%D0%BE_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D1%83), но вместе с тем стандарт языка спецификацией [SQL/PSM](https://ru.wikipedia.org/wiki/SQL/PSM) предусматривает возможность его [процедурных расширений](https://ru.wikipedia.org/wiki/SQL#%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D1%8B%D0%B5_%D1%80%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%B8%D1%8F).

MySQL Workbench — инструмент для визуального проектирования [баз данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных [MySQL](https://ru.wikipedia.org/wiki/MySQL). Является преемником [DBDesigner 4](https://ru.wikipedia.org/w/index.php?title=DBDesigner&action=edit&redlink=1" \o "DBDesigner (страница отсутствует)) от [FabForce](https://ru.wikipedia.org/w/index.php?title=FabForce&action=edit&redlink=1" \o "FabForce (страница отсутствует)).

MySQL Workbench предлагается в трёх редакциях:

Community Edition — распространяется под свободной лицензией [GNU GPL](https://ru.wikipedia.org/wiki/GNU_GPL);

Standard Edition — доступна по ежегодной оплачиваемой подписке. Эта версия включает в себя дополнительные функции, которые повышают производительность разработчиков и администраторов БД.

Enterprise Edition.

Раздел 1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1 Назначение разработки

Прикладное программное обеспечение «Бюро технической инвентаризации» предназначено для приёма заявок на техническую инвентаризацию.

1.2 Требования к программе

Разработать прикладное программное обеспечение деятельности «Бюро технической инвентаризации» по изготовлению и выдаче технических паспортов на объекты недвижимости. Перед регистрацией сделки с объектом недвижимости собственник объекта должен получить в БТИ на него технический паспорт. Ежедневно в БТИ обращается до 200 человек. Основное назначение программного комплекса — не пропустить ни одного документа. Если технический паспорт не готов в назначенный срок, то БТИ должно выплатить неустойку.

Алгоритм изготовления документа следующий. Клиент обращается к инспектору, сдает ему необходимые справки, согласовывает дату выхода техника на обмер, уплачивает аванс. Инспектор передает заявку начальнику отдела. Начальник отдела назначает исполнителя и техника. Техник выполняет обмер объекта. Исполнитель изготавливает документ и передает в отдел выдачи. В назначенный срок клиент забирает готовый документ, доплатив недостающую сумму. Один клиент (физическое или юридическое лицо) может заказать несколько технических паспортов, за изготовление которых оплата может производиться частями.

Набор данных:

* Номер заявки на изготовление документа
* ФИО заказчика
* Телефон для связи с заказчиком
* Физическое или юридическое лицо
* Банк заказчика
* Номер счёта в банке
* Адрес объекта
* Район города
* Дата приёма заявки
* Название документа
* Срочное изготовление (да/нет)
* Дата выдачи документа
* Стоимость изготовления документа
* Дата выхода техника
* Время выхода техника
* ФИО начальника отдела
* ФИО исполнителя
* Дата передачи исполнителю
* ФИО техника
* Оплата наличными (да/нет)
* Дата оплаты
* Оплаченная сумма
* Документ выдан (да/нет)
* Примечания

1.3 Стадии и этапы разработки

**Стадия 1. Проектирование**

Этапы:

1. Утверждение языков программирования JavaFX и SQL;
2. Утверждение использования MySQL Workbench;
3. Согласование с руководителем.

**Стадия 2. Создание**

Этапы:

1. Создание макета;
2. Написание кода;
3. Тестирование;

Раздел 2 ОПИСАНИЕ ПРОГРАММЫ

2.1 Функциональное назначение

В прикладном программном обеспечении «Бюро технической инвентаризации» реализована запись в таблицы базы данных посредством SQL-запросов. Основное назначение программы – получение данных из полей ввода приложения и, в зависимости от значений определённых полей, запись их в соответствующие таблицы базы данных, а также в общую таблицу.

Функциональные возможности приложения:

* Запись данных в поля ввода с определёнными ограничениями;
* Переход между полями ввода и меню нажатием клавиши Enter;
* Запись в базу данных нажатием кнопки;
* Очистка полей ввода нажатием кнопки;
* Получение уведомления об успешной записи или о незаполнении всех полей ввода.

Принцип работы программы:

Программа представляет собой стандартное оконное Windows-приложение. Осуществляет взаимодействие с пользователем при помощи различных элементов управления: графических кнопок, полей ввода, меню, информационных текстовых сообщений.

Оператор вводит данные заказа в поля ввода приложения, нажимает кнопку «Сохранить» и информация записывается в таблицу всех заявок базы данных и в таблицы, соответствующие выбранным пунктам некоторых меню.

2.2 Описание логической структуры

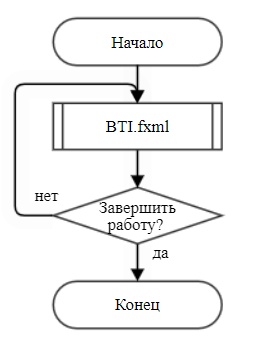


Рисунок 1 – Схема алгоритма Main.java

При запуске приложения вызывается класс Main.java, который загружает FXML-файл с макетом приложения:

Main.java

FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("BTI.fxml"));

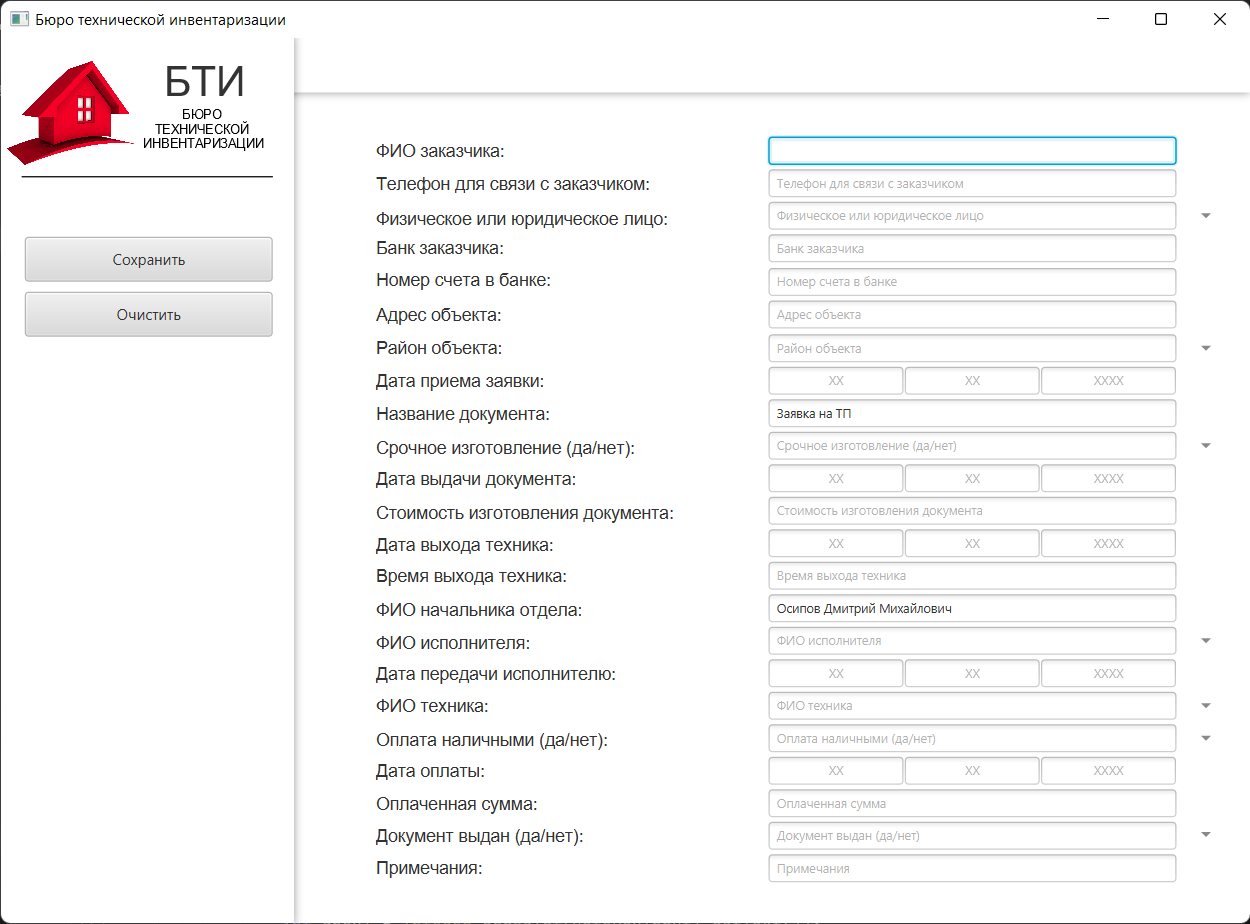


Рисунок 2 – Окно приложения.

Поле запуска приложения поле ввода «ФИО заказчика» автоматически становится активным:

Controller.java

Platform.runLater(FIOzakInput::requestFocus);

Для полей ввода созданы специальные ограничения ввода.

Ввод только кириллицы:

Controller.java

FIOzakInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я")) {  
 FIOzakInput.setText(newValue.replaceAll("[^\\sа-яА-Я ё]", ""));  
 }  
});

Ввод кириллицы и латиницы:

Controller.java

BankInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Яa-zA-Z\*")) {  
 BankInput.setText(newValue.replaceAll("[^\\sа-яА-Яa-zA-Z ё]", ""));  
 }  
});

Ввод только цифр, пробел заменяется на запятую:

Controller.java

PriceDocInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*")) {  
 PriceDocInput.setText(newValue.replaceAll("[^\\d]", ","));  
 }  
});

Ввод только цифр, пробел заменяется на двоеточие:

Controller.java

TimeTechInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*")) {  
 TimeTechInput.setText(newValue.replaceAll("[^\\d]", ":"));  
 }  
});

По нажатию клавиши Enter осуществляется переход между полями ввода и меню, что позволяет использовать только клавиатуру в процессе записи данных в приложение, не тратя время на компьютерную мышь:

Controller.java

FIOzakInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 PhoneInput.requestFocus();  
 }  
});

PhoneInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 FizUrMenu.requestFocus();  
 FizUrMenu.show();  
 }

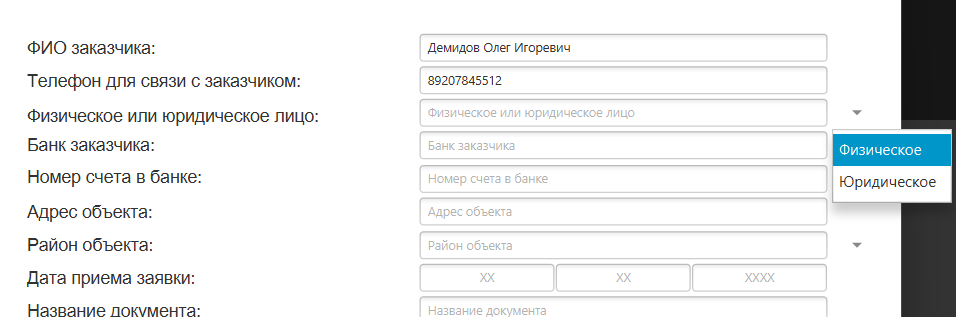


Рисунок 3 – Окно приложения. Меню.

Выбор пункта меню может осуществляться клавишами со стрелками и клавишей Enter.

После нажатия клавиши Enter происходит автоматическая запись текста выбранного пункта в поле ввода и переход к следующему полю ввода или меню:

Controller.java

MenuItem item1 = new MenuItem("Физическое");  
MenuItem item2 = new MenuItem("Юридическое");  
  
FizUrMenu.getItems().addAll(item1, item2);  
  
item1.setOnAction(event -> {  
 FizUrInput.setText("Физическое");  
 BankInput.requestFocus();  
});  
item2.setOnAction(event -> {  
 FizUrInput.setText("Юридическое");  
 BankInput.requestFocus();  
});

Прежде чем перейти к созданию связи приложения с базой данных, создадим саму базу данных. Для этого воспользуемся приложением MySQL Workbench.

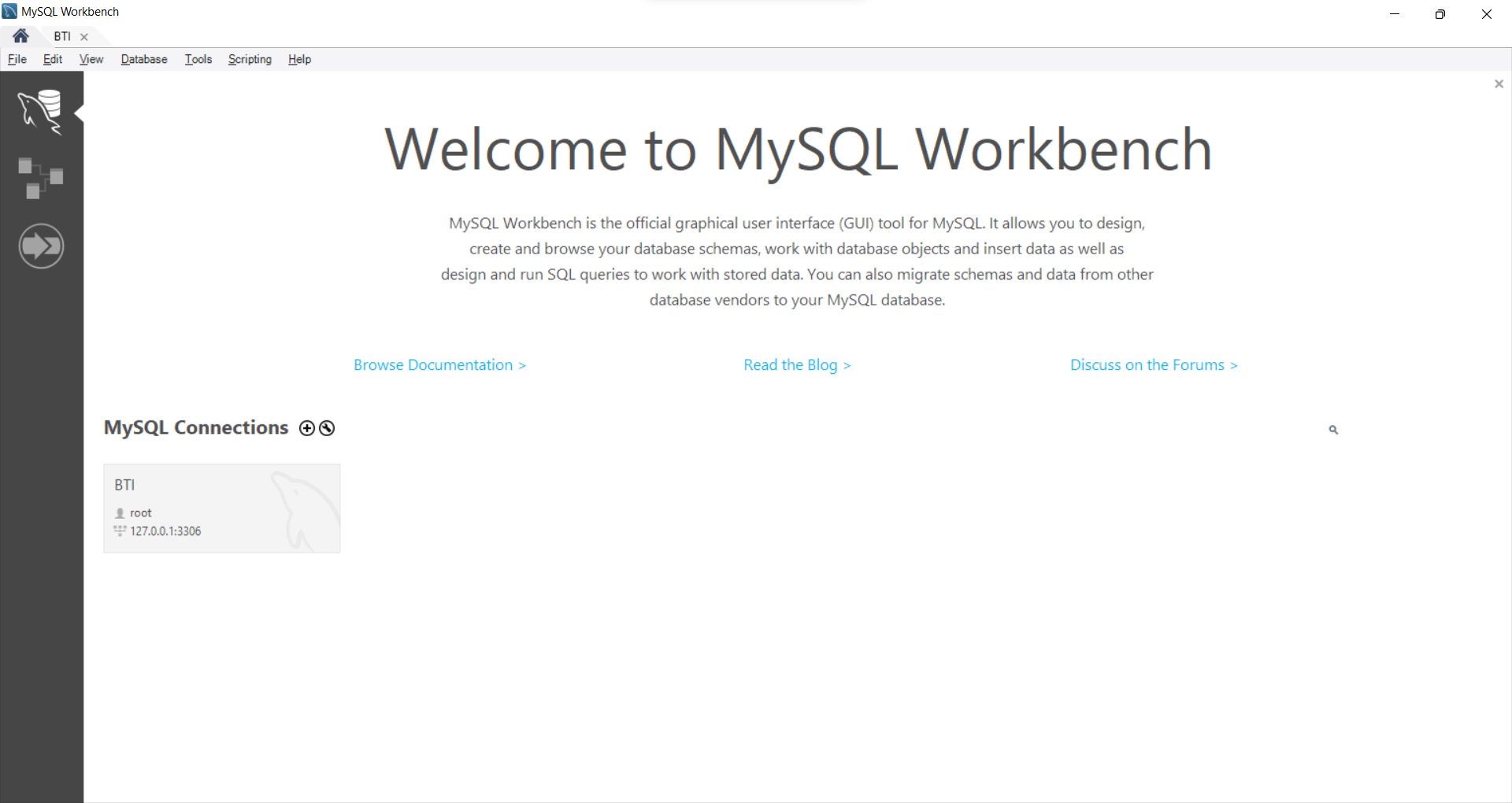


Рисунок 4 – Окно приложения MySQL Workbench.

Перейдём в окно создания базы данных нажатием на  и проведём настройку соединения.

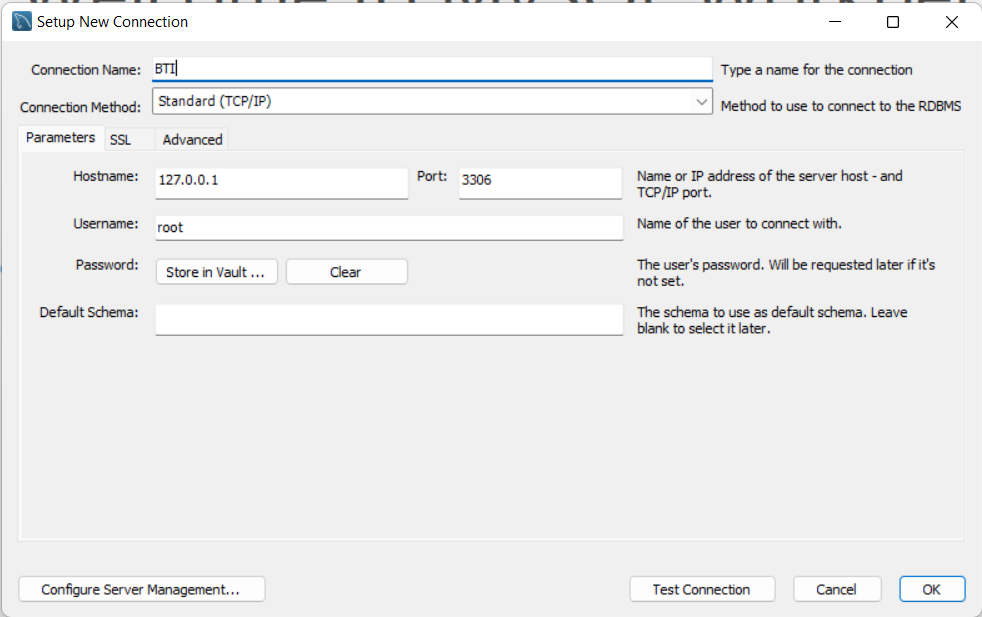


Рисунок 5 – Окно настройки соединения.

Нажатием на «Test Connection» проверим соединение.

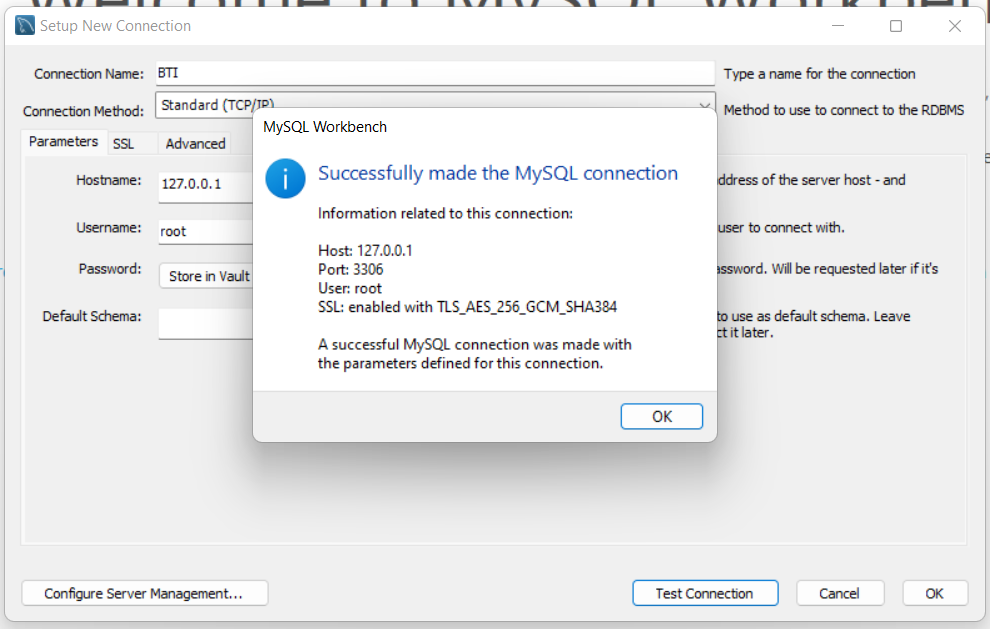


Рисунок 6 – Проверка соединения.

Наблюдаем сообщение об успешном соединении и нажимаем «Ок».

Во вкладке базы данных создаём новую схему, даём ей название и нажимаем «Apply».

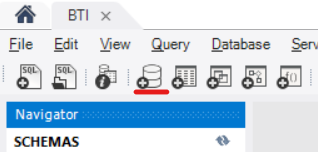


Рисунок 7 – Кнопка создания новой схемы.

Далее переходим в окно создания модели нужных нам таблиц.

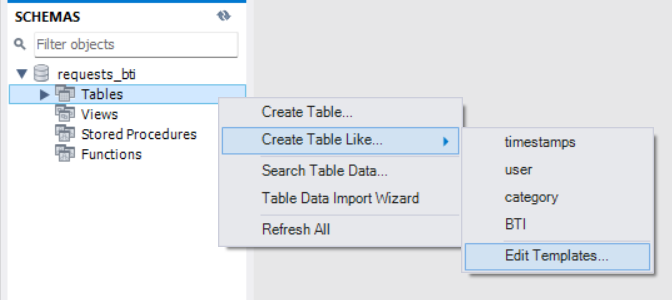


Рисунок 8 – Контекстное меню таблиц.

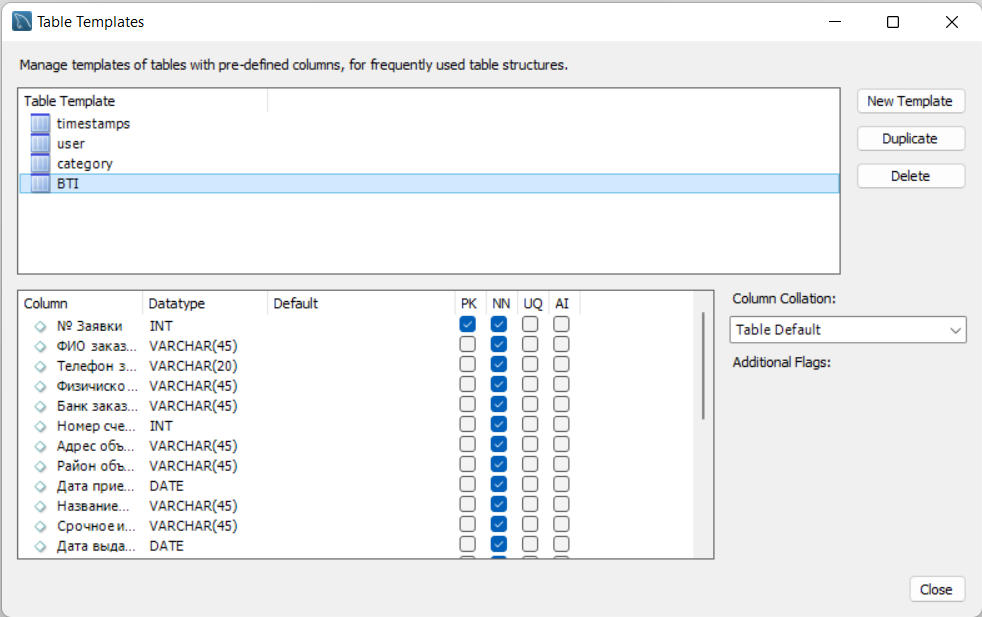


Рисунок 9 – Окно создания модели таблиц.

Нажимаем на «New Template», даём название модели и создаём необходимые столбцы с типами данных.

Затем через контекстное меню таблиц создаём таблицы по модели, указывая уникальные названия.

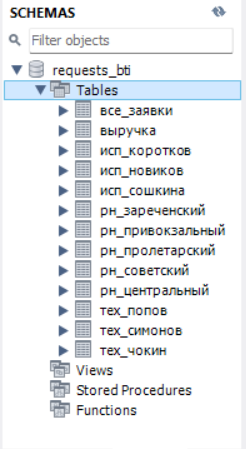


Рисунок 10 – Список таблиц.

Также для подсчёта выручки необходимо создать дополнительную таблицу «выручка» со следующим SQL-кодом:

SELECT SUM(`Оплаченная сумма`)

FROM requests\_bti.все\_заявки

WHERE `Дата оплаты`>= '2022.01.01' AND `Дата оплаты`<= '2022.12.31';

База данных создана. Теперь возвращаемся к JavaFX-проекту и создаём класс Configs.java, где указываем все характеристики соединения с базой данных:

Configs.java

protected String dbHost = "127.0.0.1";  
protected String dbPort = "3306";  
protected String dbUser = "root";  
protected String dbPass = "1234";  
protected String dbName = "requests\_bti";

Затем создаём класс-константу для соединения с таблицей «все\_заявки».

request\_const.java

public class requests\_const {  
 public static final String REQUESTS\_TABLE = "все\_заявки";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

По аналогии создаём классы соединения с другими таблицами.

В классе Controller.java создаём обработчики нажатия кнопок «Очистить» и «Сохранить».

Controller.java

CleanButton.setOnAction(event -> {  
 FIOzakInput.setText(""); PhoneInput.setText("");

...  
  
 ErrorText.setVisible(false);  
 DoneText.setVisible(false);  
  
 numReqText.setVisible(false);  
  
 FIOzakInput.requestFocus();  
 });

Кнопка «Очистить» стирает текст в полях ввода, а точнее – замещает его на пустоту.

Controller.java

SaveButton.setOnAction(event -> {

…

}

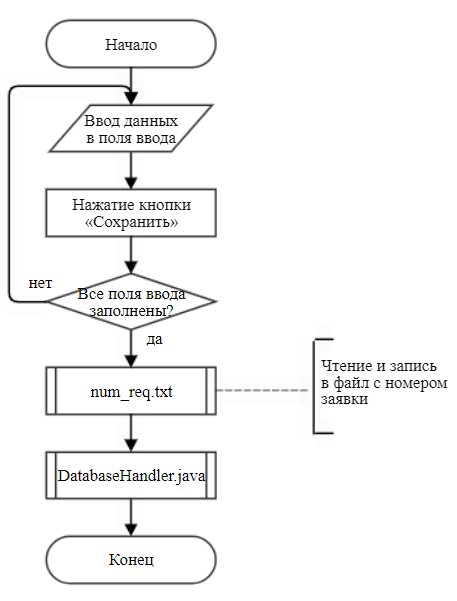


Рисунок 11 – Схема алгоритма Controller.java (SaveButton).

Особенностью SQL является зеркальное отображение дат, поэтому из всех полей ввода дат получаем текст и записываем в отдельные переменные. Пример:

Controller.java

String datereq1 = DateRequestInput1.getText();  
String datereq2 = DateRequestInput2.getText();  
String datereq3 = DateRequestInput3.getText();

Затем получаем текст из других полей ввода и записываем в переменные, используя trim() для удаления лишних пробелов, а значения дат записываем в переменные в обратном порядке через точку. Пример:

Controller.java

String DistrictText = DistrictInput.getText().trim();  
String DateRequestText = datereq3 + "." + datereq2 + "." + datereq1;

Далее создаём оператор if-else для вывода сообщений пользователю об успешной записи или о том, что заполнены не все поля ввода.

Controller.java

if(!FIOzakText.equals("") && !PhoneText.equals("") && !FizUrText.equals("") && ...) // указываем все переменные со значениями из полей ввода

{

... //вся логика

ErrorText.setVisible(false);  
 DoneText.setVisible(true);  
 System.out.println("Пользователь добавлен в базу данных");  
  
  
} else {  
 ErrorText.setVisible(true);  
 DoneText.setVisible(false);  
 System.out.println("Не все поля заполнены");  
}

Если все поля ввода заполнены, по нажатию графической кнопки «Сохранить» происходит запись заявки в таблицы базы данных и выводится сообщение об успешной записи и номер заявки, иначе выводится сообщение о том, что не все поля ввода заполнены.

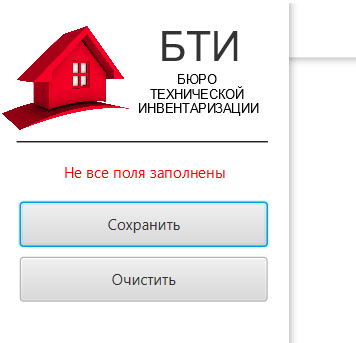
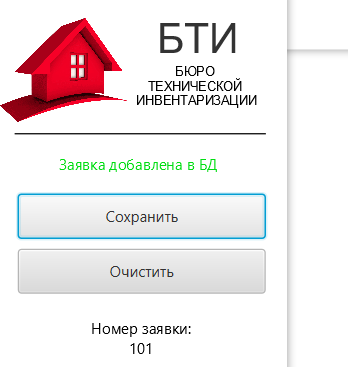


Рисунок 12 – Окно приложения. Сообщения.

Для формирования и отображения номеров заявок создаём текстовый файл num\_req.txt с текстом «0».

Controller.java

File file = new File("src/main/java/com/example/bti/num\_req.txt");  
  
FileReader fr = null;  
try {  
 fr = new FileReader(file);  
} catch (FileNotFoundException e) {  
 e.printStackTrace();  
}  
BufferedReader reader = new BufferedReader(fr);  
String stringLine = null;  
try {  
 stringLine = reader.readLine();  
} catch (IOException e) {  
 e.printStackTrace();  
}  
int intLine = Integer.parseInt(stringLine);  
intLine++;  
String numReq = Integer.toString(intLine);  
try {  
 reader.close();  
} catch (IOException e) {  
 e.printStackTrace();  
}  
  
FileWriter fw = null;  
try {  
 fw = new FileWriter(file);  
} catch (IOException e) {  
 e.printStackTrace();  
}  
BufferedWriter writer = new BufferedWriter(fw);  
try {  
 writer.write(numReq);  
} catch (IOException e) {  
 e.printStackTrace();  
}  
try {  
 writer.close();  
} catch (IOException e) {  
 e.printStackTrace();  
}  
numReqText.setText("Номер заявки:\n" + numReq);  
numReqText.setVisible(true);

Принцип работы следующий: открываем файл, читаем первую строку файла, увеличиваем число в строке на единицу, записываем значение в переменную, которую используем для вывода значения пользователю и отправки в базу данных.

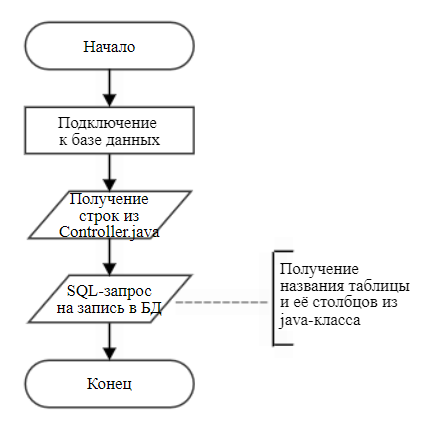


Рисунок 13 – Схема алгоритма DatabaseHandler.java

Создаём обработчик базы данных DatabaseHandler и организуем пересылку данных в класс DatabaseHandler.java для каждой таблицы.

Пример пересылки данных для таблицы «все\_заявки»:

DatabaseHandler dbHandler = new DatabaseHandler();

dbHandler.addRequest(numReq, FIOzakText, PhoneText, FizUrText,  
BankText, NumBankText, AdressText, DistrictText, DateRequestText,DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);

Создаём класс DatabaseHandler.java, наследуем класс Configs.java и программируем соединение.

DatabaseHandler.java

public class DatabaseHandler extends Configs {  
 Connection dbConnection;  
  
 public Connection getDbConnection() throws ClassNotFoundException, SQLException{  
 String connectionString = "jdbc:mysql://" + dbHost + ":" + dbPort + "/" + dbName + "?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";  
  
 Class.forName("com.mysql.cj.jdbc.Driver");  
  
 dbConnection = DriverManager.getConnection(connectionString, dbUser, dbPass);  
  
 return dbConnection;  
 }

...

}

Далее прописываем обработчик таблицы «все\_заявки» базы данных с SQL-запросом.

Берём строки из Controller.java:

DatabaseHandler.java

void addRequest(String numReq, String fiOzakText,

String phoneText,String fizUrText, String bankText,

String numBankText, String adressText, String districtText, String dateRequestText, String docNameText, String srochnoeText, String dateVidachiText, String priceDocText,

String dateTechText, String timeTechText, String fiOnachText, String fiOispText, String datePeredachiText, String fioTechText, String payText, String datePayText, String paySumText, String docIssuedText, String primechanieText)  
{

... // запрос и PreparedStatement }

Затем создаём сам SQL-запрос, наследуя класс-константу, и записываем его в переменную.

DatabaseHandler.java

String insert = "INSERT INTO " + requests\_const.REQUESTS\_TABLE + "(`" + requests\_const.NUM\_REQ + "`,`" + requests\_const.FIO\_ZAK +  
"`,`" + requests\_const.PHONE + "`,`" + requests\_const.FIZ\_UR + "`,`" + requests\_const.BANK + "`,`" + requests\_const.NUM\_BANK + "`,`" + requests\_const.ADDRESS + "`,`" + requests\_const.DISTRICT + "`,`" + requests\_const.DATE\_REQUEST + "`,`" + requests\_const.DOC\_NAME + "`,`" + requests\_const.SROCHNOE + "`,`" + requests\_const.DATE\_VIDACHI + "`,`" + requests\_const.PRICE\_DOC + "`,`" + requests\_const.DATE\_TECH + "`,`" + requests\_const.TIME\_TECH + "`,`" + requests\_const.FIO\_NACH + "`,`" + requests\_const.FIO\_ISP + "`,`" + requests\_const.DATE\_PEREDACHI + "`,`" + requests\_const.FIO\_TECH + "`,`" + requests\_const.PAY + "`,`" + requests\_const.DATE\_PAY + "`,`" + requests\_const.PAY\_SUM + "`,`" + requests\_const.DOC\_ISSUED + "`,`" + requests\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";

Создаём PreparedStatement, который предварительно компилирует запросы, содержащие входные параметры, обозначенные символом '?'.

DatabaseHandler.java

PreparedStatement prst = getDbConnection().prepareStatement(insert);  
prst.setString(1, numReq);  
prst.setString(2, fiOzakText);  
...  
prst.setString(23, docIssuedText);  
prst.setString(24, primechanieText);  
prst.executeUpdate();

Аналогичные обработчики создаём для каждой таблицы.

Раздел 3 ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ ПРОГРАММЫ

3.1 Назначение программы

Прикладной программный продукт «Бюро технической инвентаризации» предназначен для приёма заявок на техническую инвентаризацию.

Принцип работы заключается в записи заявки в таблицы базы данных.

3.2 Выполнение программы

1. Запуск приложения.

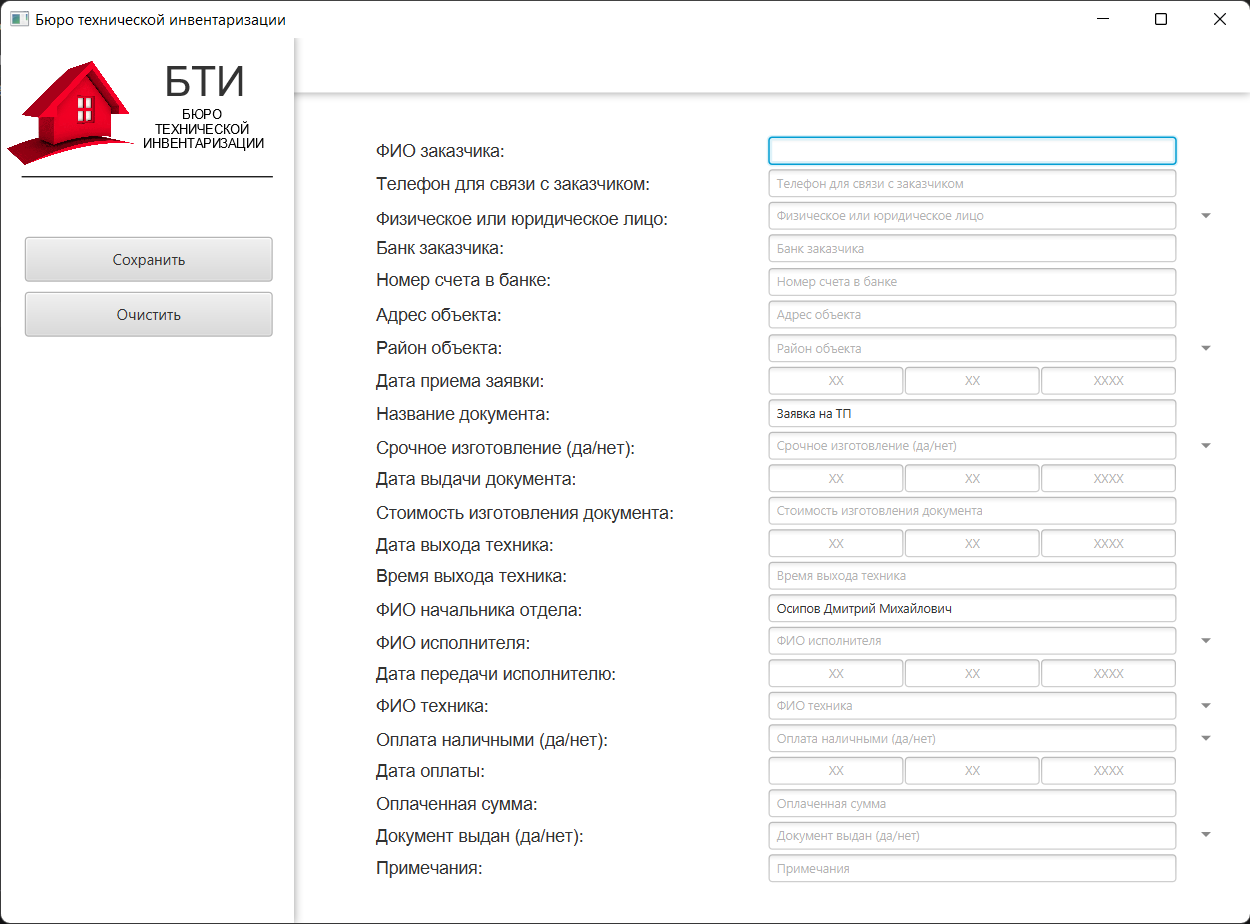


Рисунок 14 – Окно приложения.

2. Ввод данных заявки. Выбор района объекта.

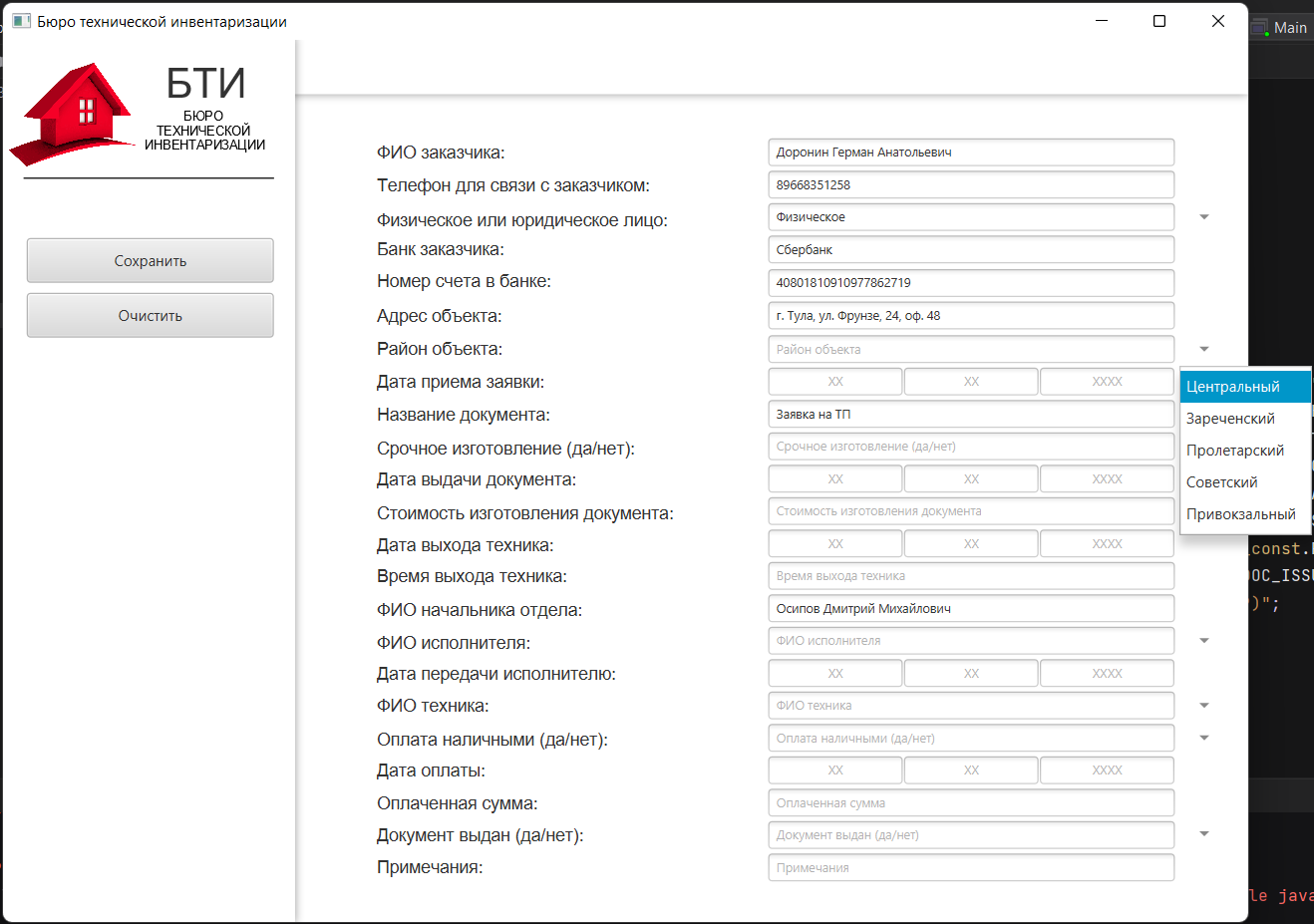


Рисунок 15 – Окно приложения. Выбор район объекта.

3. Ввод данных заявки. Выбор исполнителя.

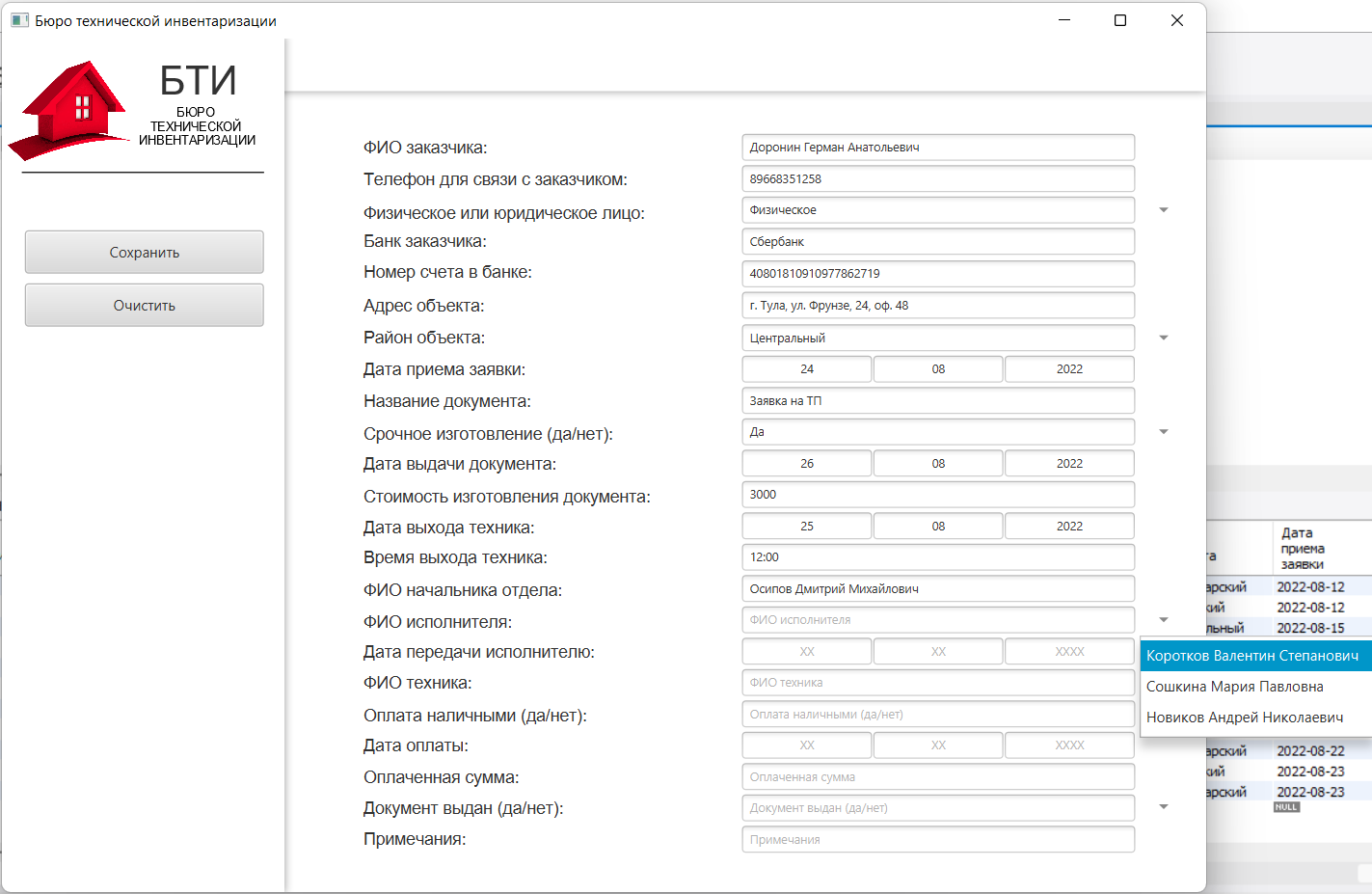


Рисунок 16 – Окно приложения. Выбор исполнителя.

1. Ввод данных заказа. Выбор техника.

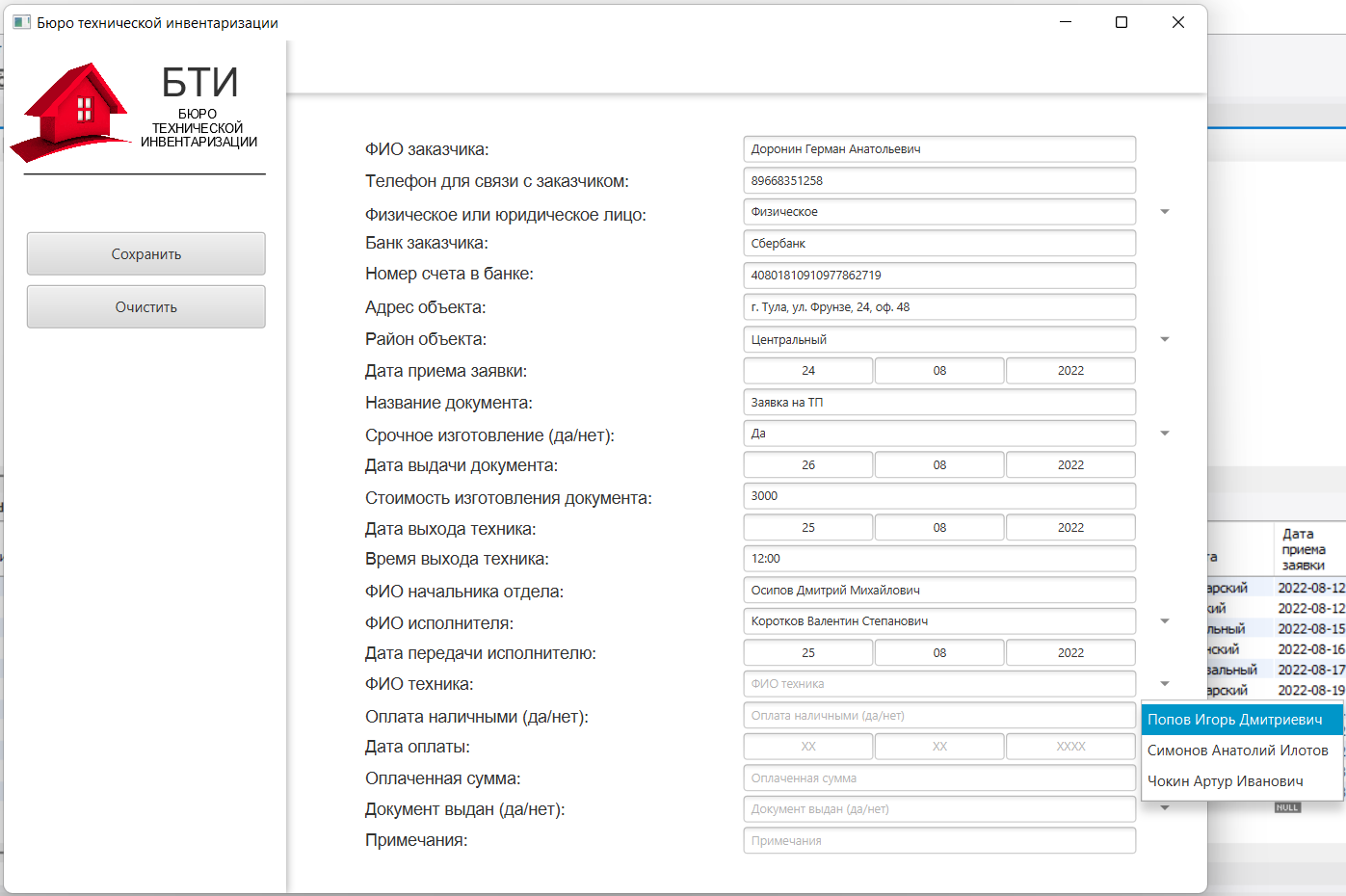


Рисунок 17 – Окно приложения. Выбор техника.

1. Сохранение заявки в базе данных.

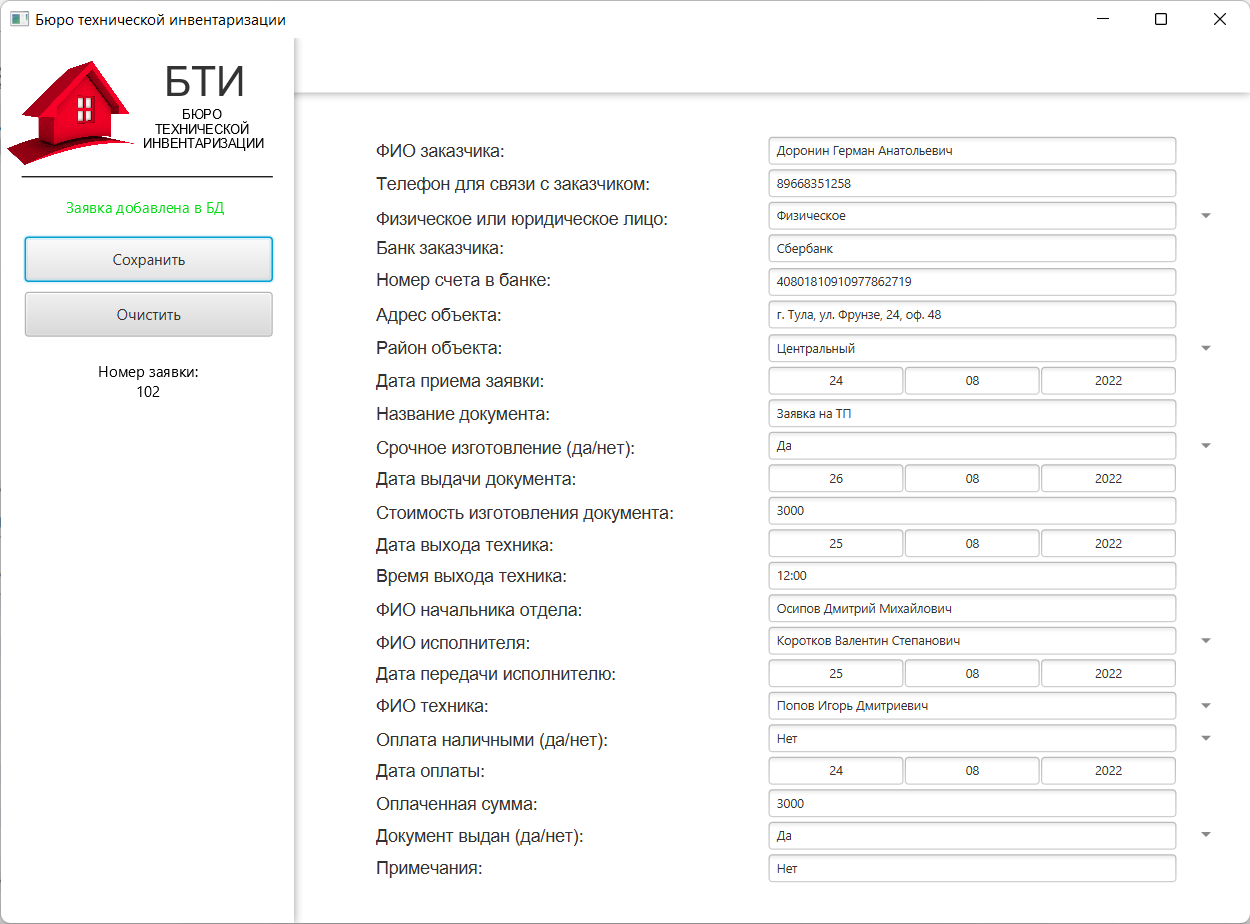
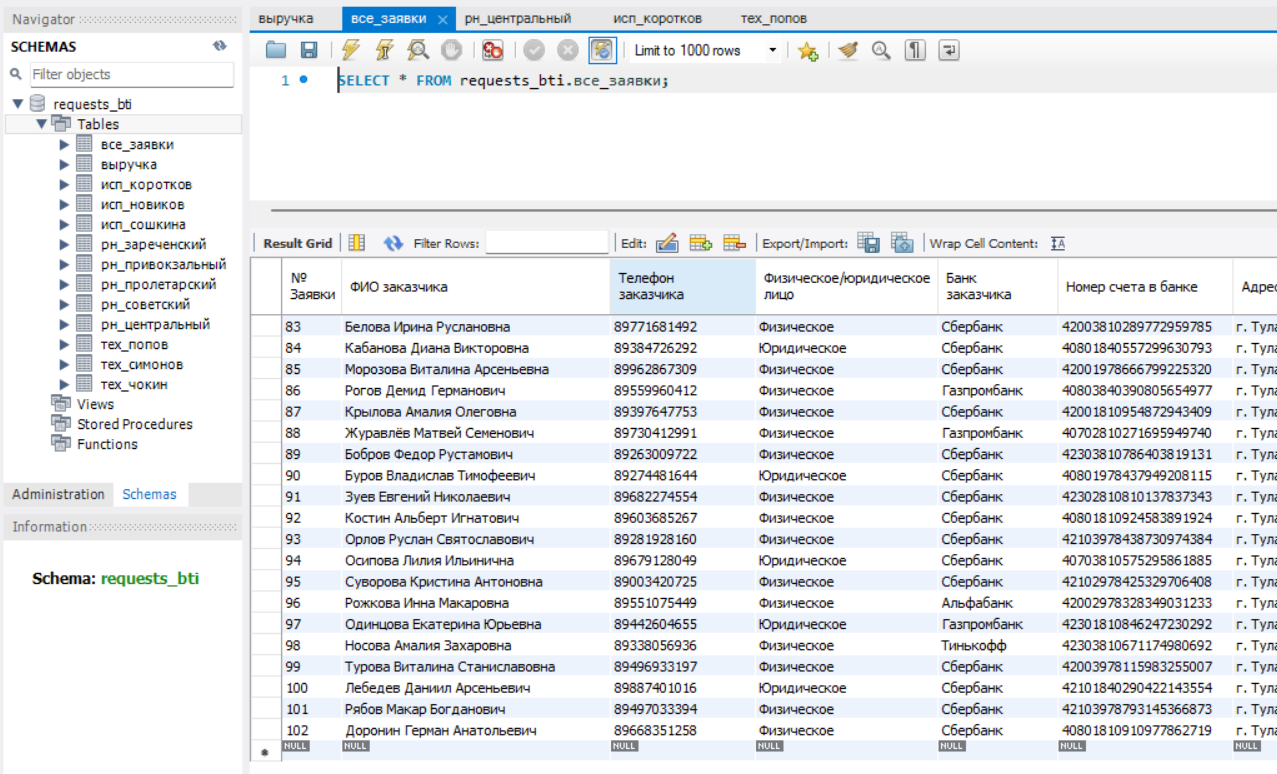
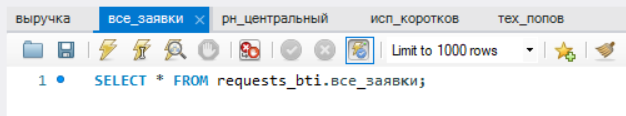


Рисунок 18 – Окно приложения. Сохранение заявки.

1. Проверка записи заявки в таблицу «все\_заявки» базы данных.





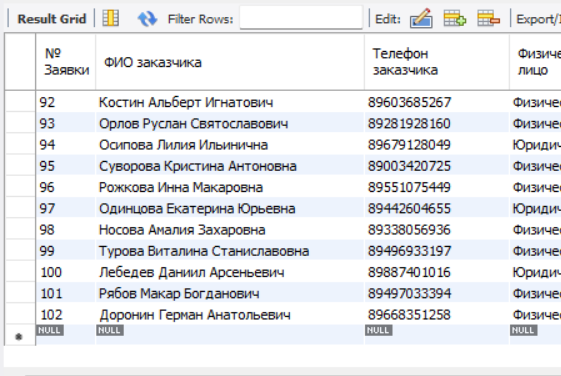
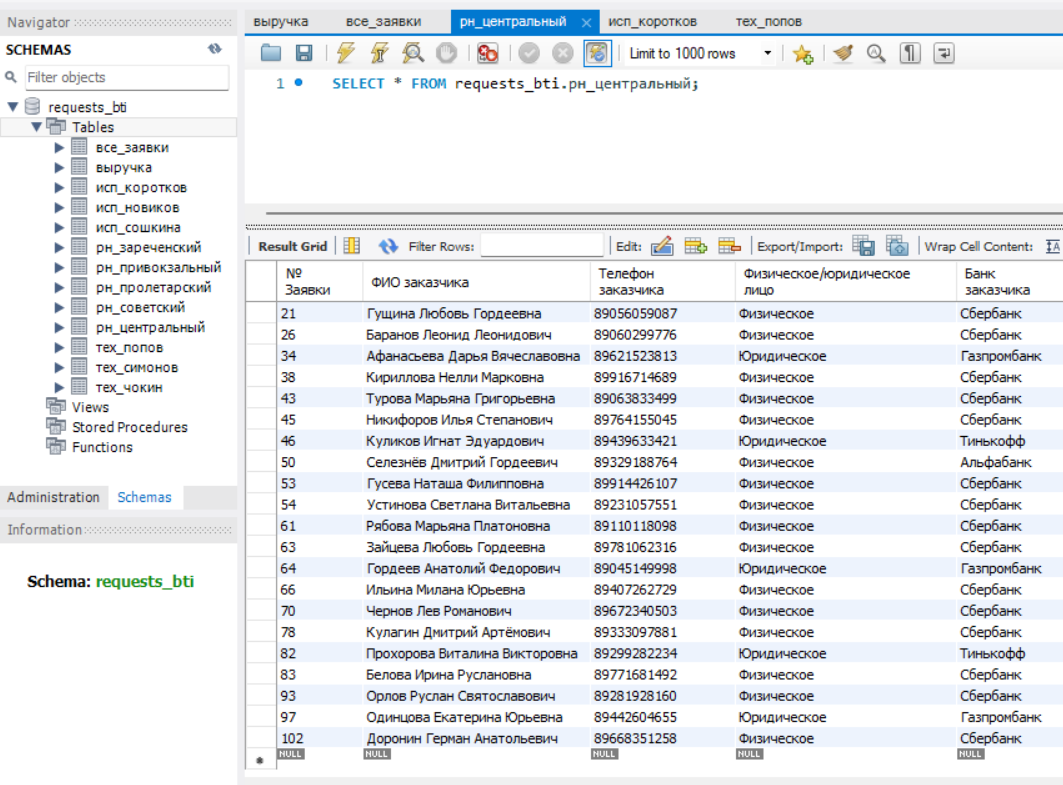
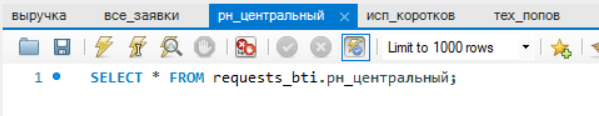


Рисунок 19 – База данных. Таблица «все\_заявки».

1. Проверка записи заявки в таблицу «рн\_центральный» базы данных.





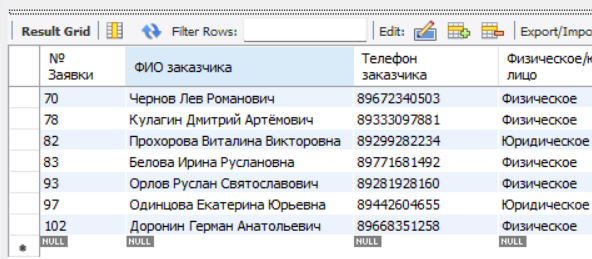
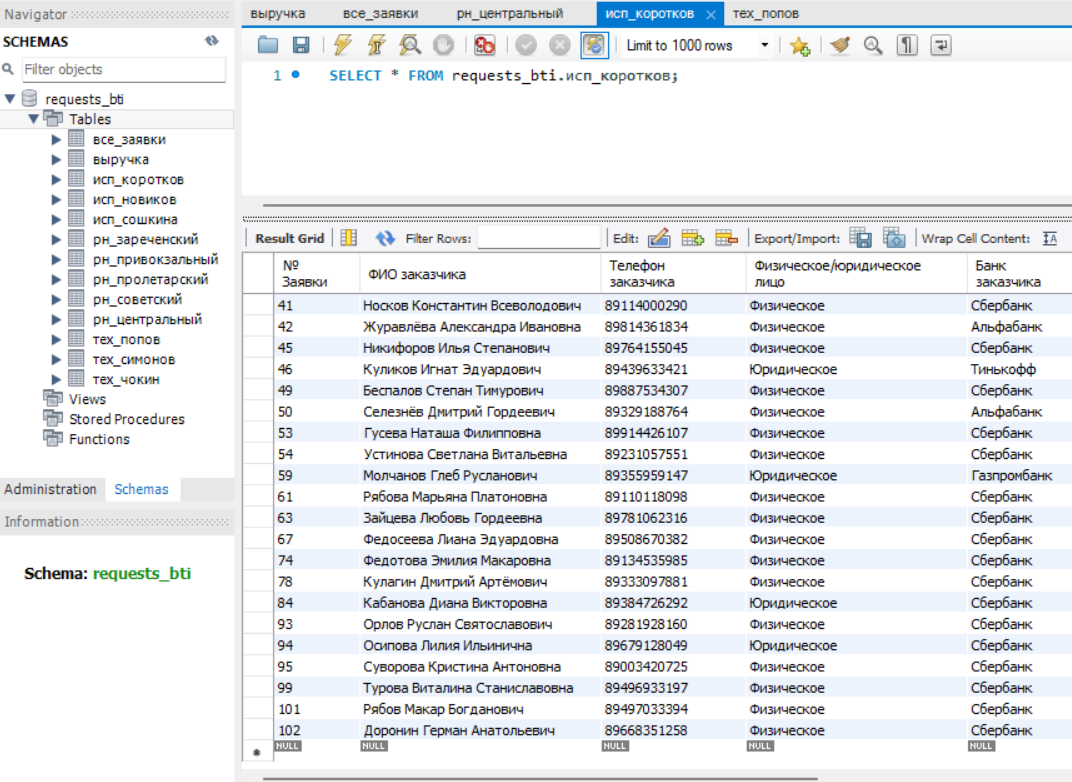
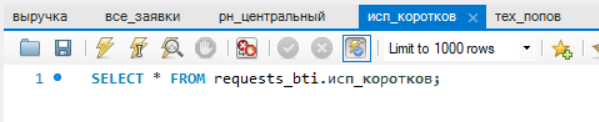


Рисунок 20 – База данных. Таблица «рн\_центральный».

1. Проверка записи заявки в таблицу «исп\_коротков» базы данных.





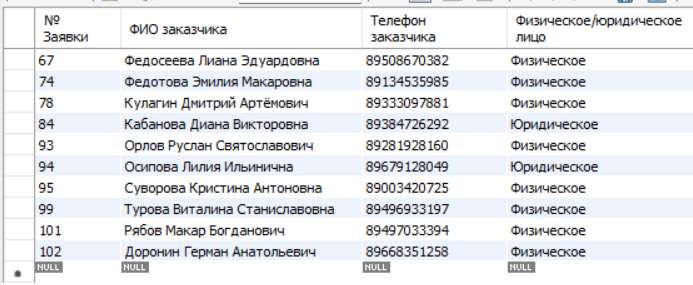
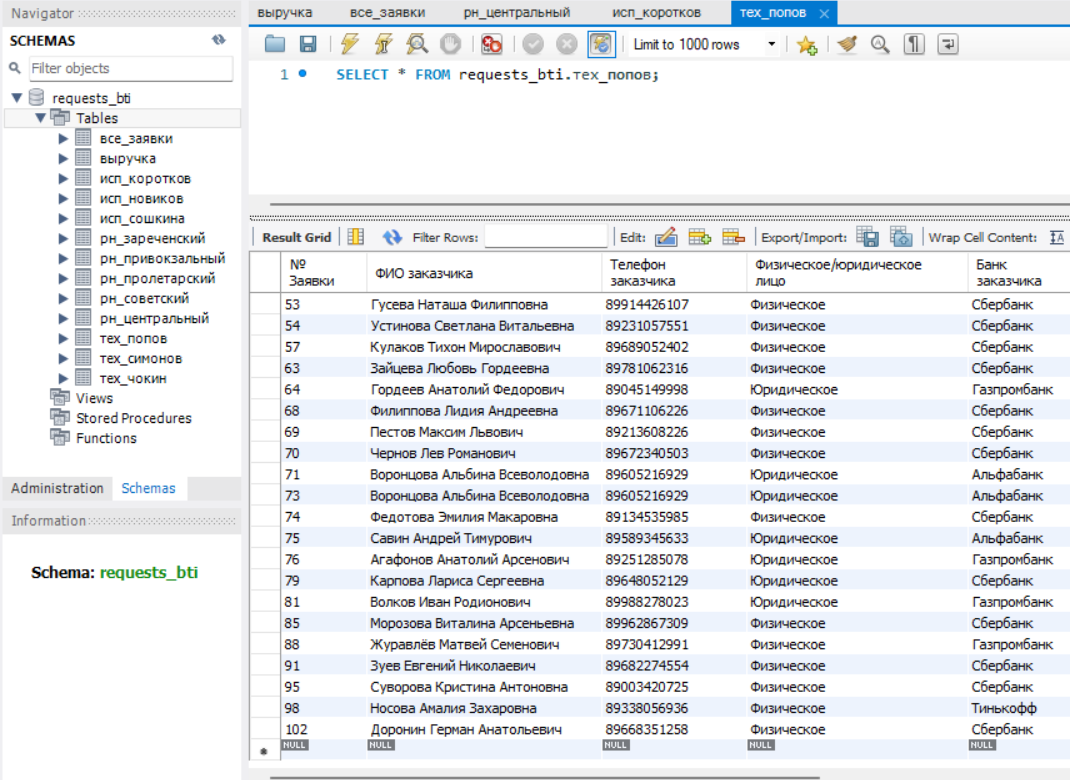
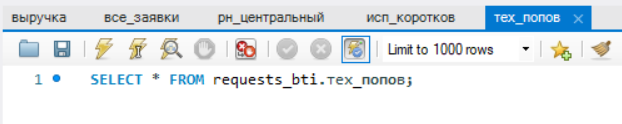


Рисунок 21 – База данных. Таблица «исп\_коротков».

1. Проверка записи заявки в таблицу «тех\_попов» базы данных.





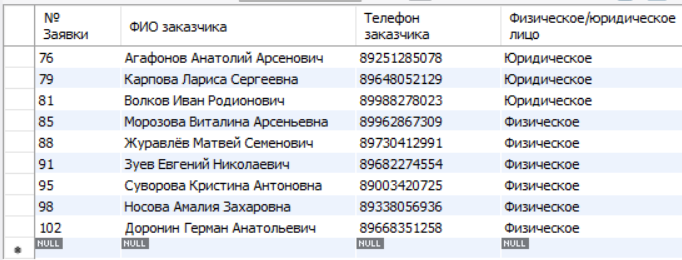


Рисунок 22 – База данных. Таблица «тех\_попов».

1. Проверка суммы выручки.

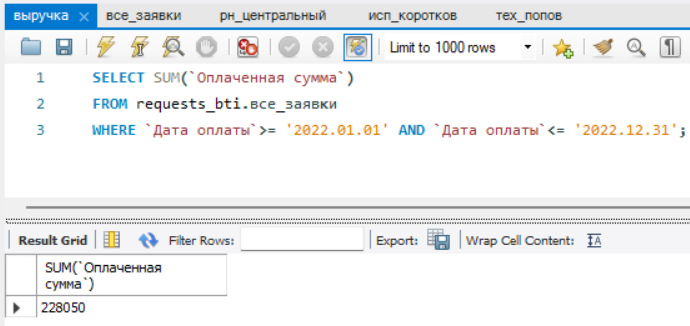


Рисунок 23 – Выручка за год.

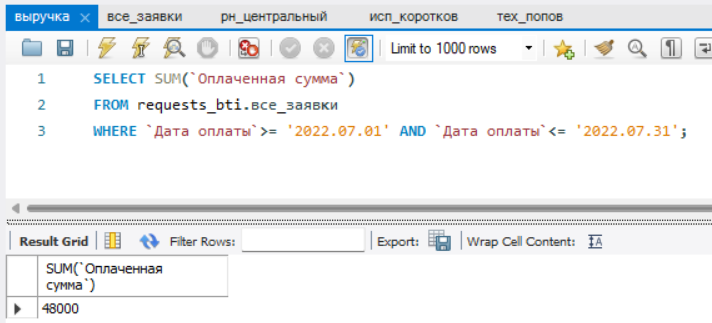


Рисунок 24 – Выручка за месяц.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы было разработано прикладное программное обеспечение на тему «Бюро технической инвентаризации», закреплены навыки работы с графикой на языке программирования JavaFX, разработано оконное Windows-приложение, реализована запись заявок из полей ввода Windows-приложения в таблицы базы данных.

Список использованных источников

1. <https://ru.wikipedia.org/>
2. <https://habr.com/> – русскоязычный [веб-сайт](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82) в формате системы тематических [коллективных блогов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%B1%D0%BB%D0%BE%D0%B3)
3. [Генератор случайных ФИО (generand.ru)](https://generand.ru/html/fio.html)
4. [Генератор случайных значений расчётных счётов (generand.ru)](https://generand.ru/html/ras.html?ysclid=l3ls1qk9s9)
5. [Генератор случайных значений адресов юридических и физических лиц (generand.ru)](https://generand.ru/html/address.html)
6. [Генератор телефонных номеров (freegenerator.ru)](https://freegenerator.ru/phone_number?)
7. <https://go.gliffy.com/> – Программное обеспечение для построения диаграмм с помощью облачного приложения HTML5.

ПРИЛОЖЕНИЕ

Листинг программ

**Main.java**

package com.example.bti;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
import java.io.IOException;  
  
public class Main extends Application {  
  
 @Override  
 public void start(Stage stage) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("BTI.fxml"));  
 Scene scene = new Scene(fxmlLoader.load(), 999, 708);  
 stage.setTitle("Бюро технической инвентаризации");  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 public static void main(String[] args) {  
 launch();  
 }  
}

**Controller.java**

package com.example.bti;  
  
import java.io.\*;  
import java.net.URL;  
import java.util.Arrays;  
import java.util.ResourceBundle;  
import java.util.function.UnaryOperator;  
import java.util.regex.Pattern;  
  
import com.sun.javafx.stage.EmbeddedWindow;  
import javafx.application.Platform;  
import javafx.beans.value.ChangeListener;  
import javafx.beans.value.ObservableValue;  
import javafx.css.converter.StringConverter;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Node;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.input.KeyCode;  
import javafx.scene.input.KeyCodeCombination;  
import javafx.scene.input.KeyCombination;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.scene.layout.AnchorPane;  
import javafx.scene.text.Text;  
import javafx.stage.Stage;  
  
public class Controller {  
  
 @FXML  
 private ResourceBundle resources;  
 @FXML  
 private URL location;  
 @FXML  
 private Text BTIname;  
 @FXML  
 private Text BTIname2;  
 @FXML  
 private Button PrintButton;  
 @FXML  
 private Button SaveButton;  
 @FXML  
 private Button CleanButton;  
 @FXML  
 private TextField FIOzakInput;  
 @FXML  
 private TextField PhoneInput;  
 @FXML  
 private TextField FizUrInput;  
 @FXML  
 private TextField NumBankInput;  
 @FXML  
 private TextField BankInput;  
 @FXML  
 private TextField AddressInput;  
 @FXML  
 private TextField DistrictInput;  
 @FXML  
 private TextField DocNameInput;  
 @FXML  
 private TextField DateRequestInput;  
 @FXML  
 private TextField SrochnoeInput;  
 @FXML  
 private TextField DateVidachiInput;  
 @FXML  
 private TextField DateTechInput;  
 @FXML  
 private TextField PriceDocInput;  
 @FXML  
 private TextField TimeTechInput;  
 @FXML  
 private TextField FIOnachInput;  
 @FXML  
 private TextField DatePeredachiInput;  
 @FXML  
 private TextField FIOispInput;  
 @FXML  
 private TextField FIOTechInput;  
 @FXML  
 private TextField PayInput;  
 @FXML  
 private TextField PaySumInput;  
 @FXML  
 private TextField DatePayInput;  
 @FXML  
 private TextField DocIssuedInput;  
 @FXML  
 private TextField PrimechanieInput;  
 @FXML  
 private Text ErrorText;  
 @FXML  
 private Text DoneText;  
 @FXML  
 private MenuButton DistrictMenu;  
 @FXML  
 private MenuButton FizUrMenu;  
 @FXML  
 private MenuButton SrochnoeMenu;  
 @FXML  
 private MenuButton PayMenu;  
 @FXML  
 private MenuButton DocIssuedMenu;  
 @FXML  
 private MenuButton FIOispMenu;  
 @FXML  
 private MenuButton FIOTechMenu;  
 @FXML  
 private Text numReqText;  
 @FXML  
 private TextField DateRequestInput1;  
 @FXML  
 private TextField DateRequestInput2;  
 @FXML  
 private TextField DateRequestInput3;  
 @FXML  
 private TextField DateVidachiInput1;  
 @FXML  
 private TextField DateVidachiInput2;  
 @FXML  
 private TextField DateVidachiInput3;  
 @FXML  
 private TextField DateTechInput1;  
 @FXML  
 private TextField DateTechInput2;  
 @FXML  
 private TextField DateTechInput3;  
 @FXML  
 private TextField DatePeredachiInput1;  
 @FXML  
 private TextField DatePeredachiInput2;  
 @FXML  
 private TextField DatePeredachiInput3;  
 @FXML  
 private TextField DatePayInput1;  
 @FXML  
 private TextField DatePayInput2;  
 @FXML  
 private TextField DatePayInput3;  
  
  
 @FXML  
 void initialize() {  
  
 DocNameInput.setText("Заявка на ТП");  
 FIOnachInput.setText("Осипов Дмитрий Михайлович");  
  
 ErrorText.setVisible(false);  
 DoneText.setVisible(false);  
 Platform.runLater(FIOzakInput::requestFocus);  
  
 DatabaseHandler dbHandler = new DatabaseHandler();  
  
 FIOzakInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 PhoneInput.requestFocus();  
 }  
 });PhoneInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 FizUrMenu.requestFocus();  
 FizUrMenu.show();  
 }  
 });BankInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 NumBankInput.requestFocus();  
 }  
 });NumBankInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 AddressInput.requestFocus();  
 }  
 });AddressInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DistrictMenu.requestFocus();  
 DistrictMenu.show();  
 }  
 });  
 DateRequestInput1.setOnKeyPressed(event ->{  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DateRequestInput2.requestFocus();  
 }  
 });  
 DateRequestInput2.setOnKeyPressed(event ->{  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DateRequestInput3.requestFocus();  
 }  
 });  
 DateRequestInput3.setOnKeyPressed(event ->{  
 if(event.getCode().equals(KeyCode.ENTER)){  
 SrochnoeMenu.requestFocus();  
 SrochnoeMenu.show();  
 }  
 });  
 DateVidachiInput1.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DateVidachiInput2.requestFocus();  
 }  
 });  
 DateVidachiInput2.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DateVidachiInput3.requestFocus();  
 }  
 });  
 DateVidachiInput3.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 PriceDocInput.requestFocus();  
 }  
 });  
 PriceDocInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DateTechInput1.requestFocus();  
 }  
 });  
 DateTechInput1.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DateTechInput2.requestFocus();  
 }  
 });  
 DateTechInput2.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DateTechInput3.requestFocus();  
 }  
 });  
 DateTechInput3.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 TimeTechInput.requestFocus();  
 }  
 });  
 TimeTechInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 FIOispMenu.requestFocus();  
 FIOispMenu.show();  
 }  
 });  
 DatePeredachiInput1.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DatePeredachiInput2.requestFocus();  
 }  
 });DatePeredachiInput2.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DatePeredachiInput3.requestFocus();  
 }  
 });DatePeredachiInput3.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 FIOTechMenu.requestFocus();  
 FIOTechMenu.show();  
 }  
 });  
 DatePayInput1.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DatePayInput2.requestFocus();  
 }  
 });DatePayInput2.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DatePayInput3.requestFocus();  
 }  
 });DatePayInput3.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 PaySumInput.requestFocus();  
 }  
 });PaySumInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 DocIssuedMenu.requestFocus();  
 DocIssuedMenu.show();  
 }  
 });PrimechanieInput.setOnKeyPressed(event -> {  
 if(event.getCode().equals(KeyCode.ENTER)){  
 FIOzakInput.requestFocus();  
 }  
 });  
  
 // ОГРАНИЧЕНИЯ ВВОДА  
 FIOzakInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я")) {  
 FIOzakInput.setText(newValue.replaceAll("[^\\sа-яА-Я ё]", ""));  
 }  
 });  
 PhoneInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 PhoneInput.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 FizUrInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я\*")) {  
 FizUrInput.setText(newValue.replaceAll("[^\\sа-яА-Я]", ""));  
 }  
 });  
 BankInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Яa-zA-Z\*")) {  
 BankInput.setText(newValue.replaceAll("[^\\sа-яА-Яa-zA-Z ё]", ""));  
 }  
 });  
 NumBankInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*")) {  
 NumBankInput.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 DistrictInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я\*")) {  
 DistrictInput.setText(newValue.replaceAll("[^\\sа-яА-Я]", ""));  
 }  
 });  
 DateRequestInput1.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DateRequestInput1.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 DateRequestInput2.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DateRequestInput2.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 DateRequestInput3.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DateRequestInput3.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
  
 DocNameInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я\*")) {  
 DocNameInput.setText(newValue.replaceAll("[^\\sа-яА-Я ё]", ""));  
 }  
 });  
 SrochnoeInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я\*")) {  
 SrochnoeInput.setText(newValue.replaceAll("[^\\sа-яА-Я]", ""));  
 }  
 });  
 DateVidachiInput1.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DateVidachiInput1.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 DateVidachiInput2.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DateVidachiInput2.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 DateVidachiInput3.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DateVidachiInput3.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
  
 PriceDocInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*")) {  
 PriceDocInput.setText(newValue.replaceAll("[^\\d]", ","));  
 }  
 });  
 DateTechInput1.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DateTechInput1.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 DateTechInput2.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DateTechInput2.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 DateTechInput3.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DateTechInput3.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 TimeTechInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*")) {  
 TimeTechInput.setText(newValue.replaceAll("[^\\d]", ":"));  
 }  
 });  
 FIOnachInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-ЯZ\*")) {  
 FIOnachInput.setText(newValue.replaceAll("[^\\sа-яА-Я ё]", ""));  
 }  
 });  
 FIOispInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я\*")) {  
 FIOispInput.setText(newValue.replaceAll("[^\\sа-яА-Я ё]", ""));  
 }  
 });  
 DatePeredachiInput1.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DatePeredachiInput1.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });DatePeredachiInput2.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DatePeredachiInput2.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });DatePeredachiInput3.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DatePeredachiInput3.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 FIOTechInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я\*")) {  
 FIOTechInput.setText(newValue.replaceAll("[^\\sа-яА-Я ё]", ""));  
 }  
 });  
 PayInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я\*")) {  
 PayInput.setText(newValue.replaceAll("[^\\sа-яА-Я]", ""));  
 }  
 });  
 DatePayInput1.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DatePayInput1.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });DatePayInput2.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DatePayInput2.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });DatePayInput3.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*") ) {  
 DatePayInput3.setText(newValue.replaceAll("[^\\d]", ""));  
 }  
 });  
 PaySumInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\d\*")) {  
 PaySumInput.setText(newValue.replaceAll("[^\\d]", ","));  
 }  
 });  
 DocIssuedInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я\*")) {  
 DocIssuedInput.setText(newValue.replaceAll("[^\\sа-яА-Я]", ""));  
 }  
 });  
 PrimechanieInput.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!newValue.matches("\\sа-яА-Я\*")) {  
 PrimechanieInput.setText(newValue.replaceAll("[^\\sа-яА-Я ё]", ""));  
 }  
 });  
  
 // Физическое/Юридическое лицо (МЕНЮ)  
 MenuItem item1 = new MenuItem("Физическое");  
 MenuItem item2 = new MenuItem("Юридическое");  
  
 FizUrMenu.getItems().addAll(item1, item2);  
  
 item1.setOnAction(event -> {  
 FizUrInput.setText("Физическое");  
 BankInput.requestFocus();  
 });  
 item2.setOnAction(event -> {  
 FizUrInput.setText("Юридическое");  
 BankInput.requestFocus();  
 });  
  
 // Район объекта (МЕНЮ)  
 MenuItem item3 = new MenuItem("Центральный");  
 MenuItem item4 = new MenuItem("Зареченский");  
 MenuItem item5 = new MenuItem("Пролетарский");  
 MenuItem item6 = new MenuItem("Советский");  
 MenuItem item7 = new MenuItem("Привокзальный");  
  
 DistrictMenu.getItems().addAll(item3, item4, item5, item6, item7);  
  
 item3.setOnAction(event -> {  
 DistrictInput.setText("Центральный");  
 DateRequestInput1.requestFocus();  
 });  
 item4.setOnAction(event -> {  
 DistrictInput.setText("Зареченский");  
 DateRequestInput1.requestFocus();  
 });  
 item5.setOnAction(event -> {  
 DistrictInput.setText("Пролетарский");  
 DateRequestInput1.requestFocus();  
 });  
 item6.setOnAction(event -> {  
 DistrictInput.setText("Советский");  
 DateRequestInput1.requestFocus();  
 });  
 item7.setOnAction(event -> {  
 DistrictInput.setText("Привокзальный");  
 DateRequestInput1.requestFocus();  
 });  
  
 // Срочное изготовление (МЕНЮ)  
 MenuItem item8 = new MenuItem("Да");  
 MenuItem item9 = new MenuItem("Нет");  
  
 SrochnoeMenu.getItems().addAll(item8, item9);  
  
 item8.setOnAction(event -> {  
 SrochnoeInput.setText("Да");  
 DateVidachiInput1.requestFocus();  
 });  
 item9.setOnAction(event -> {  
 SrochnoeInput.setText("Нет");  
 DateVidachiInput1.requestFocus();  
 });  
  
 // Оплата наличными (МЕНЮ)  
 MenuItem item10 = new MenuItem("Да");  
 MenuItem item11 = new MenuItem("Нет");  
  
 PayMenu.getItems().addAll(item10, item11);  
  
 item10.setOnAction(event -> {  
 PayInput.setText("Да");  
 DatePayInput1.requestFocus();  
 });  
 item11.setOnAction(event -> {  
 PayInput.setText("Нет");  
 DatePayInput1.requestFocus();  
 });  
  
 // Документ выдан (МЕНЮ)  
 MenuItem item12 = new MenuItem("Да");  
 MenuItem item13 = new MenuItem("Нет");  
  
 DocIssuedMenu.getItems().addAll(item12, item13);  
  
 item12.setOnAction(event -> {  
 DocIssuedInput.setText("Да");  
 PrimechanieInput.requestFocus();  
 });  
 item13.setOnAction(event -> {  
 DocIssuedInput.setText("Нет");  
 PrimechanieInput.requestFocus();  
 });  
  
 // ФИО исполнителя (МЕНЮ)  
 MenuItem item14 = new MenuItem("Коротков Валентин Степанович");  
 MenuItem item15 = new MenuItem("Сошкина Мария Павловна");  
 MenuItem item16 = new MenuItem("Новиков Андрей Николаевич");  
  
 FIOispMenu.getItems().addAll(item14, item15, item16);  
  
 item14.setOnAction(event -> {  
 FIOispInput.setText("Коротков Валентин Степанович");  
 DatePeredachiInput1.requestFocus();  
 });  
 item15.setOnAction(event -> {  
 FIOispInput.setText("Сошкина Мария Павловна");  
 DatePeredachiInput1.requestFocus();  
 });  
 item16.setOnAction(event -> {  
 FIOispInput.setText("Новиков Андрей Николаевич");  
 DatePeredachiInput1.requestFocus();  
 });  
  
 // ФИО техника (МЕНЮ)  
 MenuItem item17 = new MenuItem("Попов Игорь Дмитриевич");  
 MenuItem item18 = new MenuItem("Симонов Анатолий Илотов");  
 MenuItem item19 = new MenuItem("Чокин Артур Иванович");  
  
 FIOTechMenu.getItems().addAll(item17, item18, item19);  
  
 item17.setOnAction(event -> {  
 FIOTechInput.setText("Попов Игорь Дмитриевич");  
 PayMenu.requestFocus();  
 });  
 item18.setOnAction(event -> {  
 FIOTechInput.setText("Симонов Анатолий Илотов");  
 PayMenu.requestFocus();  
 });  
 item19.setOnAction(event -> {  
 FIOTechInput.setText("Чокин Артур Иванович");  
 PayMenu.requestFocus();  
 });  
  
  
 SaveButton.setOnAction(event -> {  
  
 String datereq1 = DateRequestInput1.getText();  
 String datereq2 = DateRequestInput2.getText();  
 String datereq3 = DateRequestInput3.getText();  
  
 String datevid1 = DateVidachiInput1.getText();  
 String datevid2 = DateVidachiInput2.getText();  
 String datevid3 = DateVidachiInput3.getText();  
  
 String datetec1 = DateTechInput1.getText();  
 String datetec2 = DateTechInput2.getText();  
 String datetec3 = DateTechInput3.getText();  
  
 String dateper1 = DatePeredachiInput1.getText();  
 String dateper2 = DatePeredachiInput2.getText();  
 String dateper3 = DatePeredachiInput3.getText();  
  
 String datepay1 = DatePayInput1.getText();  
 String datepay2 = DatePayInput2.getText();  
 String datepay3 = DatePayInput3.getText();  
  
  
 String FIOzakText = FIOzakInput.getText().trim(); //trim удаляет лишние пробелы  
 String PhoneText = PhoneInput.getText().trim();  
 String FizUrText = FizUrInput.getText().trim();  
 String BankText = BankInput.getText().trim();  
 String NumBankText = NumBankInput.getText().trim();  
 String AdressText = AddressInput.getText().trim();  
 String DistrictText = DistrictInput.getText().trim();  
 String DateRequestText = datereq3 + "." + datereq2 + "." + datereq1;  
 String DocNameText = DocNameInput.getText().trim();  
 String SrochnoeText = SrochnoeInput.getText().trim();  
 String DateVidachiText = datevid3 + "." + datevid2 + "." + datevid1;  
 String PriceDocText = PriceDocInput.getText().trim();  
 String DateTechText = datetec3 + "." + datetec2 + "." + datetec1;  
 String TimeTechText = TimeTechInput.getText().trim();  
 String FIOnachText = FIOnachInput.getText().trim();  
 String FIOispText = FIOispInput.getText().trim();  
 String DatePeredachiText = dateper3 + "." + dateper2 + "." + dateper1;  
 String FIOTechText = FIOTechInput.getText().trim();  
 String PayText = PayInput.getText().trim();  
 String DatePayText = datepay3 + "." + datepay2 + "." + datepay1;  
 String PaySumText = PaySumInput.getText().trim();  
 String DocIssuedText = DocIssuedInput.getText().trim();  
 String PrimechanieText = PrimechanieInput.getText().trim();  
  
  
 if( !FIOzakText.equals("") && !PhoneText.equals("") && !FizUrText.equals("") &&  
 !BankText.equals("") && !NumBankText.equals("") && !AdressText.equals("") && !DistrictText.equals("") &&  
  
 !DateRequestInput1.equals("") && !DateRequestInput2.equals("") && !DateRequestInput3.equals("") &&  
 !DateRequestText.equals("") &&  
  
 !DocNameText.equals("") && !SrochnoeText.equals("") &&  
  
 !DateVidachiInput1.equals("") && !DateVidachiInput2.equals("") && !DateVidachiInput3.equals("") &&  
 !DateVidachiText.equals("") &&  
  
 !PriceDocText.equals("") &&  
  
 !DateTechInput1.equals("") && !DateTechInput2.equals("") && !DateTechInput3.equals("") &&  
 !DateTechText.equals("") &&  
  
 !TimeTechText.equals("") && !FIOnachText.equals("") &&  
 !FIOispText.equals("") &&  
  
 !DatePeredachiInput1.equals("") && !DatePeredachiInput2.equals("") && !DatePeredachiInput3.equals("") &&  
 !DatePeredachiText.equals("") &&  
  
 !FIOTechText.equals("") && !PayText.equals("") &&  
  
 !DatePayInput1.equals("") && !DatePayInput2.equals("") && !DatePayInput3.equals("") &&  
 !DatePayText.equals("") &&  
  
 !PaySumText.equals("") && !DocIssuedText.equals("") && !PrimechanieText.equals(""))  
 {  
  
 File file = new File("src/main/java/com/example/bti/num\_req.txt");  
  
 FileReader fr = null;  
 try {  
 fr = new FileReader(file);  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 BufferedReader reader = new BufferedReader(fr);  
 String stringLine = null;  
 try {  
 stringLine = reader.readLine();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 int intLine = Integer.parseInt(stringLine);  
 intLine++;  
 String numReq = Integer.toString(intLine);  
 try {  
 reader.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 FileWriter fw = null;  
 try {  
 fw = new FileWriter(file);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 BufferedWriter writer = new BufferedWriter(fw);  
 try {  
 writer.write(numReq);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 try {  
 writer.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 numReqText.setText("Номер заявки:\n" + numReq);  
 numReqText.setVisible(true);  
  
 //====================================================  
  
 dbHandler.addRequest(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
  
 //====================================================  
  
 if (DistrictText.equals("Центральный")){  
 dbHandler.addCentralniy(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
 if (DistrictText.equals("Зареченский")){  
 dbHandler.addZarechenskiy(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
 if (DistrictText.equals("Пролетарский")){  
 dbHandler.addProletarskiy(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
 if (DistrictText.equals("Советский")){  
 dbHandler.addSovetskiy(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
 if (DistrictText.equals("Привокзальный")){  
 dbHandler.addPrivokzalniy(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
  
 //====================================================  
  
 if (FIOispText.equals("Коротков Валентин Степанович")){  
 dbHandler.addIspKorotkov(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
 if (FIOispText.equals("Сошкина Мария Павловна")){  
 dbHandler.addIspSoshkina(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
 if (FIOispText.equals("Новиков Андрей Николаевич")){  
 dbHandler.addIspNovikov(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
  
 //====================================================  
  
 if (FIOTechText.equals("Попов Игорь Дмитриевич")){  
 dbHandler.addTechPopov(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
 if (FIOTechText.equals("Симонов Анатолий Илотов")){  
 dbHandler.addTechSimonov(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
 if (FIOTechText.equals("Чокин Артур Иванович")){  
 dbHandler.addTechChokin(numReq, FIOzakText, PhoneText, FizUrText,  
 BankText, NumBankText, AdressText, DistrictText, DateRequestText,  
 DocNameText, SrochnoeText, DateVidachiText, PriceDocText, DateTechText,  
 TimeTechText, FIOnachText, FIOispText, DatePeredachiText, FIOTechText,  
 PayText, DatePayText, PaySumText, DocIssuedText, PrimechanieText);  
 }  
  
 ErrorText.setVisible(false);  
 DoneText.setVisible(true);  
 System.out.println("Пользователь добавлен в базу данных");  
  
  
 } else {  
 ErrorText.setVisible(true);  
 DoneText.setVisible(false);  
 System.out.println("Не все поля заполнены");  
 }  
  
 });  
  
 CleanButton.setOnAction(event -> {  
 FIOzakInput.setText(""); PhoneInput.setText(""); FizUrInput.setText("");  
 NumBankInput.setText(""); BankInput.setText(""); AddressInput.setText("");  
 DistrictInput.setText("");  
 DateRequestInput1.setText(""); DateRequestInput2.setText(""); DateRequestInput3.setText("");  
 SrochnoeInput.setText("");  
 DateVidachiInput1.setText(""); DateVidachiInput2.setText(""); DateVidachiInput3.setText("");  
 DateTechInput1.setText(""); DateTechInput2.setText(""); DateTechInput3.setText("");  
 PriceDocInput.setText(""); TimeTechInput.setText(""); FIOispInput.setText("");  
 DatePeredachiInput1.setText(""); DatePeredachiInput2.setText(""); DatePeredachiInput3.setText("");  
 FIOTechInput.setText(""); PayInput.setText("");  
 PaySumInput.setText("");  
 DatePayInput1.setText(""); DatePayInput2.setText(""); DatePayInput3.setText("");  
 DocIssuedInput.setText("");  
 PrimechanieInput.setText("");  
  
 ErrorText.setVisible(false);  
 DoneText.setVisible(false);  
  
 numReqText.setVisible(false);  
  
 FIOzakInput.requestFocus();  
 });  
  
 }  
  
}

**DatabaseHandler.java**

package com.example.bti;  
  
import java.sql.\*;  
  
public class DatabaseHandler extends Configs {  
 Connection dbConnection;  
  
 public Connection getDbConnection() throws ClassNotFoundException, SQLException{  
 String connectionString = "jdbc:mysql://" + dbHost + ":" + dbPort + "/" + dbName + "?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";  
  
 Class.forName("com.mysql.cj.jdbc.Driver");  
  
 dbConnection = DriverManager.getConnection(connectionString, dbUser, dbPass);  
  
 return dbConnection;  
 }  
  
 void addRequest(String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + requests\_const.REQUESTS\_TABLE + "(`" + requests\_const.NUM\_REQ + "`,`" + requests\_const.FIO\_ZAK +  
 "`,`" + requests\_const.PHONE + "`,`" + requests\_const.FIZ\_UR + "`,`" + requests\_const.BANK + "`,`" +  
 requests\_const.NUM\_BANK + "`,`" + requests\_const.ADDRESS + "`,`" + requests\_const.DISTRICT + "`,`" +  
 requests\_const.DATE\_REQUEST + "`,`" + requests\_const.DOC\_NAME + "`,`" + requests\_const.SROCHNOE + "`,`" +  
 requests\_const.DATE\_VIDACHI + "`,`" + requests\_const.PRICE\_DOC + "`,`" + requests\_const.DATE\_TECH + "`,`" +  
 requests\_const.TIME\_TECH + "`,`" + requests\_const.FIO\_NACH + "`,`" + requests\_const.FIO\_ISP + "`,`" +  
 requests\_const.DATE\_PEREDACHI + "`,`" + requests\_const.FIO\_TECH + "`,`" + requests\_const.PAY + "`,`" +  
 requests\_const.DATE\_PAY + "`,`" + requests\_const.PAY\_SUM + "`,`" + requests\_const.DOC\_ISSUED + "`,`" +  
 requests\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 void addCentralniy(String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + centralniy\_const.REQUESTS\_TABLE + "(`" + centralniy\_const.NUM\_REQ + "`,`" + centralniy\_const.FIO\_ZAK +  
 "`,`" + centralniy\_const.PHONE + "`,`" + centralniy\_const.FIZ\_UR + "`,`" + centralniy\_const.BANK + "`,`" +  
 centralniy\_const.NUM\_BANK + "`,`" + centralniy\_const.ADDRESS + "`,`" + centralniy\_const.DISTRICT + "`,`" +  
 centralniy\_const.DATE\_REQUEST + "`,`" + centralniy\_const.DOC\_NAME + "`,`" + centralniy\_const.SROCHNOE + "`,`" +  
 centralniy\_const.DATE\_VIDACHI + "`,`" + centralniy\_const.PRICE\_DOC + "`,`" + centralniy\_const.DATE\_TECH + "`,`" +  
 centralniy\_const.TIME\_TECH + "`,`" + centralniy\_const.FIO\_NACH + "`,`" + centralniy\_const.FIO\_ISP + "`,`" +  
 centralniy\_const.DATE\_PEREDACHI + "`,`" + centralniy\_const.FIO\_TECH + "`,`" + centralniy\_const.PAY + "`,`" +  
 centralniy\_const.DATE\_PAY + "`,`" + centralniy\_const.PAY\_SUM + "`,`" + centralniy\_const.DOC\_ISSUED + "`,`" +  
 centralniy\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void addZarechenskiy (String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
  
 String insert = "INSERT INTO " + zarechenskiy\_const.REQUESTS\_TABLE + "(`" + zarechenskiy\_const.NUM\_REQ + "`,`" + zarechenskiy\_const.FIO\_ZAK +  
 "`,`" + zarechenskiy\_const.PHONE + "`,`" + zarechenskiy\_const.FIZ\_UR + "`,`" + zarechenskiy\_const.BANK + "`,`" +  
 zarechenskiy\_const.NUM\_BANK + "`,`" + zarechenskiy\_const.ADDRESS + "`,`" + zarechenskiy\_const.DISTRICT + "`,`" +  
 zarechenskiy\_const.DATE\_REQUEST + "`,`" + zarechenskiy\_const.DOC\_NAME + "`,`" + zarechenskiy\_const.SROCHNOE + "`,`" +  
 zarechenskiy\_const.DATE\_VIDACHI + "`,`" + zarechenskiy\_const.PRICE\_DOC + "`,`" + zarechenskiy\_const.DATE\_TECH + "`,`" +  
 zarechenskiy\_const.TIME\_TECH + "`,`" + zarechenskiy\_const.FIO\_NACH + "`,`" + zarechenskiy\_const.FIO\_ISP + "`,`" +  
 zarechenskiy\_const.DATE\_PEREDACHI + "`,`" + zarechenskiy\_const.FIO\_TECH + "`,`" + zarechenskiy\_const.PAY + "`,`" +  
 zarechenskiy\_const.DATE\_PAY + "`,`" + zarechenskiy\_const.PAY\_SUM + "`,`" + zarechenskiy\_const.DOC\_ISSUED + "`,`" +  
 zarechenskiy\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void addProletarskiy (String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + proletarskiy\_const.REQUESTS\_TABLE + "(`" + proletarskiy\_const.NUM\_REQ + "`,`" + proletarskiy\_const.FIO\_ZAK +  
 "`,`" + proletarskiy\_const.PHONE + "`,`" + proletarskiy\_const.FIZ\_UR + "`,`" + proletarskiy\_const.BANK + "`,`" +  
 proletarskiy\_const.NUM\_BANK + "`,`" + proletarskiy\_const.ADDRESS + "`,`" + proletarskiy\_const.DISTRICT + "`,`" +  
 proletarskiy\_const.DATE\_REQUEST + "`,`" + proletarskiy\_const.DOC\_NAME + "`,`" + proletarskiy\_const.SROCHNOE + "`,`" +  
 proletarskiy\_const.DATE\_VIDACHI + "`,`" + proletarskiy\_const.PRICE\_DOC + "`,`" + proletarskiy\_const.DATE\_TECH + "`,`" +  
 proletarskiy\_const.TIME\_TECH + "`,`" + proletarskiy\_const.FIO\_NACH + "`,`" + proletarskiy\_const.FIO\_ISP + "`,`" +  
 proletarskiy\_const.DATE\_PEREDACHI + "`,`" + proletarskiy\_const.FIO\_TECH + "`,`" + proletarskiy\_const.PAY + "`,`" +  
 proletarskiy\_const.DATE\_PAY + "`,`" + proletarskiy\_const.PAY\_SUM + "`,`" + proletarskiy\_const.DOC\_ISSUED + "`,`" +  
 proletarskiy\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void addSovetskiy (String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + sovetskiy\_const.REQUESTS\_TABLE + "(`" + sovetskiy\_const.NUM\_REQ + "`,`" + sovetskiy\_const.FIO\_ZAK +  
 "`,`" + sovetskiy\_const.PHONE + "`,`" + sovetskiy\_const.FIZ\_UR + "`,`" + sovetskiy\_const.BANK + "`,`" +  
 sovetskiy\_const.NUM\_BANK + "`,`" + sovetskiy\_const.ADDRESS + "`,`" + sovetskiy\_const.DISTRICT + "`,`" +  
 sovetskiy\_const.DATE\_REQUEST + "`,`" + sovetskiy\_const.DOC\_NAME + "`,`" + sovetskiy\_const.SROCHNOE + "`,`" +  
 sovetskiy\_const.DATE\_VIDACHI + "`,`" + sovetskiy\_const.PRICE\_DOC + "`,`" + sovetskiy\_const.DATE\_TECH + "`,`" +  
 sovetskiy\_const.TIME\_TECH + "`,`" + sovetskiy\_const.FIO\_NACH + "`,`" + sovetskiy\_const.FIO\_ISP + "`,`" +  
 sovetskiy\_const.DATE\_PEREDACHI + "`,`" + sovetskiy\_const.FIO\_TECH + "`,`" + sovetskiy\_const.PAY + "`,`" +  
 sovetskiy\_const.DATE\_PAY + "`,`" + sovetskiy\_const.PAY\_SUM + "`,`" + sovetskiy\_const.DOC\_ISSUED + "`,`" +  
 sovetskiy\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void addPrivokzalniy (String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + privokzalniy\_const.REQUESTS\_TABLE + "(`" + privokzalniy\_const.NUM\_REQ + "`,`" + privokzalniy\_const.FIO\_ZAK +  
 "`,`" + privokzalniy\_const.PHONE + "`,`" + privokzalniy\_const.FIZ\_UR + "`,`" + privokzalniy\_const.BANK + "`,`" +  
 privokzalniy\_const.NUM\_BANK + "`,`" + privokzalniy\_const.ADDRESS + "`,`" + privokzalniy\_const.DISTRICT + "`,`" +  
 privokzalniy\_const.DATE\_REQUEST + "`,`" + privokzalniy\_const.DOC\_NAME + "`,`" + privokzalniy\_const.SROCHNOE + "`,`" +  
 privokzalniy\_const.DATE\_VIDACHI + "`,`" + privokzalniy\_const.PRICE\_DOC + "`,`" + privokzalniy\_const.DATE\_TECH + "`,`" +  
 privokzalniy\_const.TIME\_TECH + "`,`" + privokzalniy\_const.FIO\_NACH + "`,`" + privokzalniy\_const.FIO\_ISP + "`,`" +  
 privokzalniy\_const.DATE\_PEREDACHI + "`,`" + privokzalniy\_const.FIO\_TECH + "`,`" + privokzalniy\_const.PAY + "`,`" +  
 privokzalniy\_const.DATE\_PAY + "`,`" + privokzalniy\_const.PAY\_SUM + "`,`" + privokzalniy\_const.DOC\_ISSUED + "`,`" +  
 privokzalniy\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void addIspKorotkov(String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + isp\_korotkov\_const.REQUESTS\_TABLE + "(`" + isp\_korotkov\_const.NUM\_REQ + "`,`" + isp\_korotkov\_const.FIO\_ZAK +  
 "`,`" + isp\_korotkov\_const.PHONE + "`,`" + isp\_korotkov\_const.FIZ\_UR + "`,`" + isp\_korotkov\_const.BANK + "`,`" +  
 isp\_korotkov\_const.NUM\_BANK + "`,`" + isp\_korotkov\_const.ADDRESS + "`,`" + isp\_korotkov\_const.DISTRICT + "`,`" +  
 isp\_korotkov\_const.DATE\_REQUEST + "`,`" + isp\_korotkov\_const.DOC\_NAME + "`,`" + isp\_korotkov\_const.SROCHNOE + "`,`" +  
 isp\_korotkov\_const.DATE\_VIDACHI + "`,`" + isp\_korotkov\_const.PRICE\_DOC + "`,`" + isp\_korotkov\_const.DATE\_TECH + "`,`" +  
 isp\_korotkov\_const.TIME\_TECH + "`,`" + isp\_korotkov\_const.FIO\_NACH + "`,`" + isp\_korotkov\_const.FIO\_ISP + "`,`" +  
 isp\_korotkov\_const.DATE\_PEREDACHI + "`,`" + isp\_korotkov\_const.FIO\_TECH + "`,`" + isp\_korotkov\_const.PAY + "`,`" +  
 isp\_korotkov\_const.DATE\_PAY + "`,`" + isp\_korotkov\_const.PAY\_SUM + "`,`" + isp\_korotkov\_const.DOC\_ISSUED + "`,`" +  
 isp\_korotkov\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void addIspSoshkina(String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + isp\_soshkina\_const.REQUESTS\_TABLE + "(`" + isp\_soshkina\_const.NUM\_REQ + "`,`" + isp\_soshkina\_const.FIO\_ZAK +  
 "`,`" + isp\_soshkina\_const.PHONE + "`,`" + isp\_soshkina\_const.FIZ\_UR + "`,`" + isp\_soshkina\_const.BANK + "`,`" +  
 isp\_soshkina\_const.NUM\_BANK + "`,`" + isp\_soshkina\_const.ADDRESS + "`,`" + isp\_soshkina\_const.DISTRICT + "`,`" +  
 isp\_soshkina\_const.DATE\_REQUEST + "`,`" + isp\_soshkina\_const.DOC\_NAME + "`,`" + isp\_soshkina\_const.SROCHNOE + "`,`" +  
 isp\_soshkina\_const.DATE\_VIDACHI + "`,`" + isp\_soshkina\_const.PRICE\_DOC + "`,`" + isp\_soshkina\_const.DATE\_TECH + "`,`" +  
 isp\_soshkina\_const.TIME\_TECH + "`,`" + isp\_soshkina\_const.FIO\_NACH + "`,`" + isp\_soshkina\_const.FIO\_ISP + "`,`" +  
 isp\_soshkina\_const.DATE\_PEREDACHI + "`,`" + isp\_soshkina\_const.FIO\_TECH + "`,`" + isp\_soshkina\_const.PAY + "`,`" +  
 isp\_soshkina\_const.DATE\_PAY + "`,`" + isp\_soshkina\_const.PAY\_SUM + "`,`" + isp\_soshkina\_const.DOC\_ISSUED + "`,`" +  
 isp\_soshkina\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void addIspNovikov(String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + isp\_novikov\_const.REQUESTS\_TABLE + "(`" + isp\_novikov\_const.NUM\_REQ + "`,`" + isp\_novikov\_const.FIO\_ZAK +  
 "`,`" + isp\_novikov\_const.PHONE + "`,`" + isp\_novikov\_const.FIZ\_UR + "`,`" + isp\_novikov\_const.BANK + "`,`" +  
 isp\_novikov\_const.NUM\_BANK + "`,`" + isp\_novikov\_const.ADDRESS + "`,`" + isp\_novikov\_const.DISTRICT + "`,`" +  
 isp\_novikov\_const.DATE\_REQUEST + "`,`" + isp\_novikov\_const.DOC\_NAME + "`,`" + isp\_novikov\_const.SROCHNOE + "`,`" +  
 isp\_novikov\_const.DATE\_VIDACHI + "`,`" + isp\_novikov\_const.PRICE\_DOC + "`,`" + isp\_novikov\_const.DATE\_TECH + "`,`" +  
 isp\_novikov\_const.TIME\_TECH + "`,`" + isp\_novikov\_const.FIO\_NACH + "`,`" + isp\_novikov\_const.FIO\_ISP + "`,`" +  
 isp\_novikov\_const.DATE\_PEREDACHI + "`,`" + isp\_novikov\_const.FIO\_TECH + "`,`" + isp\_novikov\_const.PAY + "`,`" +  
 isp\_novikov\_const.DATE\_PAY + "`,`" + isp\_novikov\_const.PAY\_SUM + "`,`" + isp\_novikov\_const.DOC\_ISSUED + "`,`" +  
 isp\_novikov\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void addTechPopov(String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + tech\_popov\_const.REQUESTS\_TABLE + "(`" + tech\_popov\_const.NUM\_REQ + "`,`" + tech\_popov\_const.FIO\_ZAK +  
 "`,`" + tech\_popov\_const.PHONE + "`,`" + tech\_popov\_const.FIZ\_UR + "`,`" + tech\_popov\_const.BANK + "`,`" +  
 tech\_popov\_const.NUM\_BANK + "`,`" + tech\_popov\_const.ADDRESS + "`,`" + tech\_popov\_const.DISTRICT + "`,`" +  
 tech\_popov\_const.DATE\_REQUEST + "`,`" + tech\_popov\_const.DOC\_NAME + "`,`" + tech\_popov\_const.SROCHNOE + "`,`" +  
 tech\_popov\_const.DATE\_VIDACHI + "`,`" + tech\_popov\_const.PRICE\_DOC + "`,`" + tech\_popov\_const.DATE\_TECH + "`,`" +  
 tech\_popov\_const.TIME\_TECH + "`,`" + tech\_popov\_const.FIO\_NACH + "`,`" + tech\_popov\_const.FIO\_ISP + "`,`" +  
 tech\_popov\_const.DATE\_PEREDACHI + "`,`" + tech\_popov\_const.FIO\_TECH + "`,`" + tech\_popov\_const.PAY + "`,`" +  
 tech\_popov\_const.DATE\_PAY + "`,`" + tech\_popov\_const.PAY\_SUM + "`,`" + tech\_popov\_const.DOC\_ISSUED + "`,`" +  
 tech\_popov\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void addTechSimonov(String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + tech\_simonov\_const.REQUESTS\_TABLE + "(`" + tech\_simonov\_const.NUM\_REQ + "`,`" + tech\_simonov\_const.FIO\_ZAK +  
 "`,`" + tech\_simonov\_const.PHONE + "`,`" + tech\_simonov\_const.FIZ\_UR + "`,`" + tech\_simonov\_const.BANK + "`,`" +  
 tech\_simonov\_const.NUM\_BANK + "`,`" + tech\_simonov\_const.ADDRESS + "`,`" + tech\_simonov\_const.DISTRICT + "`,`" +  
 tech\_simonov\_const.DATE\_REQUEST + "`,`" + tech\_simonov\_const.DOC\_NAME + "`,`" + tech\_simonov\_const.SROCHNOE + "`,`" +  
 tech\_simonov\_const.DATE\_VIDACHI + "`,`" + tech\_simonov\_const.PRICE\_DOC + "`,`" + tech\_simonov\_const.DATE\_TECH + "`,`" +  
 tech\_simonov\_const.TIME\_TECH + "`,`" + tech\_simonov\_const.FIO\_NACH + "`,`" + tech\_simonov\_const.FIO\_ISP + "`,`" +  
 tech\_simonov\_const.DATE\_PEREDACHI + "`,`" + tech\_simonov\_const.FIO\_TECH + "`,`" + tech\_simonov\_const.PAY + "`,`" +  
 tech\_simonov\_const.DATE\_PAY + "`,`" + tech\_simonov\_const.PAY\_SUM + "`,`" + tech\_simonov\_const.DOC\_ISSUED + "`,`" +  
 tech\_simonov\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void addTechChokin(String numReq, String fiOzakText, String phoneText,  
 String fizUrText, String bankText, String numBankText,  
 String adressText, String districtText, String dateRequestText,  
 String docNameText, String srochnoeText, String dateVidachiText,  
 String priceDocText, String dateTechText, String timeTechText,  
 String fiOnachText, String fiOispText, String datePeredachiText,  
 String fioTechText, String payText, String datePayText,  
 String paySumText, String docIssuedText, String primechanieText)  
 {  
 String insert = "INSERT INTO " + tech\_chokin\_const.REQUESTS\_TABLE + "(`" + tech\_chokin\_const.NUM\_REQ + "`,`" + tech\_chokin\_const.FIO\_ZAK +  
 "`,`" + tech\_chokin\_const.PHONE + "`,`" + tech\_chokin\_const.FIZ\_UR + "`,`" + tech\_chokin\_const.BANK + "`,`" +  
 tech\_chokin\_const.NUM\_BANK + "`,`" + tech\_chokin\_const.ADDRESS + "`,`" + tech\_chokin\_const.DISTRICT + "`,`" +  
 tech\_chokin\_const.DATE\_REQUEST + "`,`" + tech\_chokin\_const.DOC\_NAME + "`,`" + tech\_chokin\_const.SROCHNOE + "`,`" +  
 tech\_chokin\_const.DATE\_VIDACHI + "`,`" + tech\_chokin\_const.PRICE\_DOC + "`,`" + tech\_chokin\_const.DATE\_TECH + "`,`" +  
 tech\_chokin\_const.TIME\_TECH + "`,`" + tech\_chokin\_const.FIO\_NACH + "`,`" + tech\_chokin\_const.FIO\_ISP + "`,`" +  
 tech\_chokin\_const.DATE\_PEREDACHI + "`,`" + tech\_chokin\_const.FIO\_TECH + "`,`" + tech\_chokin\_const.PAY + "`,`" +  
 tech\_chokin\_const.DATE\_PAY + "`,`" + tech\_chokin\_const.PAY\_SUM + "`,`" + tech\_chokin\_const.DOC\_ISSUED + "`,`" +  
 tech\_chokin\_const.PRIMECHANIE + "`)" + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
  
 try {  
 PreparedStatement prst = getDbConnection().prepareStatement(insert);  
 prst.setString(1, numReq);  
 prst.setString(2, fiOzakText);  
 prst.setString(3, phoneText);  
 prst.setString(4, fizUrText);  
 prst.setString(5, bankText);  
 prst.setString(6, numBankText);  
 prst.setString(7, adressText);  
 prst.setString(8, districtText);  
 prst.setString(9, dateRequestText);  
 prst.setString(10, docNameText);  
 prst.setString(11, srochnoeText);  
 prst.setString(12, dateVidachiText);  
 prst.setString(13, priceDocText);  
 prst.setString(14, dateTechText);  
 prst.setString(15, timeTechText);  
 prst.setString(16, fiOnachText);  
 prst.setString(17, fiOispText);  
 prst.setString(18, datePeredachiText);  
 prst.setString(19, fioTechText);  
 prst.setString(20, payText);  
 prst.setString(21, datePayText);  
 prst.setString(22, paySumText);  
 prst.setString(23, docIssuedText);  
 prst.setString(24, primechanieText);  
  
 prst.executeUpdate();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
}

**Configs.java**

package com.example.bti;  
  
public class Configs {  
 protected String dbHost = "127.0.0.1";  
 protected String dbPort = "3306";  
 protected String dbUser = "root";  
 protected String dbPass = "1234";  
 protected String dbName = "requests\_bti";  
}

**requests\_const.java**

package com.example.bti;  
  
public class requests\_const {  
 public static final String REQUESTS\_TABLE = "все\_заявки";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**centralniy\_const.java**

package com.example.bti;  
  
public class centralniy\_const {  
 public static final String REQUESTS\_TABLE = "рн\_центральный";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**privokzalniy\_const.java**

package com.example.bti;  
  
public class privokzalniy\_const {  
 public static final String REQUESTS\_TABLE = "рн\_привокзальный";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**proletarskiy\_const.java**

package com.example.bti;  
  
public class proletarskiy\_const {  
 public static final String REQUESTS\_TABLE = "рн\_пролетарский";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**sovetskiy\_const.java**

package com.example.bti;  
  
public class sovetskiy\_const {  
 public static final String REQUESTS\_TABLE = "рн\_советский";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**zarechenskiy\_const.java**

package com.example.bti;  
  
public class zarechenskiy\_const {  
 public static final String REQUESTS\_TABLE = "рн\_зареченский";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**isp\_korotkov\_const.java**

package com.example.bti;  
  
public class isp\_korotkov\_const {  
 public static final String REQUESTS\_TABLE = "исп\_коротков";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**isp\_novikov \_const.java**

package com.example.bti;  
  
public class isp\_novikov\_const {  
 public static final String REQUESTS\_TABLE = "исп\_новиков";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**isp\_soshkina\_const.java**

package com.example.bti;  
  
public class isp\_soshkina\_const {  
 public static final String REQUESTS\_TABLE = "исп\_сошкина";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**tech\_chokin\_const.java**

package com.example.bti;  
  
public class tech\_chokin\_const {  
 public static final String REQUESTS\_TABLE = "тех\_чокин";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**tech\_popov\_const.java**

package com.example.bti;  
  
public class tech\_popov\_const {  
 public static final String REQUESTS\_TABLE = "тех\_попов";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**tech\_simonov\_const.java**

package com.example.bti;  
  
public class tech\_simonov\_const {  
 public static final String REQUESTS\_TABLE = "тех\_симонов";  
 public static final String NUM\_REQ = "№ Заявки";  
 public static final String FIO\_ZAK = "ФИО заказчика";  
 public static final String PHONE = "Телефон заказчика";  
 public static final String FIZ\_UR = "Физическое/юридическое лицо";  
 public static final String BANK = "Банк заказчика";  
 public static final String NUM\_BANK = "Номер счета в банке";  
 public static final String ADDRESS = "Адрес объекта";  
 public static final String DISTRICT = "Район объекта";  
 public static final String DATE\_REQUEST = "Дата приема заявки";  
 public static final String DOC\_NAME = "Название документа";  
 public static final String SROCHNOE = "Срочное изготовление (да/нет)";  
 public static final String DATE\_VIDACHI = "Дата выдачи документа";  
 public static final String PRICE\_DOC = "Стоимость изготовления документа";  
 public static final String DATE\_TECH = "Дата выхода техника";  
 public static final String TIME\_TECH = "Время выхода техника";  
 public static final String FIO\_NACH = "ФИО начальника отдела";  
 public static final String FIO\_ISP = "ФИО исполнителя";  
 public static final String DATE\_PEREDACHI = "Дата передачи исполнителю";  
 public static final String FIO\_TECH = "ФИО техника";  
 public static final String PAY = "Оплата наличными (да/нет)";  
 public static final String DATE\_PAY = "Дата оплаты";  
 public static final String PAY\_SUM = "Оплаченная сумма";  
 public static final String DOC\_ISSUED = "Документ выдан (да/нет)";  
 public static final String PRIMECHANIE = "Примечания";  
}

**BTI.fxml**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.Cursor?>  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.MenuButton?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.effect.DropShadow?>  
<?import javafx.scene.effect.Glow?>  
<?import javafx.scene.image.Image?>  
<?import javafx.scene.image.ImageView?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.paint.Color?>  
<?import javafx.scene.shape.Line?>  
<?import javafx.scene.text.Font?>  
<?import javafx.scene.text.Text?>  
  
<AnchorPane fx:id="MainPanel" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="708.0" prefWidth="999.0" style="-fx-background-color: wite;" xmlns="http://javafx.com/javafx/18" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.bti.Controller">  
 <AnchorPane fx:id="TopPanel" layoutY="-2.0" prefHeight="45.0" prefWidth="999.0" style="-fx-background-color: #fff;">  
 <effect>  
 <DropShadow>  
 <color>  
 <Color opacity="0.41034480929374695" />  
 </color>  
 </DropShadow>  
 </effect>  
 </AnchorPane>  
 <AnchorPane fx:id="LeftPanel" layoutY="-3.0" prefHeight="712.0" prefWidth="234.0" style="-fx-background-color: #fff;">  
 <effect>  
 <DropShadow>  
 <color>  
 <Color opacity="0.41034480929374695" />  
 </color>  
 </DropShadow>  
 </effect>  
 <Text fx:id="BTIname" layoutX="110.0" layoutY="49.0" opacity="0.8" strokeType="OUTSIDE" strokeWidth="0.0" text="БТИ" textAlignment="CENTER" wrappingWidth="106.40331649780273">  
 <font>  
 <Font name="Arial" size="33.0" />  
 </font>  
 </Text>  
 <Text fx:id="BTIname2" layoutX="112.0" layoutY="68.0" strokeType="OUTSIDE" strokeWidth="0.0" text="БЮРО ТЕХНИЧЕСКОЙ ИНВЕНТАРИЗАЦИИ" textAlignment="CENTER" wrappingWidth="100.06915760040283">  
 <font>  
 <Font name="Arial" size="10.0" />  
 </font>  
 </Text>  
 <Line endX="100.0" layoutX="117.0" layoutY="114.0" startX="-100.0" />  
 <ImageView fx:id="logo" fitHeight="115.0" fitWidth="112.0" layoutX="-2.0" layoutY="2.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@../../../logo.png" />  
 </image>  
 </ImageView>  
 <Button fx:id="SaveButton" layoutX="19.0" layoutY="162.0" mnemonicParsing="false" prefHeight="36.0" prefWidth="198.0" text="Сохранить">  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor></Button>  
 <Text fx:id="ErrorText" fill="RED" layoutX="54.0" layoutY="143.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Не все поля заполнены" textAlignment="CENTER" />  
 <Text fx:id="DoneText" fill="#00dd12" layoutX="52.0" layoutY="143.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Заявка добавлена в БД" />  
 <Button fx:id="CleanButton" layoutX="19.0" layoutY="206.0" mnemonicParsing="false" prefHeight="36.0" prefWidth="198.0" text="Очистить">  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 </Button>  
 <Text fx:id="numReqText" layoutX="53.0" layoutY="274.0" strokeType="OUTSIDE" strokeWidth="0.0" textAlignment="CENTER" wrappingWidth="129.52148437499994" />  
 </AnchorPane>  
 <TextField fx:id="FIOzakInput" layoutX="614.0" layoutY="79.0" prefHeight="22.0" prefWidth="326.0" promptText="ФИО заказчика">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="PhoneInput" layoutX="614.0" layoutY="105.0" prefHeight="22.0" prefWidth="326.0" promptText="Телефон для связи с заказчиком">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="FizUrInput" layoutX="614.0" layoutY="131.0" prefHeight="22.0" prefWidth="326.0" promptText="Физическое или юридическое лицо">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="NumBankInput" layoutX="614.0" layoutY="184.0" prefHeight="22.0" prefWidth="326.0" promptText="Номер счета в банке">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="BankInput" layoutX="614.0" layoutY="157.0" prefHeight="22.0" prefWidth="326.0" promptText="Банк заказчика">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="AddressInput" layoutX="614.0" layoutY="210.0" prefHeight="22.0" prefWidth="326.0" promptText="Адрес объекта">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DistrictInput" layoutX="614.0" layoutY="237.0" prefHeight="22.0" prefWidth="326.0" promptText="Район объекта">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DocNameInput" layoutX="614.0" layoutY="289.0" prefHeight="22.0" prefWidth="326.0" promptText="Название документа">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DateRequestInput1" alignment="CENTER" layoutX="614.0" layoutY="263.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="SrochnoeInput" layoutX="614.0" layoutY="315.0" prefHeight="22.0" prefWidth="326.0" promptText="Срочное изготовление (да/нет)">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="PriceDocInput" layoutX="614.0" layoutY="367.0" prefHeight="22.0" prefWidth="326.0" promptText="Стоимость изготовления документа">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="TimeTechInput" layoutX="614.0" layoutY="419.0" prefHeight="22.0" prefWidth="326.0" promptText="Время выхода техника">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="FIOnachInput" layoutX="614.0" layoutY="445.0" prefHeight="22.0" prefWidth="326.0" promptText="ФИО начальника отдела">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="FIOispInput" layoutX="614.0" layoutY="471.0" prefHeight="22.0" prefWidth="326.0" promptText="ФИО исполнителя">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="FIOTechInput" layoutX="614.0" layoutY="523.0" prefHeight="22.0" prefWidth="326.0" promptText="ФИО техника">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="PayInput" layoutX="614.0" layoutY="549.0" prefHeight="22.0" prefWidth="326.0" promptText="Оплата наличными (да/нет)">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="PaySumInput" layoutX="614.0" layoutY="601.0" prefHeight="22.0" prefWidth="326.0" promptText="Оплаченная сумма">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DocIssuedInput" layoutX="614.0" layoutY="627.0" prefHeight="22.0" prefWidth="326.0" promptText="Документ выдан (да/нет)">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="PrimechanieInput" layoutX="614.0" layoutY="653.0" prefHeight="22.0" prefWidth="326.0" promptText="Примечания">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <Label fx:id="FIOzakLabel" layoutX="300.0" layoutY="80.0" prefHeight="20.0" prefWidth="362.0" text="ФИО заказчика:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="PhoneLabel" layoutX="300.0" layoutY="106.0" prefHeight="20.0" prefWidth="362.0" text="Телефон для связи с заказчиком:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="FizUrLabel" layoutX="300.0" layoutY="134.0" prefHeight="20.0" prefWidth="362.0" text="Физическое или юридическое лицо:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="BankLabel" layoutX="300.0" layoutY="157.0" prefHeight="20.0" prefWidth="362.0" text="Банк заказчика:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="NumBankLabel" layoutX="300.0" layoutY="183.0" prefHeight="20.0" prefWidth="362.0" text="Номер счета в банке:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="AddressLabel" layoutX="300.0" layoutY="211.0" prefHeight="20.0" prefWidth="362.0" text="Адрес объекта:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="RegionLabel" layoutX="300.0" layoutY="237.0" prefHeight="20.0" prefWidth="362.0" text="Район объекта:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="DateZaiavkiLabel" layoutX="300.0" layoutY="264.0" prefHeight="20.0" prefWidth="362.0" text="Дата приема заявки:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="DocNameLabel" layoutX="300.0" layoutY="290.0" prefHeight="20.0" prefWidth="362.0" text="Название документа:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="SrochnoeLabel" layoutX="300.0" layoutY="317.0" prefHeight="20.0" prefWidth="362.0" text="Срочное изготовление (да/нет):">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="DateVidachiLabel" layoutX="300.0" layoutY="342.0" prefHeight="20.0" prefWidth="362.0" text="Дата выдачи документа:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="PriceDocLabel" layoutX="300.0" layoutY="369.0" prefHeight="20.0" prefWidth="362.0" text="Стоимость изготовления документа:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="DateTechLabel" layoutX="300.0" layoutY="395.0" prefHeight="20.0" prefWidth="362.0" text="Дата выхода техника:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="TimeTechLabel" layoutX="300.0" layoutY="420.0" prefHeight="20.0" prefWidth="362.0" text="Время выхода техника:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="FIOnachLabel" layoutX="300.0" layoutY="447.0" prefHeight="20.0" prefWidth="362.0" text="ФИО начальника отдела:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="FIOispLabel" layoutX="300.0" layoutY="473.0" prefHeight="20.0" prefWidth="362.0" text="ФИО исполнителя:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="DatePeredachiLabel" layoutX="300.0" layoutY="498.0" prefHeight="20.0" prefWidth="362.0" text="Дата передачи исполнителю:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="FIOTechLabel" layoutX="300.0" layoutY="524.0" prefHeight="20.0" prefWidth="362.0" text="ФИО техника:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="PayLabel" layoutX="300.0" layoutY="551.0" prefHeight="20.0" prefWidth="362.0" text="Оплата наличными (да/нет):">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="DatePayLabel" layoutX="300.0" layoutY="576.0" prefHeight="20.0" prefWidth="362.0" text="Дата оплаты:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="PaySumLabel" layoutX="300.0" layoutY="602.0" prefHeight="20.0" prefWidth="362.0" text="Оплаченная сумма:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="DocIssuedLabel" layoutX="300.0" layoutY="628.0" prefHeight="20.0" prefWidth="362.0" text="Документ выдан (да/нет):">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <Label fx:id="PrimechanieLabel" layoutX="300.0" layoutY="653.0" prefHeight="20.0" prefWidth="362.0" text="Примечания:">  
 <font>  
 <Font name="Arial" size="14.0" />  
 </font>  
 </Label>  
 <MenuButton fx:id="DistrictMenu" layoutX="944.0" layoutY="235.0" mnemonicParsing="false" prefHeight="26.0" prefWidth="27.0" style="-fx-background-color: white;">  
 <effect>  
 <Glow />  
 </effect>  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 </MenuButton>  
 <MenuButton fx:id="FizUrMenu" layoutX="944.0" layoutY="129.0" mnemonicParsing="false" prefHeight="26.0" prefWidth="19.0" style="-fx-background-color: white;">  
 <effect>  
 <Glow />  
 </effect>  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 </MenuButton>  
 <MenuButton fx:id="SrochnoeMenu" layoutX="944.0" layoutY="313.0" mnemonicParsing="false" prefHeight="26.0" prefWidth="27.0" style="-fx-background-color: white;">  
 <effect>  
 <Glow />  
 </effect>  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 </MenuButton>  
 <MenuButton fx:id="PayMenu" layoutX="944.0" layoutY="547.0" mnemonicParsing="false" prefHeight="26.0" prefWidth="27.0" style="-fx-background-color: white;">  
 <effect>  
 <Glow />  
 </effect>  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 </MenuButton>  
 <MenuButton fx:id="DocIssuedMenu" layoutX="944.0" layoutY="624.0" mnemonicParsing="false" prefHeight="26.0" prefWidth="27.0" style="-fx-background-color: white;">  
 <effect>  
 <Glow />  
 </effect>  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 </MenuButton>  
 <TextField fx:id="DateRequestInput2" alignment="CENTER" layoutX="723.0" layoutY="263.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DateRequestInput3" alignment="CENTER" layoutX="832.0" layoutY="263.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DateVidachiInput1" alignment="CENTER" layoutX="614.0" layoutY="341.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DateVidachiInput2" alignment="CENTER" layoutX="723.0" layoutY="341.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DateVidachiInput3" alignment="CENTER" layoutX="832.0" layoutY="341.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DateTechInput1" alignment="CENTER" layoutX="614.0" layoutY="393.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DateTechInput2" alignment="CENTER" layoutX="723.0" layoutY="393.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DateTechInput3" alignment="CENTER" layoutX="832.0" layoutY="393.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DatePeredachiInput1" alignment="CENTER" layoutX="614.0" layoutY="497.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DatePeredachiInput2" alignment="CENTER" layoutX="723.0" layoutY="497.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DatePeredachiInput3" alignment="CENTER" layoutX="832.0" layoutY="497.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DatePayInput1" alignment="CENTER" layoutX="614.0" layoutY="575.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DatePayInput2" alignment="CENTER" layoutX="723.0" layoutY="575.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="DatePayInput3" alignment="CENTER" layoutX="832.0" layoutY="575.0" prefHeight="22.0" prefWidth="108.0" promptText="ХХХХ">  
 <font>  
 <Font size="10.0" />  
 </font>  
 </TextField>  
 <MenuButton fx:id="FIOispMenu" layoutX="944.0" layoutY="469.0" mnemonicParsing="false" prefHeight="26.0" prefWidth="27.0" style="-fx-background-color: white;">  
 <effect>  
 <Glow />  
 </effect>  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 </MenuButton>  
 <MenuButton fx:id="FIOTechMenu" layoutX="944.0" layoutY="521.0" mnemonicParsing="false" prefHeight="26.0" prefWidth="27.0" style="-fx-background-color: white;">  
 <effect>  
 <Glow />  
 </effect>  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 </MenuButton>  
</AnchorPane>