Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рыбинский государственный авиационный технический университет

имени П. А. Соловьева»

Институт «Информационные технологии и системы управления»

Кафедра математического и программного обеспечения электронных вычислительных средств

**КУРСОВАЯ РАБОТА**

Тестирование программы Торговая фирма

по дисциплине «Тестирование и отладка программного обеспечения»

**Пояснительная записка**

|  |  |
| --- | --- |
| Студент группы ИПБ-20 | Никанов П. А. |
| Руководитель | Пруктишина В. А. |

Рыбинск 2023

Оглавление

[1. Постановка задачи 3](#_Toc155742559)

[2. Описание программы 4](#_Toc155742560)

[3. Управляющие графы программы 5](#_Toc155742561)

[4. Наборы тестов для тестирования по критериям С0, С1, С2 9](#_Toc155742562)

[4.1. Тестирование управляющего графа авторизации 10](#_Toc155742563)

[4.2. Тестирование управляющего графа администратора 10](#_Toc155742564)

[4.3. Тестирование управляющего графа кладовщика 11](#_Toc155742565)

[4.4. Тестирование управляющего графа поставщика 12](#_Toc155742566)

[5. Описание модулей программы 15](#_Toc155742567)

[6. Методика модульного тестирования 16](#_Toc155742568)

[7. Технологии, используемые для написания тестов 19](#_Toc155742569)

[8. Результаты модульного тестирования 20](#_Toc155742570)

[9. Системное тестирование 24](#_Toc155742571)

[9.1. Функциональное тестирование 24](#_Toc155742572)

[9.2. Нефункциональное тестирование 43](#_Toc155742573)

[10. Заключение 45](#_Toc155742574)

[11. Список использованных источников 46](#_Toc155742575)

[Приложение А 47](#_Toc155742576)

[Приложение Б 48](#_Toc155742577)

[Приложение В 49](#_Toc155742578)

[Приложение Г 50](#_Toc155742579)

# Постановка задачи

В ходе выполнения курсовой работы необходимо:

1. Выбрать и согласовать программу для тестирования. Кратко описать суть программы, ее функционал, интерфейсы/модули, которые предполагается тестировать.

2. Создать наборы тестов для тестирования по критериям С0, С1, С2, управляющий граф программы, описать наборы входных данных для путей графа.

3. Обосновать и реализовать набор модульных тестов (применить и предоставить результаты тестирования).

4. Обосновать и реализовать набор интеграционных тестов (применить и предоставить результаты тестирования).

5. Разработать все необходимое для проведения системного тестирования (система тестов для функционального и нефункционального тестирования, предложения по автоматизации тестирования

# Описание программы

Приложение предназначено для управления и хранения информации о товарах, кладовщиках и поставщиках, для заказа товаров у поставщиков, учёт приёмов и отпусков товаров и создания справок и накладных о совершённых действиях. Товар на фирму может поступать по ее заявкам или по предложению поставщика без предварительного заказа. Отпуск товаров производится по заявкам. Выходными документами в программе являются товарный ярлык, приходный ордер, накладная на отпуск товаров на сторону, отчёт о движении товарно-материальных ценностей в местах хранения.

Приложение написано средствами языка Java, связь с базой данных осуществляется при помощи JDBC-драйвера [6].

Действия в приложении:

* прием предложения поставщика о возможности поставить товар в виде накладной;
* прием товара на склад или отказ в приеме;
* формирование заказа поставщику на поставку товара;
* отпуск товара или отказ;
* получение справок о сделанных заказах поставщикам, о предложениях поставщиков, о заявках на отпуск товаров;
* редактирование и создание товара;
* редактирование и создание кладовщиков и поставщиков.

Модули для тестов:

* Авторизация;
* Администратор;
* Кладовщик;
* Поставщик.

# Управляющие графы программы

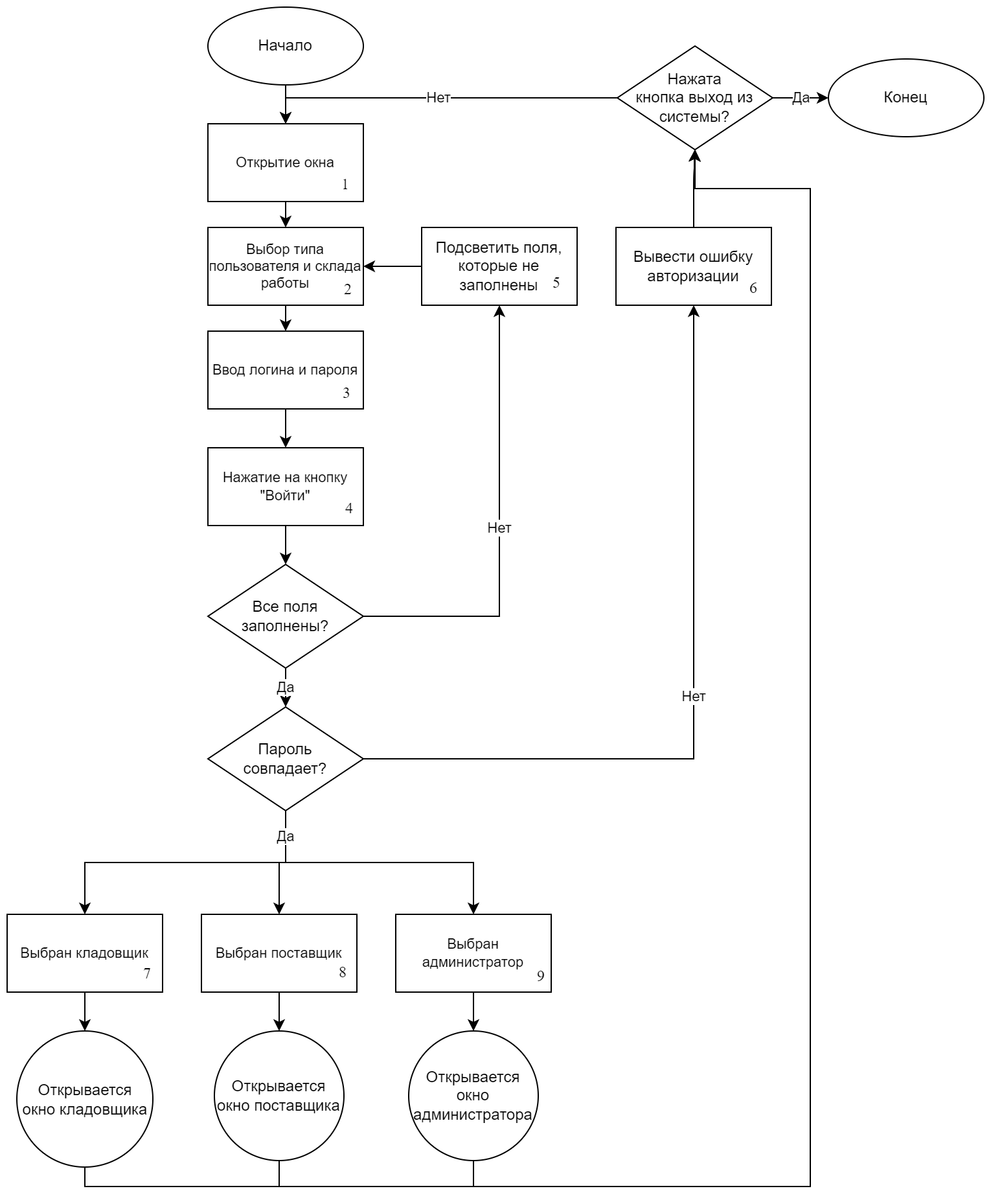


Рисунок 3.1 – Управляющий граф окна авторизации

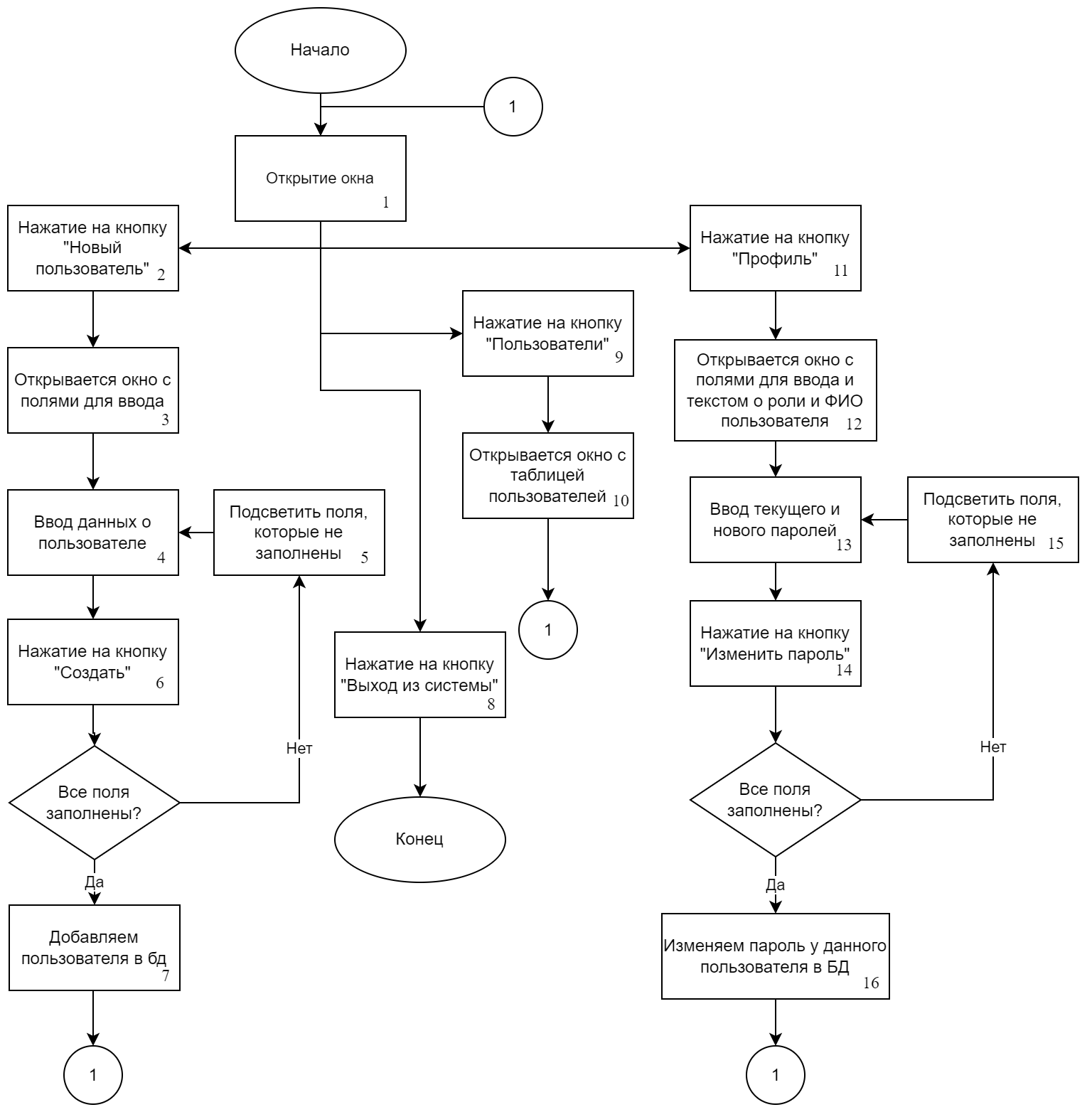


Рисунок 3.2 – Управляющий граф окна Администратора

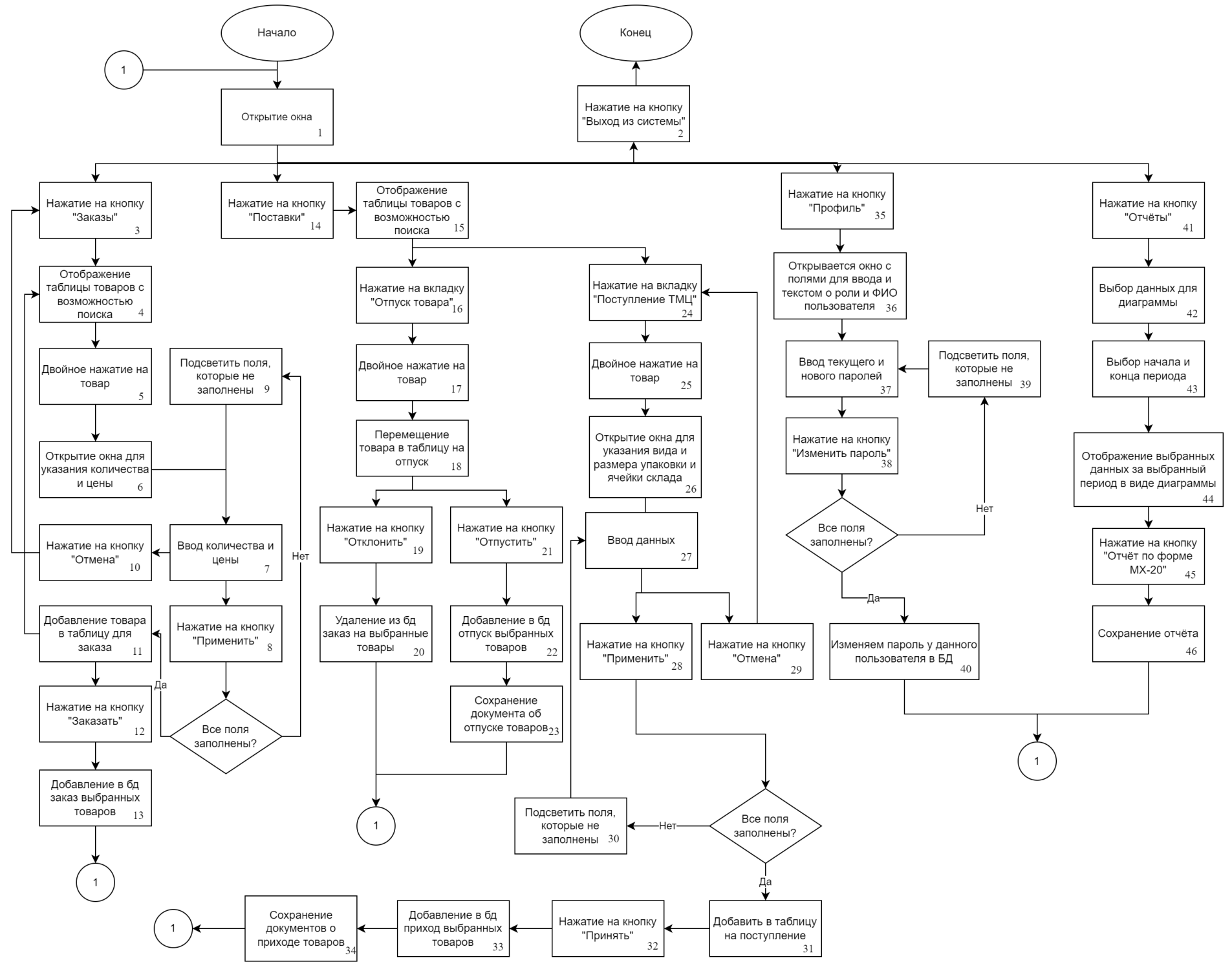


Рисунок 3.3 – Управляющий граф окна Кладовщика

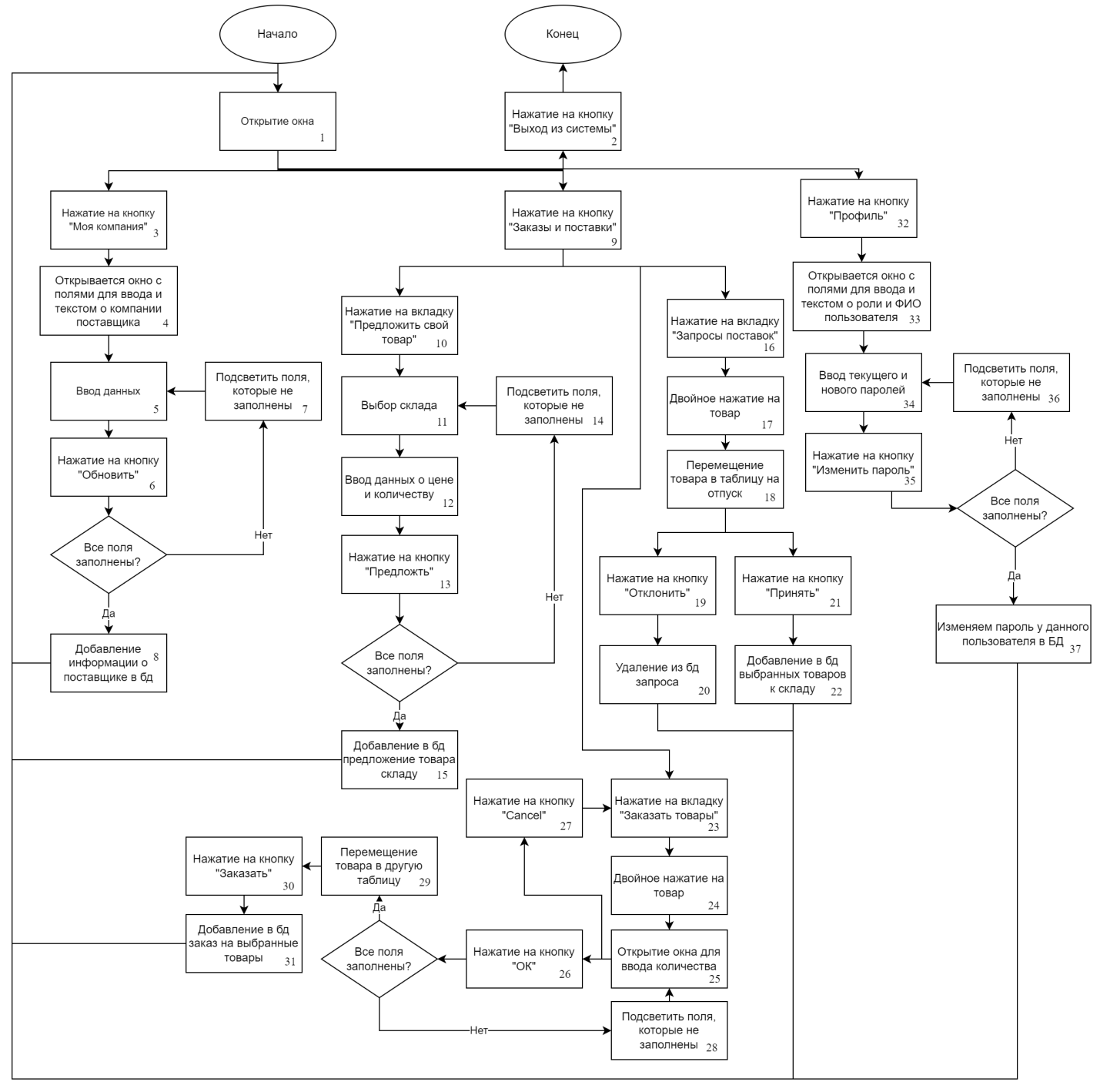


Рисунок 3.4 – Управляющий граф окна Поставщика

# Наборы тестов для тестирования по критериям С0, С1, С2

В тестировании стремятся выделить идеальный критерий тестирования. Существует множество совокупностей различных критериев, которые претендуют стать решением на задачу поиска идеального критерия. Один из таких критериев – структурный.

Структурные критерии используют модель программы в виде "белого ящика", что предполагает знание исходного текста программы или спецификации программы в виде потокового графа управления. Структурная информация понятна и доступна разработчикам подсистем и модулей приложения, поэтому данный класс критериев часто используется на этапах модульного и интеграционного тестирования (Unit testing, Integration testing).

Структурные критерии базируются на основных элементах УГП, операторах, ветвях и путях.

* Условие критерия тестирования команд (критерий С0) - набор тестов в совокупности должен обеспечить прохождение каждой команды не менее одного раза. Это слабый критерий, он, как правило, используется в больших программных системах, где другие критерии применить невозможно.
* Условие критерия тестирования ветвей (критерий С1) - набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза. Это достаточно сильный и при этом экономичный критерий, поскольку множество ветвей в тестируемом приложении конечно и не так уж велико. Данный критерий часто используется в системах автоматизации тестирования.
* Условие критерия тестирования путей (критерий С2) - набор тестов в совокупности должен обеспечить прохождение каждого пути не менее 1 раза. Если программа содержит цикл (в особенности с неявно заданным числом итераций), то число итераций ограничивается константой (часто - 2, или числом классов выходных путей).

## Тестирование управляющего графа авторизации

Критерии тестирования С0, С1, С2 для .

* Наборы тестов для тестирования по критерию C0:
  + 1-2-3-4-5-2-3-4-6-1-2-3-4-7-1-2-3-4-8-1-2-3-4-9.
* Наборы тестов для тестирования по критерию C1:
  + 4-5-2;
  + 4-6-1;
  + 4-7-1;
  + 4-8-1;
  + 4-9-1.
* Наборы тестов для тестирования по критерию C2:
  + 1-2-3-4-6;
  + 1-2-3-4-7;
  + 1-2-3-4-8;
  + 1-2-3-4-9;
  + 1-2-3-4.

## Тестирование управляющего графа администратора

Критерии тестирования С0, С1, С2 для .

* Наборы тестов для тестирования по критерию C0:
  + 1-2-3-4-6-5-4-6-7-1-9-10-1-11-12-13-14-15-13-14-16-1-8.
* Наборы тестов для тестирования по критерию C1:
  + 1-2-3-4-6;
  + 6-5-4;
  + 1-8;
  + 1-9-10;
  + 1-11-12-13-14;
  + 14-15-13.
* Наборы тестов для тестирования по критерию C2:
  + 1-2-3-4-6-5-4-6-7;
  + 1-2-3-4-6-7;
  + 1-8;
  + 1-9-10;
  + 1-11-12-13-14-16;
  + 1-11-12-13-14-15-13-14-16.

## Тестирование управляющего графа кладовщика

Критерии тестирования С0, С1, С2 для .

* Наборы тестов для тестирования по критерию C0:
  + 1-3-4-5-6-7-10-3-4-5-6-7-8-9-7-8-11-12-13-1-14-15-16-17-18-19-20-1-14-15-16-17-18-21-22-23-1-14-15-24-25-26-27-29-24-25-26-27-28-30-27-28-31-32-33-34-1-35-36-37-38-39-37-38-40-1-41-42-43-44-45-46-1-2.
* Наборы тестов для тестирования по критерию C1:
  + 1-2;
  + 1-3-4-5-6-7;
  + 1-3-4-5-6-7-8;
  + 1-14-15;
  + 1-35-36;
  + 1-41-42;
  + 8-9;
  + 8-11;
  + 15-16;
  + 15-24;
  + 18-19;
  + 18-21;
  + 27-28;
  + 27-29-24;
  + 28-30-27;
  + 28-31;
  + 38-39-37;
  + 38-40.
* Наборы тестов для тестирования по критерию C2:
  + 1-2;
  + 1-3-4-5-6-7-10-3-4-5-6-7-8-11-12-13;
  + 1-3-4-5-6-7-10-3-4-5-6-7-8-9-7-8-11-12-13;
  + 1-14-15-16-17-18-19-20;
  + 1-14-15-16-17-18-21-22-23;
  + 1-14-15-24-25-26-27-28-30-27-28-31-32-33-34;
  + 1-14-15-24-25-26-27-29-24-25-26-28-31-32-33-34;
  + 1-14-15-24-25-26-27-28-31-32-33-34;
  + 1-14-15-24-25-26-27-29-24-25-26-27-28-30-27-28-31-32-33-34;
  + 1-35-36-37-38-40;
  + 1-35-36-37-38-39-37-38-40;
  + 1-41-42-43-44-45-46.

## Тестирование управляющего графа поставщика

Критерии тестирования С0, С1, С2 для .

* Наборы тестов для тестирования по критерию C0:
  + 1-3-4-5-6-7-5-6-8-1-9-10-11-12-13-14-11-12-13-1-9-16-17-18-19-20-1-9-16-17-18-21-22-1-9-23-24-25-27-23-24-25-26-28-25-26-29-30-31-1-32-33-34-35-36-34-35-37-1-2.
* Наборы тестов для тестирования по критерию C1:
  + 1-2;
  + 1-3-4;
  + 1-9;
  + 1-32;
  + 6-7;
  + 6-8;
  + 13-14;
  + 13-15;
  + 9-10-11;
  + 9-23-24;
  + 9-16-17;
  + 18-19;
  + 18-21;
  + 25-27;
  + 25-26;
  + 26-28;
  + 26-29;
  + 35-36;
  + 35-37.
* Наборы тестов для тестирования по критерию C2:
  + 1-2;
  + 1-3-4-5-6-8;
  + 1-3-4-5-6-7-5-6-8;
  + 1-9-10-11-12-13;
  + 1-9-10-11-12-13-14-11-12-13;
  + 1-9-16-17-18-19-20;
  + 1-9-16-17-18-21-22;
  + 1-9-23-24-25-26-28-25-26-29-30-31;
  + 1-9-23-24-25-26-29-30-31;
  + 1-9-23-24-25-27-23-24-25-26-29-30-31;
  + 1-9-23-24-25-27-23-24-25-26-28-25-26-29-30-31;
  + 1-32-33-34-35-37;
  + 1-32-33-34-35-36-34-35-37.

# Описание модулей программы

* **Модуль «Авторизация»** - модуль, отвечающий за авторизацию пользователей и определения их роли;
* **Модуль «Администратор»** - модуль, отвечающий управление пользователями;
* **Модуль «Кладовщик»** - модуль, отвечающий за управление складом, а именно:
  + Заказ товаров на склад;
  + Поставки ТМЦ;
  + Отчёты;
  + Профиль.
* **Модуль «Поставщик»** - модуль, отвечающий за управления поставщиком, а именно:
  + Моя компания;
  + Заказы / поставки товаров;
  + Профиль.

# Методика модульного тестирования

При проведении модульного тестирования необходимо проверить:

* **Модуль «Авторизация»** - корректность авторизации пользователей.
* **Модуль «Администратор»** - корректность создания пользователей.
* **Модуль «Кладовщик»** - корректность работы складского учёта.
* **Модуль «Поставщик»** - корректность работы поставщиков.

Для проведения тестирования подготовим ряд тестов, указанных в Таблице 6.1.

Таблица 6.1 - Описание тестов

| Номер | Вход | Ожидаемый результат |
| --- | --- | --- |
| 1 | **trade\_company.logic.administrator.NewPersonLogicTest.testCheckIfLoginUnique**: проверить на уникальность логин “login”. | Уникальный логин. |
| 2 | **trade\_company.logic.administrator.UserListLogicTest.testInitTableData**: проинициализировать данные в таблице. | Данные проинициализированы. |
| 3 | **trade\_company.logic.administrator.UserListLogicTest.testRefreshTable**: Получить данные по фильтру «». | Загружено больше 0 данных. |
| 4 | **trade\_company.logic.login.LoginLogicTest.testPersonIsAuthorized**: Можно-ли авторизоваться в качестве администратора. | Пользователь авторизован. |
| 5 | **trade\_company.logic.login.LoginLogicTest.testFindUserRoleByIdAdmin**: проверка существования роли администратора. | Роль существует. |
| 6 | **trade\_company.logic.login.LoginLogicTest.testFindUserRoleByIdStorkeeper**: проверка существования роли кладовщика. | Роль существует. |
| 7 | **trade\_company.logic.login.LoginLogicTest.testFindUserRoleByIdSupplier**: проверка существования роли поставщика. | Роль существует. |
| 8 | **trade\_company.logic.storekeeper.OrderProductLogicTest.testInitSuppliers**: инициализация поставщика | Инициализация успешна. |
| 9 | **trade\_company.logic.storekeeper.OrderProductLogicTest.testInitComboBoxCity**: инициализация списка городов. | Инициализация успешна. |
| 10 | **trade\_company.logic.storekeeper.OrderProductLogicTest.testSearch**: проверка работы поиска | Поиск выполняется корректно. |
| 11 | **trade\_company.logic.supplier.tabs.OrdersLogicTest.testInitProducts**: инициализация продуктов | Инициализация успешна. |
| 12 | **trade\_company.logic.supplier.tabs.OrdersLogicTest.testSearch**: проверка работы поиска | Поиск работает корректно. |
| 13 | **trade\_company.logic.supplier.tabs.OrdersLogicTest.testInitWarehouses**: инициализация склада | Инициализация успешна. |
| 14 | **trade\_company.logic.supplier.tabs.RequestsControllerLogicTest.testInitWarehouses**: инициализация склада | Инициализация успешна. |

# Технологии, используемые для написания тестов

В ходе создания автоматических тестов использовалась библиотека JUnit [7].

JUnit – это Java фреймворк для тестирования, т. е. тестирования отдельных участков кода, например, методов или классов. Опыт, полученный при работе с JUnit, важен в разработке концепций тестирования программного обеспечения.

JUnit позволяет в любой момент быстро убедиться в работоспособности кода. Если программа не является совсем простой и включает множество классов и методов, то для её проверки может потребоваться значительное время. Естественно, что данный процесс лучше автоматизировать. Использование JUnit позволяет проверить код программы без значительных усилий и не занимает много времени.

# Результаты модульного тестирования

Результаты модульного тестирования указаны в Таблице 8.1. Результат работы всех тестов изображены на Рисунке 8.1.

Таблица 8.1 - Результаты модульного тестирования

| Номер | Вход | Ожидаемый результат | Результат работы программы |
| --- | --- | --- | --- |
| 1 | **trade\_company.logic.administrator.NewPersonLogicTest.testCheckIfLoginUnique**: проверить на уникальность логин “login”. | Уникальный логин. | Корректно |
| 2 | **trade\_company.logic.administrator.UserListLogicTest.testInitTableData**: проинициализировать данные в таблице. | Данные проинициализированы. | Корректно |
| 3 | **trade\_company.logic.administrator.UserListLogicTest.testRefreshTable**: Получить данные по фильтру «». | Загружено больше 0 данных. | Корректно |
| 4 | **trade\_company.logic.login.LoginLogicTest.testPersonIsAuthorized**: Можно-ли авторизоваться в качестве администратора. | Пользователь авторизован. | Корректно |
| 5 | **trade\_company.logic.login.LoginLogicTest.testFindUserRoleByIdAdmin**: проверка существования роли администратора. | Роль существует. | Корректно |
| 6 | **trade\_company.logic.login.LoginLogicTest.testFindUserRoleByIdStorkeeper**: проверка существования роли кладовщика. | Роль существует. | Корректно |
| 7 | **trade\_company.logic.login.LoginLogicTest.testFindUserRoleByIdSupplier**: проверка существования роли поставщика. | Роль существует. | Корректно |
| 8 | **trade\_company.logic.storekeeper.OrderProductLogicTest.testInitSuppliers**: инициализация поставщика | Инициализация успешна. | Корректно |
| 9 | **trade\_company.logic.storekeeper.OrderProductLogicTest.testInitComboBoxCity**: инициализация списка городов. | Инициализация успешна. | Корректно |
| 10 | **trade\_company.logic.storekeeper.OrderProductLogicTest.testSearch**: проверка работы поиска | Поиск выполняется корректно. | Корректно |
| 11 | **trade\_company.logic.supplier.tabs.OrdersLogicTest.testInitProducts**: инициализация продуктов | Инициализация успешна. | Корректно |
| 12 | **trade\_company.logic.supplier.tabs.OrdersLogicTest.testSearch**: проверка работы поиска | Поиск работает корректно. | Корректно |
| 13 | **trade\_company.logic.supplier.tabs.OrdersLogicTest.testInitWarehouses**: инициализация склада | Инициализация успешна. | Корректно |
| 14 | **trade\_company.logic.supplier.tabs.RequestsControllerLogicTest.testInitWarehouses**: инициализация склада | Инициализация успешна. | Корректно |

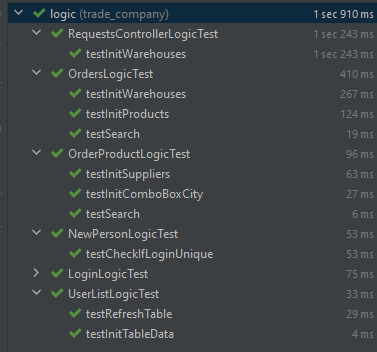


Рисунок 8.1 - Результаты тестирования

# Системное тестирование

## Функциональное тестирование

Торговая фирма занимается оптовой торговлей. Фирма имеет сеть поставщиков товаров. Поставщики находятся в разных городах. Каждый поставщик специализируется на поставке одного товара, склады фирмы позволяют разместить ограниченное количество каждого из товаров.

Товар на фирму может поступать по ее заявкам или по предложению поставщика без предварительного заказа. Отпуск товаров производится по заявкам.

Система должна обеспечивать выполнение следующих операций:

* прием предложения поставщика о возможности поставить товар в виде накладной;
* прием товара на склад или отказ в приеме;
* формирование заказа поставщику на поставку товара;
* отпуск товара или отказ;
* получение справок о сделанных заказах поставщикам, о предложениях поставщиков, о заявках на отпуск товаров;
* получение других справок.

При запуске программы появляется стартовый экран, который представлен на Рисунке 9.1, он состоит из:

* Список типов пользователя состоит из 3 типов, указанных на Рисунке 9.2;
* Поле «Логин» - поле, в которое необходимо ввести логин для пользователя с данным типом;
* Поле «Пароль» - поле, в которое необходимо ввести пароль для данного пользователя;
* Выбор склада присутствует только у пользователя с типом «Кладовщик»;
* Кнопка «Войти» - кнопка, при нажатии на которую происходит вход или выводится ошибка о том, что пользователя не существует (как на Рисунке 9.3).

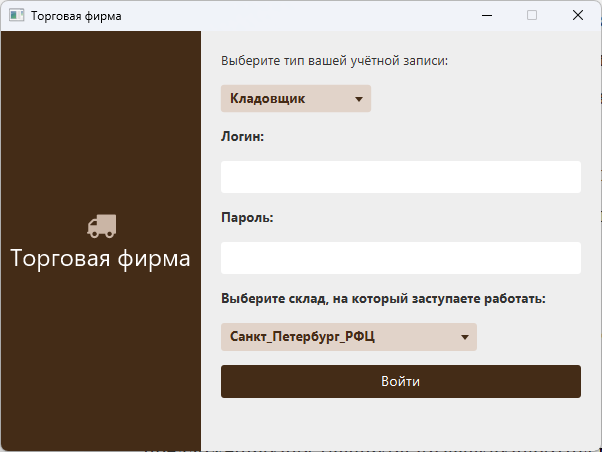


Рисунок 9.1 - Стартовый экран

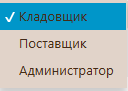


Рисунок 9.2 - Список типов пользователей

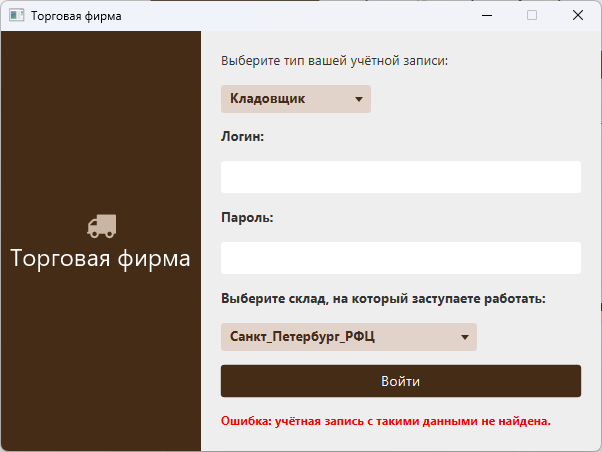


Рисунок 9.3 - Стартовый экран с ошибкой

После успешного входа с типом пользователя «Администратор» закрывается стартовый экран и открывается экран управления администратора (приведён на Рисунке 9.4), оно содержит следующие элементы:

* Кнопка «Нов. Пользователь» - открывает окно создания пользователя (как на Рисунке 9.5);
* Кнопка «Пользователи» - открывает окно просмотра существующих пользователей (как на Рисунке 9.6), на нём возможно просматривать должность, логин, фамилию, имя и отчества существующих пользователей с возможностью сортировки и фильтра по ФИО;
* Кнопка «Профиль» - открывает окно профиля (как на Рисунке 9.9), где указывается должность и ФИО залогиненного пользователя, а также возможностью смены пароля;
* Кнопка «Выход из системы» - закрывает активное окно и открывается стартовое окно.

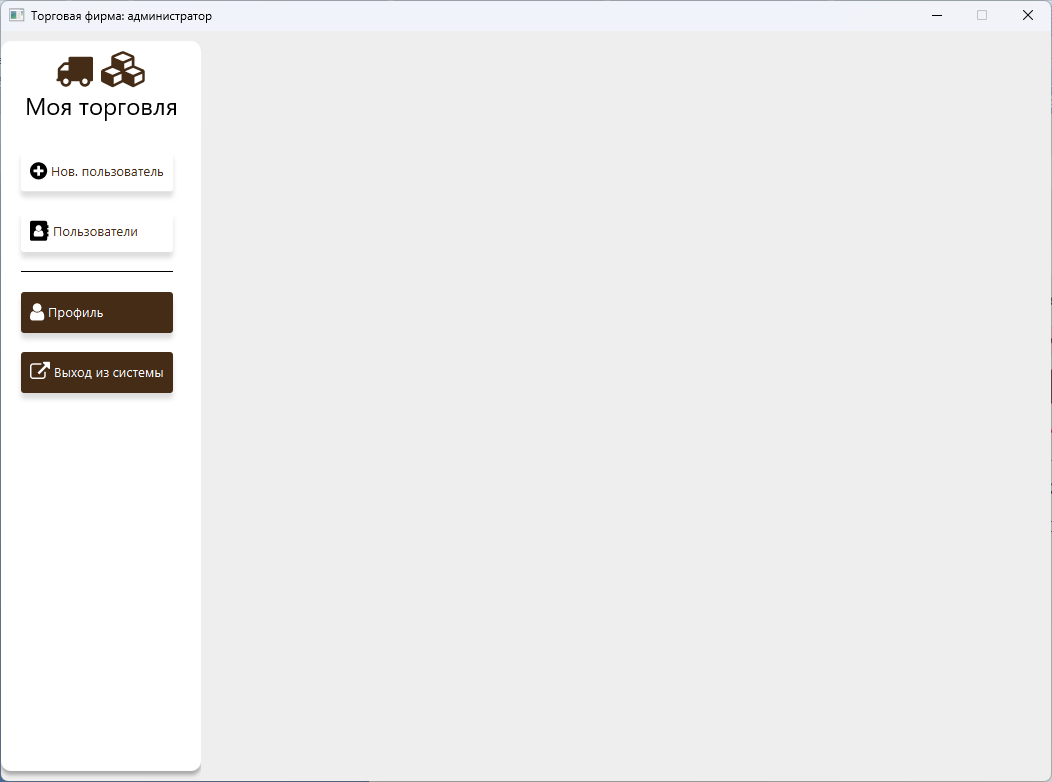


Рисунок 9.4 - Экран управления администратора

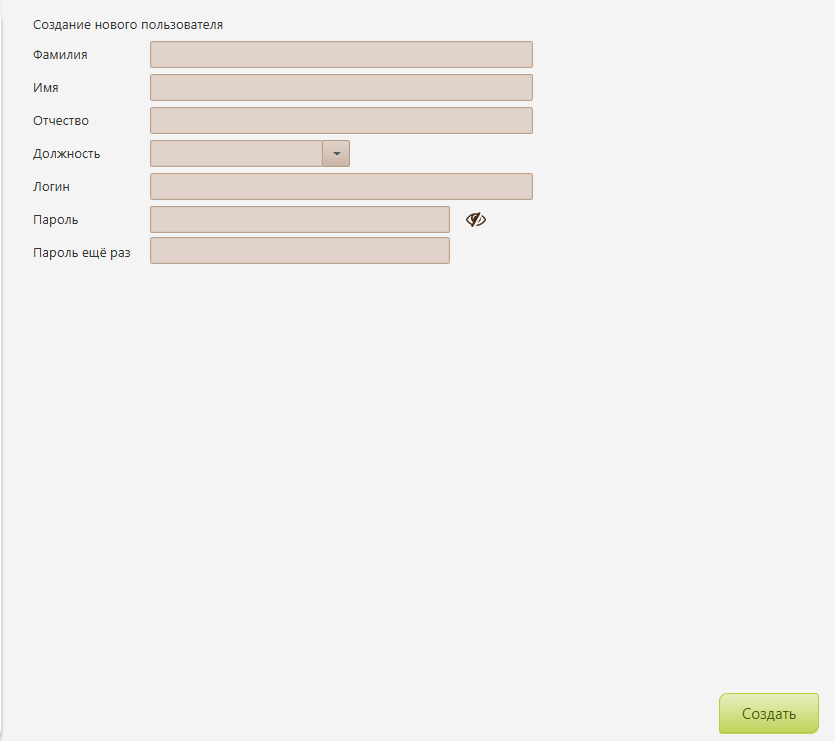


Рисунок 9.5 - Окно создания нового пользователя

На окне создания нового пользователя присутствуют следующие элементы:

* Поля для информации о пользователе;
* Кнопка «Создать» - создаёт пользователя, если все поля заполнены верно, иначе появляется сообщение об ошибке (как на Рисунке 9.6), где присутствует кнопка «ОК», которая закрывает окно об ошибке, а при несовпадении паролей появляется сообщение об ошибке (как на Рисунке 9.7), где присутствует кнопка «ОК», которая закрывает окно об ошибке.

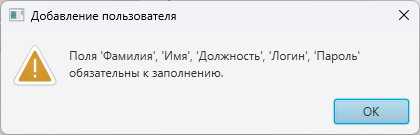


Рисунок 9.6 - Сообщение об ошибке при создании нового пользователя

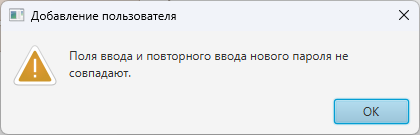


Рисунок 9.7 - Сообщение об ошибке при создании нового пользователя (2)

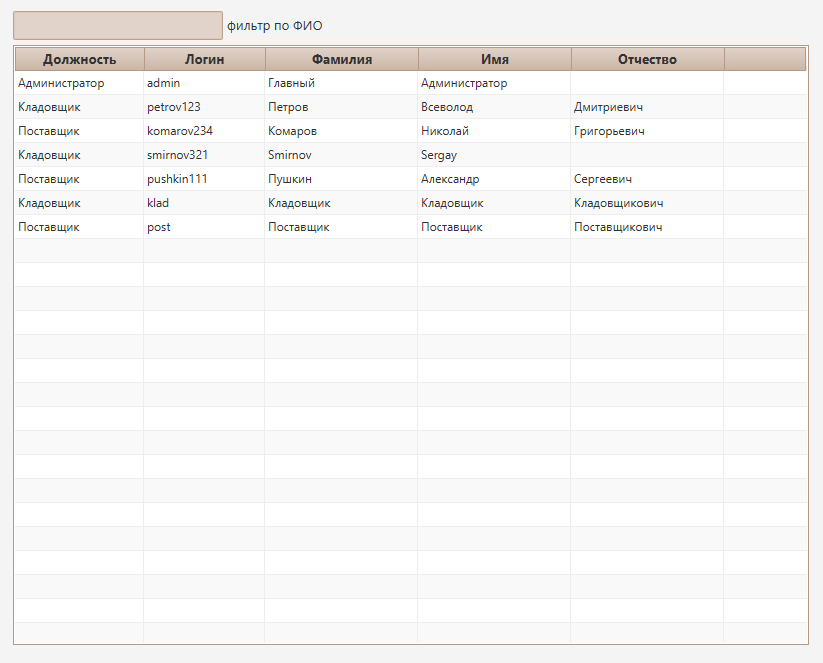


Рисунок 9.8 - Просмотр пользователей

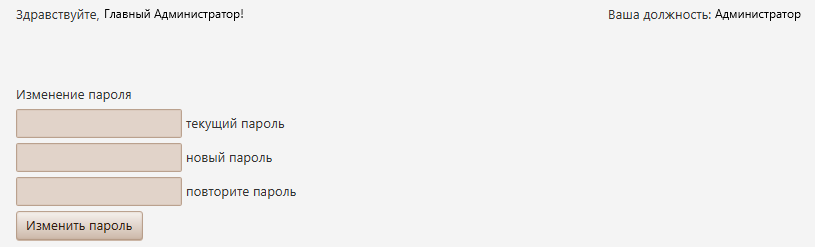


Рисунок 9.9 - Окно просмотра профиля

После успешного входа с типом пользователя «Кладовщик» закрывается стартовый экран и открывается экран управления складом (приведён на Рисунке 9.10), оно содержит следующие элементы:

* Кнопка «Заказы» - открывает окно заказов (как на Рисунке 9.11);
* Кнопка «Поставки» - открывает окно просмотра поставок (как на Рисунке 9.12);
* Кнопка «Отчёты» - открывает окно просмотра отчётов;
* Кнопка «Профиль» - открывает окно профиля (как на Рисунке 9.9), где указывается должность и ФИО залогиненного пользователя, а также возможностью смены пароля;
* Кнопка «Выход из системы» - закрывает активное окно и открывается стартовое окно.

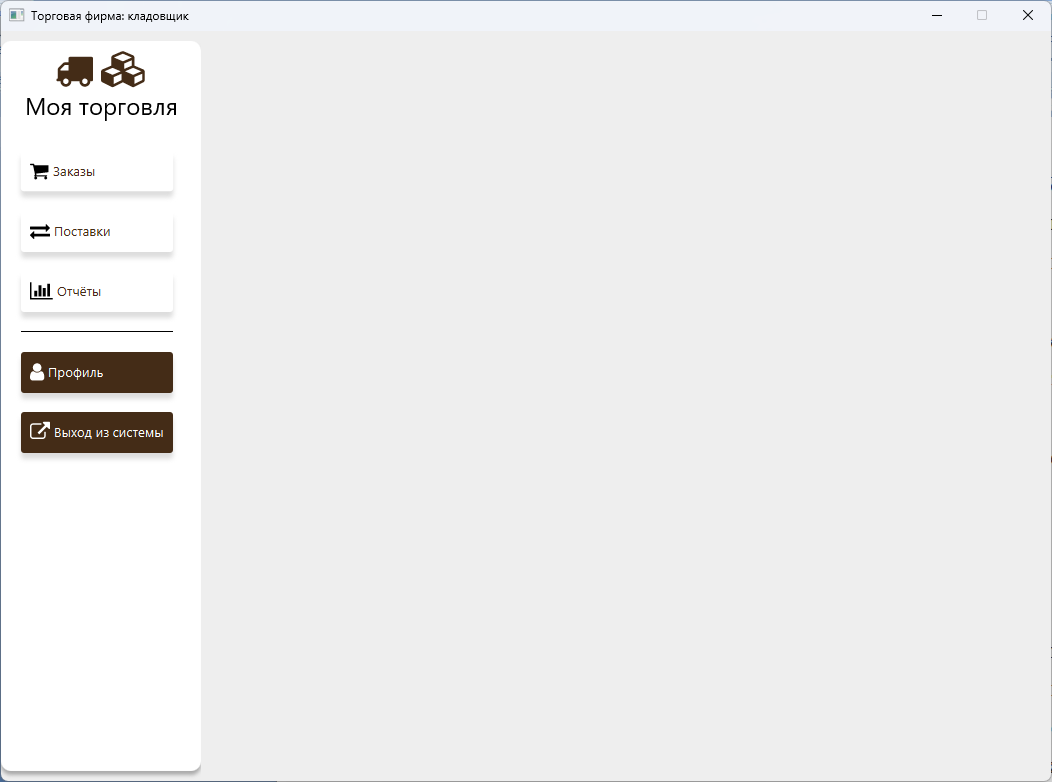


Рисунок 9.10 - Экран управления складом кладовщика

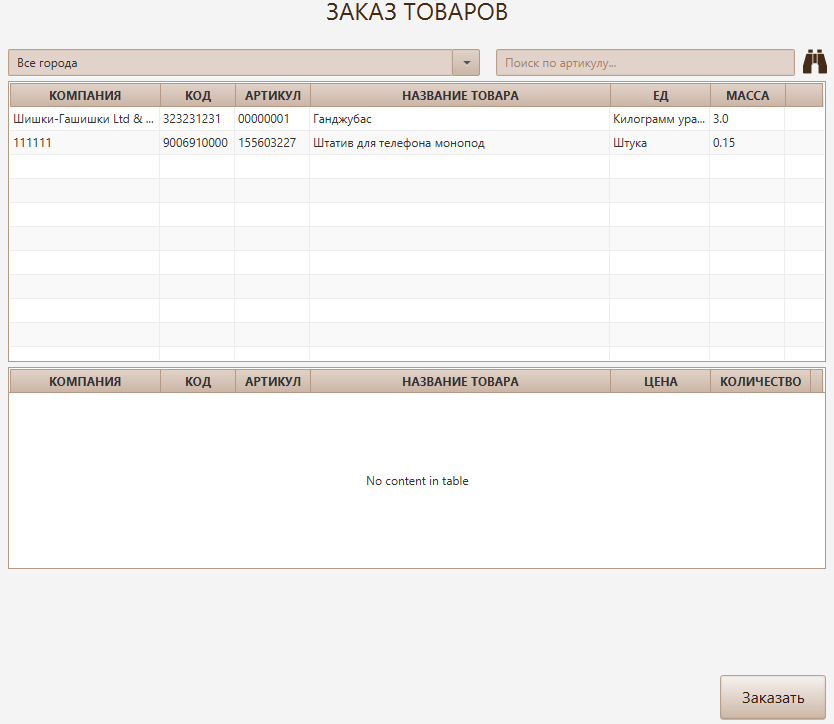


Рисунок 9.11 - Окно заказа товаров

Окно заказа товаров содержит в себе:

* Таблица заказа заполнена товарами, на которые можно нажать двойным кликом мыши, и откроется окно деталей заказа (как на Рисунке 9.12) с кнопками «Отмена», при нажатии которой товар не перемещается в таблицу очереди на заказ, и «Применить», которая сохраняет введённые данные и переводит выбранный товар в очередь на заказ;
* Кнопка «Заказать» - заказывает у поставщиков товары, которые в очереди на заказ.

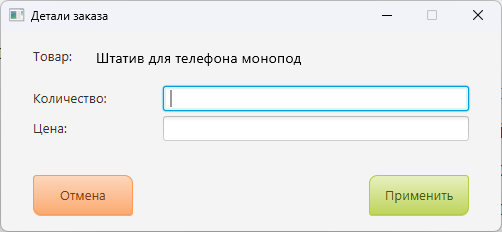


Рисунок 9.12 - Окно деталей заказа



Рисунок 9.13 - Окно просмотра поставок

При нажатии на кнопку «Поставки» открывается окно поступления ТМЦ (как на Рисунке 9.13), и на нём присутствует:

* Таблица прихода заполнена товарами, которые поставляются на, выбранный на начальном экране, склад, на которые можно нажать двойным кликом мыши и откроется окно заполнения деталей поступающего товара (как на Рисунке 9.14) с кнопками «Отмена», при нажатии которой товар не перемещается в таблицу на приём, и «Применить», которая сохраняет введённые данные и переводит выбранный товар на приём;
* Кнопка «Принять» - все товары из таблицы приёмки добавляются в базу данных вместе с ранее введёнными деталями поступления;
* Кнопка «Отпуск товара» - открывает окно отпуска товаров (как на Рисунке 9.15).

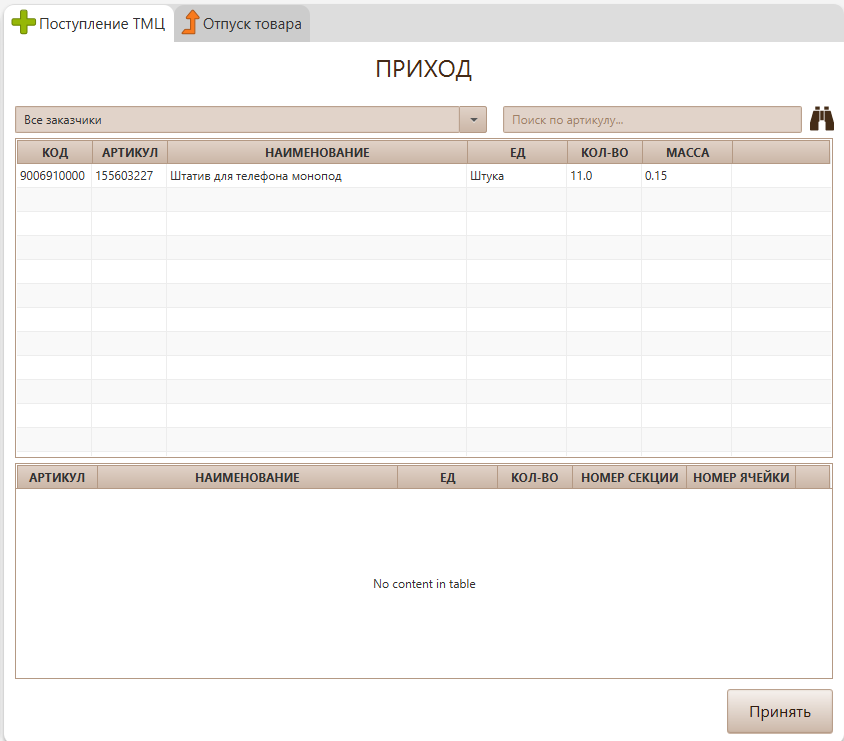


Рисунок 9.14 - Окно поступления товаров

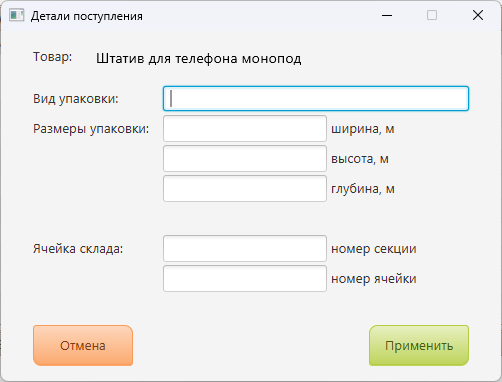


Рисунок 9.15 - Окно деталей поступления

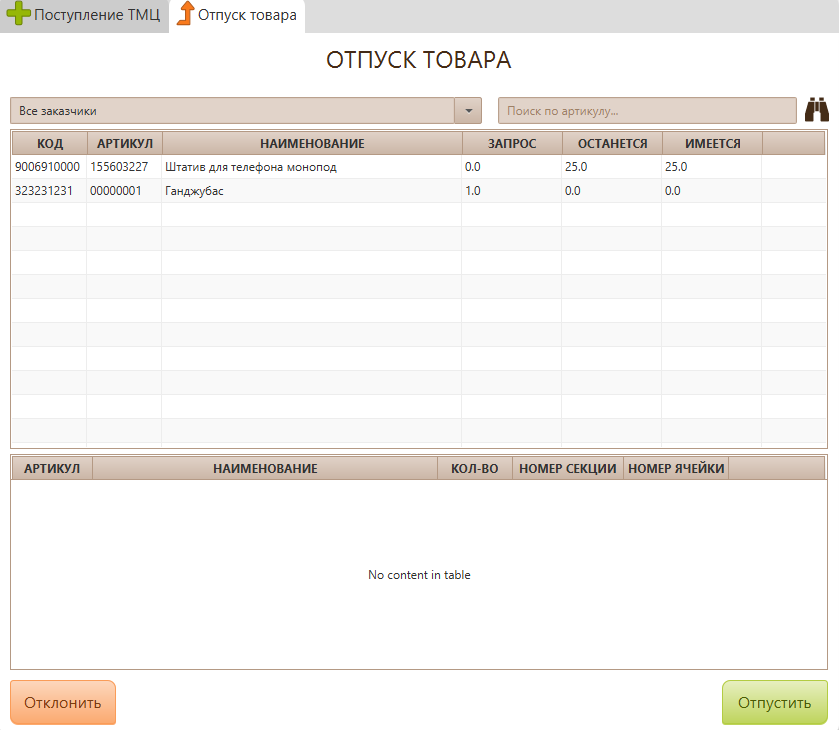


Рисунок 9.16 - Окно отпуска товаров

Окно отпуска товаров состоит из:

* Таблица отпуска заполнена товарами, которые отпускаются с, выбранного на начальном экране, складе, на которые можно нажать двойным кликом мыши и переводит выбранный товар в очередь на отпуск;
* Кнопка «Отклонить» - удаляет все запросы об отпуске товаров из очереди на отпуск;
* Кнопка «Отпустить» - удовлетворяет все запросы об отпуске товаров из очереди на отпуск.

После успешного входа с типом пользователя «Поставщик» закрывается стартовый экран и открывается экран управления поставщика (приведён на Рисунке 9.17), оно содержит следующие элементы:

* Кнопка «Моя компания» - открывает окно обновления компании (как на Рисунке 9.18), где можно вписать новую информацию о компании и, по нажатию кнопки «Обновить», она обновиться в базе данных;
* Кнопка «Заказы/поставки» - открывает окно заказов и поставок (как на Рисунке 9.19);
* Кнопка «Профиль» - открывает окно профиля (как на Рисунке 9.9), где указывается должность и ФИО залогиненного пользователя, а также возможностью смены пароля;
* Кнопка «Выход из системы» - закрывает активное окно и открывается стартовое окно.

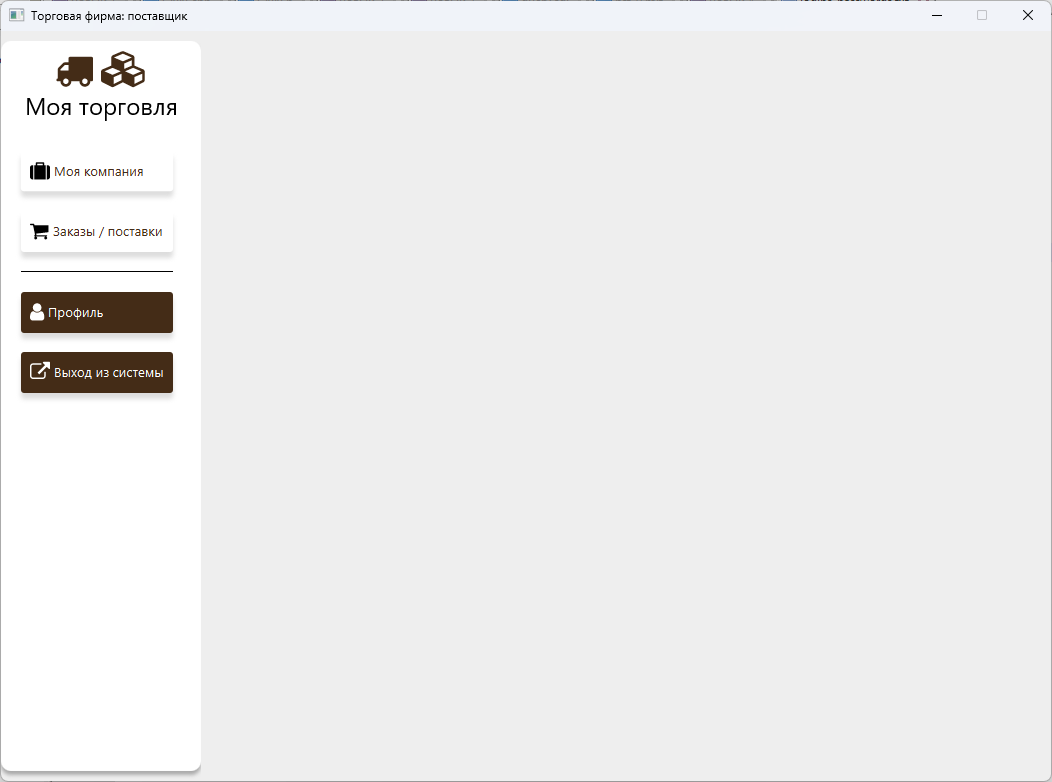


Рисунок 9.17 - Экран управления поставщика

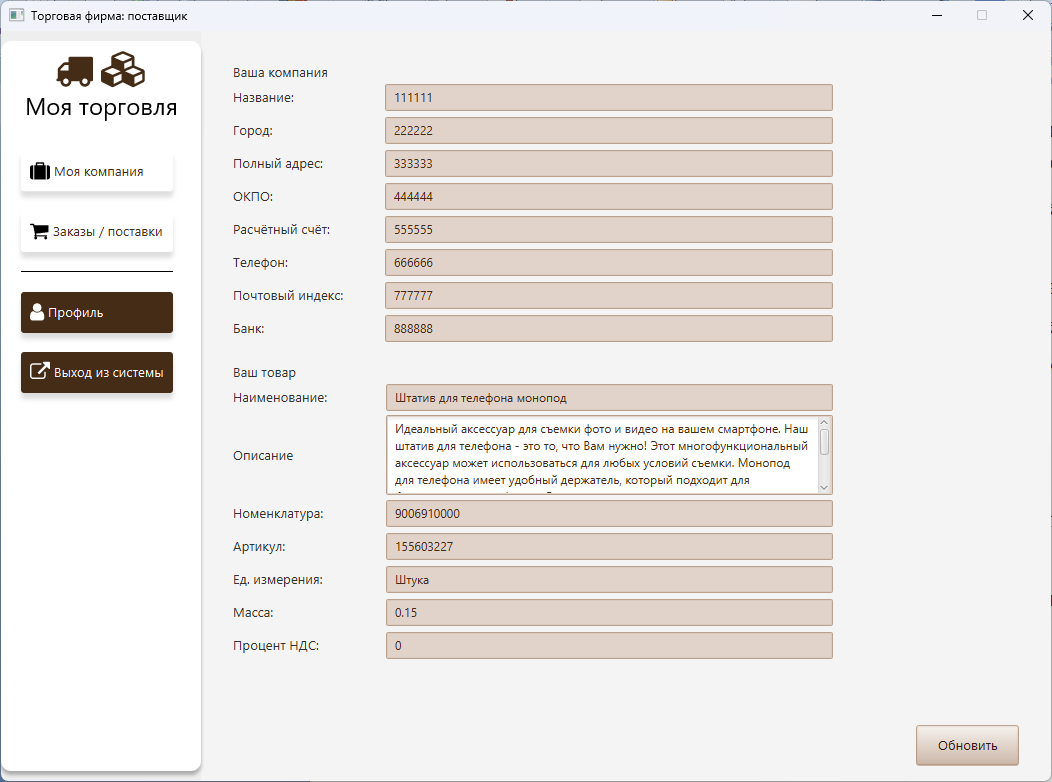


Рисунок 9.18 - Окно обновления компании



Рисунок 9.19 - Окно заказов и поставок

Окно заказов и поставок состоит из следующих элементов:

* Вкладки «Предложить свой товар» (как на );
* Вкладки «Запросы поставок» (как на );
* Вкладки «Заказать товар» (как на Рисунке 9.22).

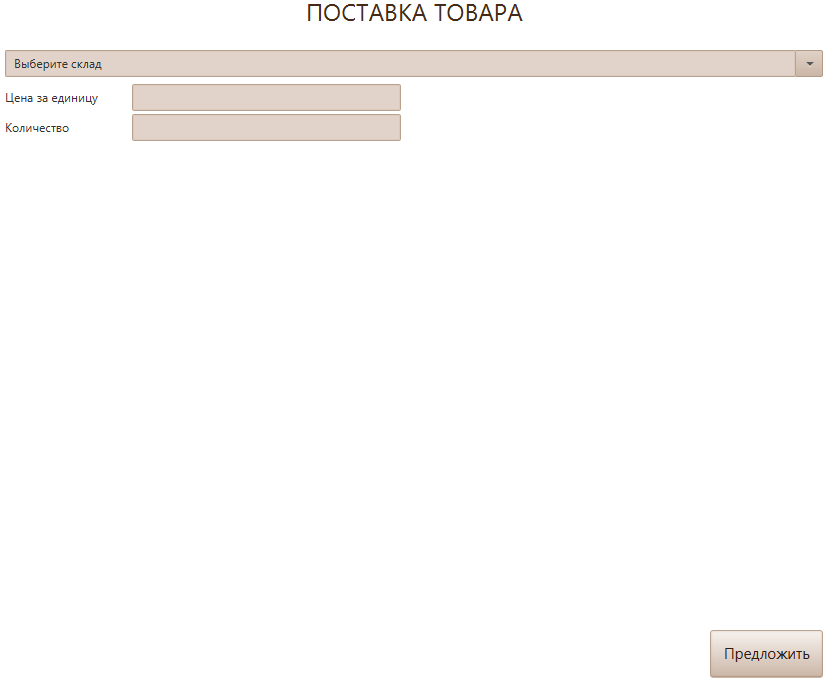


Рисунок 9.20 - Окно поставки товара

На окне поставки товара содержится:

* Выбор склада, на который будет поставляться товар данного поставщика;
* Поле цены за единицу;
* Поле количества;
* Кнопка «Предложить» - добавляет предложение выбранному складу свой товар указанного количества и за указанную цену.

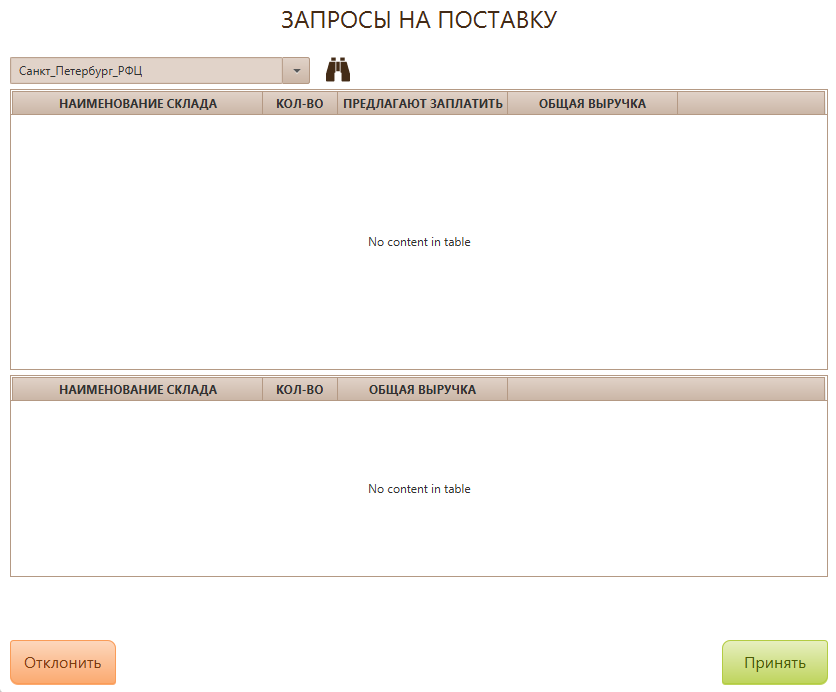


Рисунок 9.21 - Окно запроса поставок

На окне запроса поставок у поставщика содержится:

* Таблица отпуска заполнена складами, которые запросили товар у данного поставщика, на которые можно нажать двойным кликом мыши и переводит выбранный запрос в очередь на отпуск;
* Кнопка «Отклонить» - удаляет все запросы об отпуске товаров из очереди на отпуск;
* Кнопка «Отпустить» - удовлетворяет все запросы об отпуске товаров из очереди на отпуск.

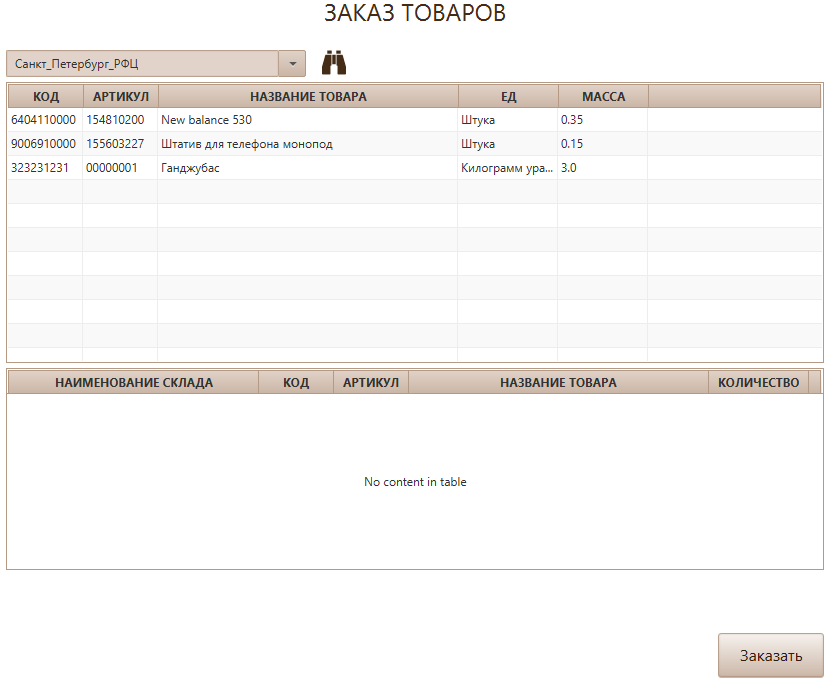


Рисунок 9.22 - Окно заказа товаров поставщиком

Окно заказа товаров поставщиком состоит из:

* Таблица заказа заполнена товарами, на которые можно нажать двойным кликом мыши, и откроется окно создания запроса (как на Рисунке 9.23) с кнопками «Cancel», при нажатии которой товар не перемещается в таблицу очереди на заказ, и «OK», которая сохраняет введённые данные и переводит выбранный товар в очередь на заказ;
* Кнопка «Заказать» - заказывает у складов товары, которые в очереди на заказ.

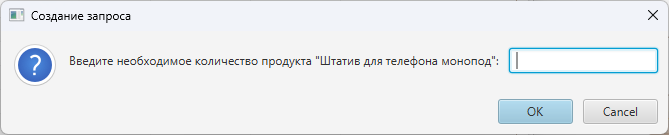


Рисунок 9.23 - Окно создания запроса

## Нефункциональное тестирование

В Таблице 9.1 представлено нефункциональное тестирование ПО.

Таблица 9.1 - Нефункциональное тестирование по типам

|  |  |
| --- | --- |
| Тип нефункционального тестирования | Метрика |
| Производительность | При одновременном доступе 3 пользователей программа использует не более 500 МБ памяти и не нагружает процессор более, чем на 10%. |
| Надёжность | В процессе работы система сохраняет данные о пользователях, складах и товарах. При внештатных ситуациях система прекращает работать |
| Безопасность | Для защиты любых непубличных данных используется аутентификация и авторизация пользователя. |
| Скорость | Обработка запроса в базу данных занимает не более 0.2 секунды;  Исполнение простых операций занимает не более 0.1 секунды;  Сохранение файлов в формате Word занимает не более 2 секунд;  Обработка сложных действий (выборка больших массивов данных или сложная фильтрация по критериям) занимает до 0.7 секунд. |
| Удобство | Интерфейс приложения интуитивно понятный, каждое действие прописано в самой программе. |
| Совместимость | ПО совместимо со всеми устройствами и ОС, поддерживающими Java 17. |
| Локализация | В ПО представлен только русский язык. |

# Заключение

В результате выполнения курсовой работы была протестирована программа калькулятор. Тщательный подход к изучению программы обеспечило составление УГП и набора модульных тестов по критериям С0, С1, С2. Чтобы написать программную реализацию тестирования я изучил библиотеки JUnit. Также я разобрался в различии модульных, интеграционных и системных тестов.

Результаты работы всех реализованных тестов оказались корректными. Таким образом, можно сделать вывод, что программа работает правильно в рамках созданных тестов.

# Список использованных источников

1. Савин Р. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах. Изд. Ridero, 2017;
2. GitHub [Электронный ресурс]. – Режим доступа: <https://github.com/paha2315/tradeCompany_With_Test>, Дата доступа: 02.01.2024;
3. Протестинг [Электронный ресурс]. – Режим доступа: <http://www.protesting.ru/testing/testlevels.html>, Дата доступа: 06.01.2024;
4. ИНТУИТ [Электронный ресурс]. – Режим доступа: <https://intuit.ru/studies/courses/48/48/lecture/1428>, Дата доступа: 05.01.2024;
5. Java-online [Электронный ресурс]. – Режим доступа: <http://java-online.ru/blog-junit.xhtml>, Дата доступа: 07.01.2024;
6. Oracle [Электронный ресурс]. – Режим доступа: <https://docs.oracle.com/en/database/oracle/oracle-database/19/jjdbc/jdbc-developers-guide.pdf>, Дата доступа: 07.01.2024;
7. JUnit [Электронный ресурс]. – Режим доступа: <https://junit.org/junit5/docs/current/user-guide/>, Дата доступа: 07.01.2024.

# Приложение А

Текст модуля LoginLogicTest

package trade\_company.logic.login;

import org.junit.Assert;

import org.junit.Before;

import org.junit.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import trade\_company.logic.sql\_object.Warehouse;

import trade\_company.views.options.LoginRolesOptions;

import java.util.ArrayList;

import java.util.Optional;

import static org.mockito.Mockito.\*;

public class LoginLogicTest {

@InjectMocks

LoginLogic loginLogic;

@Before

public void setUp() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testPersonIsAuthorized() throws Exception {

boolean result = loginLogic.personIsAuthorized("Администратор", "admin", "admin");

Assert.assertEquals(true, result);

}

@Test

public void testFindUserRoleByIdAdmin() throws Exception {

Optional<LoginRolesOptions> result = loginLogic.findUserRoleById(3);

Assert.assertEquals(LoginRolesOptions.SUPPLIER, result.get());

}

@Test

public void testFindUserRoleByIdStorkeeper() throws Exception {

Optional<LoginRolesOptions> result = loginLogic.findUserRoleById(2);

Assert.assertEquals(LoginRolesOptions.STOREKEEPER, result.get());

}

@Test

public void testFindUserRoleByIdSupplier() throws Exception {

Optional<LoginRolesOptions> result = loginLogic.findUserRoleById(1);

Assert.assertEquals(LoginRolesOptions.ADMIN, result.get());

}

}

# Приложение Б

Текст модуля NewPersonLogicTest

package trade\_company.logic.administrator;

import org.junit.Assert;

import org.junit.Before;

import org.junit.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import java.util.ArrayList;

import static org.mockito.Mockito.\*;

public class NewPersonLogicTest {

@InjectMocks

NewPersonLogic newPersonLogic;

@Before

public void setUp() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testCheckIfLoginUnique() throws Exception {

boolean result = newPersonLogic.checkIfLoginUnique("login");

Assert.assertEquals(true, result);

}

}

package trade\_company.logic.administrator;

import org.junit.Assert;

import org.junit.Before;

import org.junit.Test;

import org.mockito.InjectMocks;

import org.mockito.MockitoAnnotations;

public class UserListLogicTest {

@InjectMocks

UserListLogic userListLogic;

@Before

public void setUp() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testInitTableData() throws Exception {

userListLogic.initTableData();

}

@Test

public void testRefreshTable() throws Exception {

userListLogic.refreshTable("");

System.out.println(userListLogic.personData.size());

Assert.assertNotEquals(0, userListLogic.personData.size());

}

}

# Приложение В

Текст модуля OrderProductLogicTest

package trade\_company.logic.storekeeper;

import javafx.collections.ObservableList;

import org.junit.Assert;

import org.junit.Before;

import org.junit.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import trade\_company.logic.sql\_object.Order;

import trade\_company.logic.sql\_object.Supplier;

import java.util.ArrayList;

import static org.mockito.Mockito.\*;

public class OrderProductLogicTest {

@InjectMocks

OrderProductLogic orderProductLogic;

@Before

public void setUp() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testInitSuppliers() throws Exception {

orderProductLogic.initSuppliers();

}

@Test

public void testInitComboBoxCity() throws Exception {

orderProductLogic.initComboBoxCity();

}

@Test

public void testSearch() throws Exception {

orderProductLogic.search(orderProductLogic.ALL\_CITIES, "");

}

}

# Приложение Г

Текст модуля OrdersLogicTest

package trade\_company.logic.supplier.tabs;

import org.junit.Before;

import org.junit.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import trade\_company.logic.sql\_object.Warehouse;

import static org.mockito.Mockito.when;

public class OrdersLogicTest {

@Mock

Warehouse curWarehouse;

@InjectMocks

OrdersLogic ordersLogic;

@Before

public void setUp() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testInitWarehouses() throws Exception {

when(curWarehouse.getName()).thenReturn("getNameResponse");

ordersLogic.initWarehouses();

}

@Test

public void testInitProducts() throws Exception {

ordersLogic.initProducts();

}

@Test

public void testSearch() throws Exception {

when(curWarehouse.getId()).thenReturn(0);

ordersLogic.search();

}

}

package trade\_company.logic.supplier.tabs;

import org.junit.Before;

import org.junit.Test;

import org.mockito.InjectMocks;

import org.mockito.MockitoAnnotations;

public class RequestsControllerLogicTest {

@InjectMocks

RequestsControllerLogic requestsControllerLogic;

@Before

public void setUp() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testInitWarehouses() throws Exception {

requestsControllerLogic.initWarehouses();

}

}