МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ПЕРМСКОГО КРАЯ

государственное бюджетное профессиональное образовательное учреждение   
**«Пермский химико-технологический техникум»**

**(ГБПОУ «ПХТТ»)**

|  |  |
| --- | --- |
| Рекомендовано к защите  Председатель ПЦК ИТ и программирования  \_\_\_\_\_\_\_\_\_\_\_\_(Е.А.Соковнина)  Протокол №\_\_\_от «\_\_\_»\_\_\_\_\_\_\_\_\_\_2023 г. | Допущено к защите  заместитель директора  \_\_\_\_\_\_\_\_\_\_\_\_(О.В.Князева)  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023 г.  Специальность 09.02.07 Информационные системы и программирование |

**ДИПЛОМНЫЙ ПРОЕКТ**

|  |  |
| --- | --- |
| **Тема:** | **Разработка приложения для учета размещения материальных** |
|  | **ценностей в помещениях Пермского государственного** |
|  | **института культуры** |

Обучающийся группы ИС-20-11 (И.А.Большаков)

Руководитель (Л.Ф.Абдалова)

2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ПЕРМСКОГО КРАЯ

государственное бюджетное профессиональное образовательное учреждение   
«Пермский химико-технологический техникум»

(ГБПОУ «ПХТТ»)

**«УТВЕРЖДАЮ»**

Заместитель директора

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /О.В. Князева/

**ЗАДАНИЕ**

**на дипломный проект (работу)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Обучающийся(аяся) | | | | Большаков Игорь Алексеевич | | |
| Группа | | ИС-20-11 | | | | |
| Специальность | | | | | | 09.02.07 «Информационные системы и |
| программирование» | | | | | | |
| Тема задания | | | | | Разработка приложения для учета размещения | |
| Материальных ценностей в помещениях Пермского государственного | | | | | | |
| Института культуры | | | | | | |
| Форма ГИА | Дипломный проект | | | | | |
|  | | |  | | | |
| Исходные данные | | | Техническое задание | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |

Структура выпускной квалификационной работы

|  |  |  |
| --- | --- | --- |
|  | *Объем выполне-*  *ния в % от всего задания* | *Срок выполнения* |
| Введение  Актуальность, новизна и практическая значимость выбранной темы, ее связь с задачами конкретного предприятия.  Цель и задачи дипломного проекта  Необходимость выполнения работы по выбранной теме. | 3 | 6 - 9 апреля |
| 1.Теоретическая часть  Анализ предметной области.  Требования к программному продукту.  Сравнительный анализ существующих программных продуктов. | 30 | 10– 24 апреля |
| 2. Практическая часть  Постановка задачи.  Выбор средств и технологии создания программного продукта.  Моделирование предметной области.  Алгоритм функционирования программного продукта.  Описание работы программного продукта.  Перспективы развития программного продукта. | 60 | 25 апреля – 27 мая |
| Заключение  Краткие выводы, оценка результатов выполнен­ной работы, преимущества решений.  Предложения по использованию разработанного программного продукта, а также научная, социальная и экономическая ценность работы. | 2 | 28-31 мая |
| Приложения  Текст программы.  Руководство пользователя.  Руководство программиста. | 5 | 1- 7 июня |
| Нормоконтроль |  | 8-11 июня |
| Получение отзыва на выпускную квалификационную работу |  | 11 – 14 июня |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Руководитель работы*** | | |  | | | | | ***/*** | ***Л.Ф.Абдалова*** | | | ***/*** | |
|  | |  | | | | | |  |  | | |  | |
| Рассмотрено на заседании ПЦК | | | | | ***ИТ и программирования*** | | | | | | | |
| Протокол № | ***8*** | от « | | ***06*** | | » | ***апреля*** | | | 20 | ***23*** | г. | |
| Председатель ПЦК | |  | | | | | | ***/*** | ***Е.А.Соковнина*** | | | ***/*** | |
|  |  |  | |  | |  |  | | |  |  |  | |
| Дата выдачи задания | | | « | ***6*** | | » | ***апреля*** | | | 20 | ***23*** | г. | |
| Срок выполнения задания « | | | | ***14*** | | » | ***июня*** | | | 20 | ***23*** | г. | |
|  | | | |  | |  |  | | |  |  |  | |
| Задание получил | |  | | | | | | / | ***И.А.Большаков*** | | | / | |

Оглавление

[Введение 4](#_Toc138014545)

[1. Теоретическая часть 5](#_Toc138014546)

[1.1. Анализ предметной области 5](#_Toc138014547)

[1.2. Требования к программному продукту 6](#_Toc138014548)

[1.3. Сравнительный анализ существующих программных продуктов 7](#_Toc138014549)

[2. Практическая часть 8](#_Toc138014550)

[2.1. Постановка задачи 8](#_Toc138014551)

[2.2. Выбор средств и технологии создания программного продукта 9](#_Toc138014552)

[2.3. Моделирование предметной области 10](#_Toc138014553)

[2.4. Алгоритм функционирования программного продукта 11](#_Toc138014554)

[2.5. Описание работы программного продукта 12](#_Toc138014555)

[2.6. Перспективы развития программного продукта 13](#_Toc138014556)

# ВВЕДЕНИЕ

Материальные ценности – это все предметы, которые обладают какой-либо стоимостью. Под материальными ценностями в ПГИК подразумевается такое имущество института, как мебель и оборудование.

Институт использует большое количество материальных ценностей, включая различное имущество для научных и учебных целей. Однако, в Пермском государственном институте культуры учет материальных ценностей осуществляется вручную на бумаге, что часто приводит к ошибкам и неэффективному использованию ресурсов.

Система позволит институту упростить учет материальных ценностей и повысить эффективность использования имущества. Благодаря будущей системе, институт сможет быстро и легко определить местоположение имущества, а также следить за его перемещением.

Кроме того, система позволит отслеживать расположение имущества в режиме реального времени, что сократит время на его поиск и позволит быстрее реагировать на проблемы с материальными ценностями.

Целью дипломного проекта является разработка программного продукта, который позволит упростить учет размещения материальных ценностей, повысить эффективность использования имущества и улучшить мониторинг перемещения имущества в институте. Кроме того, целью является обеспечение удобного и быстрого доступа к информации о размещенных материальных ценностях и их местоположении в каждой аудитории, что позволит сократить время на поиск и улучшить управление ресурсами института.

Для реализации цели дипломного проекта необходимо выполнить следующие задачи:

1. Проанализировать предметную область;

2. Спроектировать базу данных;

3. Разработать пользовательский интерфейс приложения, позволяющий быстро и удобно получать информацию о местоположении имущества в помещениях;

4. Разработать приложение для управления перемещением материальных ценностей в институте;

5. Разработать техническую документацию по использованию приложения, чтобы обеспечить его эффективное использование в институтах.

# Теоретическая часть

## Анализ предметной области

Пермский государственный институт культуры — это высшее учебное заведение культурного профиля в г. Перми. В нём находится множество материальных ценностей, предназначенных для обучения и научных исследований.

Материальная ценность в контексте имущества — это уникальная идентифицирующая метка, присваиваемая каждому отдельному экземпляру имущества. Это может быть, например: мебель, техника, костюмы, реквизит и т.д. Знание расположения каждого имущества в помещениях корпусов очень важно для института, так как это позволяет быстро находить и контролировать имущество института. Отсутствие такого контроля может привести к потере имущества или его неэффективному использованию. Потеря имущества ведет к убыткам института. Кроме того, существует необходимость поиска определенного имущества, которое может занимать много времени и замедлять рабочий процесс.

У материальных ценностей есть такие характеристики как: наименование, описание, цена, инвентарный номер.

В институте материальные ценности хранятся в различных помещениях. У института есть различное количество корпусов. Помещения в институте могут присваиваться к разным типам такие как: аудитория, склад, коридор.

Информационной системой смогут пользоваться различные пользователи с разными должностями, а конкретно: кладовщик, менеджер и администратор.

Кладовщику фиксирует размещение имущества на бумажном носителе. После чего кладовщик передает эту информацию менеджеру, который в свою очередь размещает имущество согласно бумажному документу.

На основе проведенного анализа были выделены основные требования к разрабатываемому приложению. Для успешной реализации проекта необходимо учитывать эти требования и разработать систему, которая обеспечит эффективный учет и управление размещением материальных ценностей в помещениях института.

## Требования к программному продукту

Был выделен следующий список требований для учета размещения материальных ценностей в помещениях института:

1. Удобный интерфейс пользователя: приложение должно иметь интуитивно понятный интерфейс для пользователя, который позволит ему быстро и легко освоиться с программой и выполнять все необходимые действия.
2. Учет материальных ценностей: приложение должно предоставлять возможность вести учет всех материальных ценностей, имеющихся в институте. Для каждой материальной ценности должна быть возможность указать ее название, серийный номер, стоимость, дату приобретения и другие характеристики.
3. Учет перемещения материальных ценностей: приложение должно отслеживать перемещение имущества между помещениями и обновлять информацию об его местоположении в базе данных.
4. Генерация отчетов: приложение должно иметь возможность генерировать различные отчеты на основе данных, связанных с учетом материальных ценностей, такие как список имущества, перемещения имущества.
5. Безопасность данных: приложение должно обеспечивать безопасность хранения и обработки данных, а также защиту от несанкционированного доступа.
6. Поддержка нескольких пользователей: приложение должно иметь возможность поддерживать несколько пользователей, с различными уровнями доступа к данным в системе.

Функциональная диаграмма (Рис. 1)

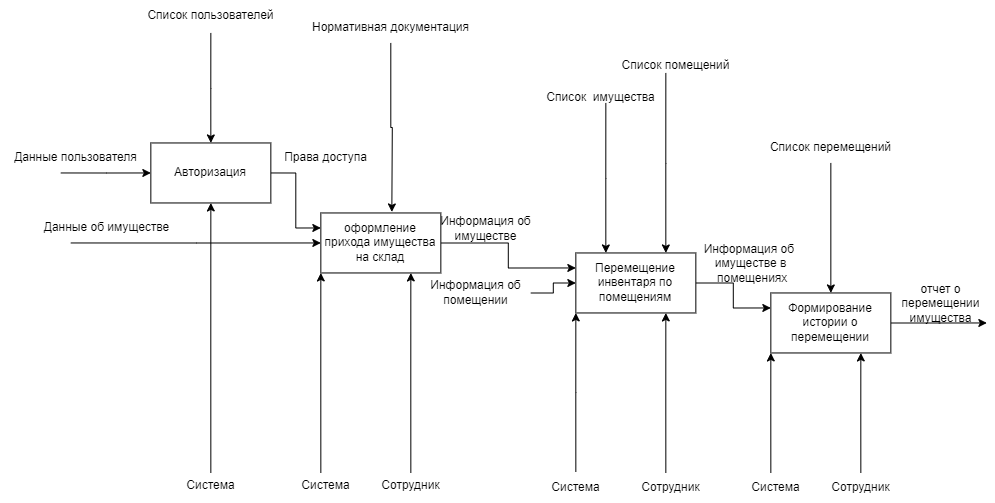


Рис.

## Сравнительный анализ существующих программных продуктов

Существует множество программ для учета материальных ценностей, функционал которых схож с функциями, которые должны быть реализованы в разрабатываемом приложении. Рассмотрим несколько из них:

1. Asset Panda - это облачный сервис учета материальных ценностей, который позволяет отслеживать перемещения оборудования и хранить историю его эксплуатации. Asset Panda имеет широкий функционал, включая возможность интеграции с другими системами и создание пользовательских отчетов. Однако, использование данного сервиса может быть сопряжено с дополнительными затратами на подписку.
2. Snipe-IT - это бесплатная система учета материальных ценностей с открытым исходным кодом. Snipe-IT предоставляет возможность вести учет перемещений оборудования и отслеживать его историю эксплуатации, а также генерировать отчеты на основе данных. Однако, у данного продукта отсутствует широкий функционал и возможность интеграции с другими системами.
3. Еxcel - электронные таблицы Excel могут использоваться для учета материальных ценностей в институте. Они позволяют вести учет оборудования и перемещений, а также генерировать отчеты на основе данных. Однако, использование Excel может быть неэффективным в случае большого количества материальных ценностей, а также недостаточно безопасным для хранения конфиденциальной информации.

В сравнении с указанными программными продуктами, разрабатываемое приложение для учета материальных ценностей в институте будет иметь более широкий функционал и возможность интеграции с другими системами, а также обеспечивать безопасность хранения и обработки данных. Приложение будет разработано на базе C# Windows Forms, Microsoft SQL server, что обеспечит его стабильную работу и высокую производительность.

# Практическая часть

## Постановка задачи

Задача дипломного проекта заключается в разработке приложения для учета размещения материальных ценностей в помещениях института. Для этого необходимо разработать следующий функционал системы:

* Хранение информации об имуществе, его технических характеристиках и местонахождении в аудиториях института;
* Возможность добавления, редактирования и удаления информации об имуществе и его местонахождении в базе данных;
* Возможность просмотра расположения имущества в аудиториях, а также осуществлять поиск имущества по его названию или другим характеристикам;
* Возможность перемещения имущества между аудиториями, с возможностью отслеживания изменений местоположения имущества в базе данных;
* Возможность формирования отчётов по перемещаемому имуществу.

Приложение разрабатывается для сотрудников института. В системе должно быть предусмотрено разграничение прав доступа, в связи с этим можно выделить несколько ролей: администратор, менеджер и кладовщик.

Для каждой роли пользователя должен быть разработан свой функционал.

Необходимо реализовать механизм авторизации. При авторизации пользователь вводит выданный администратором логин и пароль. Данная информация будет хранится в базе данных.

Менеджер после авторизации должен видеть список материальных ценностей, перемещать имущество, так же должен иметь возможность формирования отчётности по имуществу, а именно, где находится имущество в текущий момент и смотреть историю его перемещения по помещениям. При перемещении имущества менеджер должен выбрать имущество, которое он хочет переместить, выбирать корпус и помещение. При формировании отчета менеджер должен увидеть следующую информацию: дату перемещения, ФИО менеджера, инвентарный номер имущества, наименование имущества, от куда и куда оно перемещалось, и в каком корпусе это перемещение осуществлялось.

Кладовщик после авторизации в системе сможет просматривать полный список имущества и осуществлять приход новой единицы имущества, указав при этом его инвентарный номер, наименование, описание, цену, корпус, склад(помещение). При просмотре списка имущества он должен видеть следующую информацию о нём: инвентарный номер, наименование и описание имущества.

У администратора после авторизации должны быть доступны те же функции, какие и у менеджера и кладовщика, но также он должен обладать следующими функциями: добавление и редактирование информации о сотруднике, добавление и редактирование информации о помещениях института и типов этих помещений, добавление и редактирование адреса корпусов. При добавлении нового корпуса, администратору необходимо указать адрес. При добавлении нового сотрудника администратор должен заполнять следующие данные: логин, фамилию, имя, отчество, пароль и должность сотрудника. При добавлении помещения администратор должен заполнять наименование этого помещения, его этаж, тип и в каком корпусе будет находится данное помещение. Если нужного типа помещения нет, администратор сможет добавить новый тип, указав его наименование.

Так же необходимо разработать понятный и удобный пользовательский интерфейс программы.

Диаграмма прецедентов (Рис. 2)

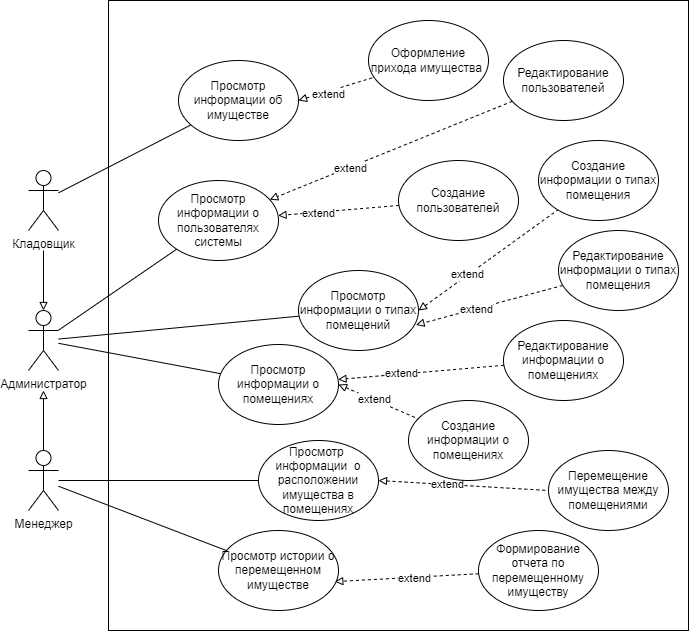


Рис.

## Выбор средств и технологии создания программного продукта

Для разработки систем для учета материальных ценностей в институте были выбраны следующие средства:

1. Среда разработки – Microsoft Visual Studio

* Microsoft Visual Studio – это интегрированная среда разработки;
* Microsoft Visual Studio позволяет разрабатывать приложение с графическим интерфейсом.

1. Язык программирования: C#

* C# является объектно-ориентированным языком программирования, который обладает широкими возможностями для разработки приложений под платформу Windows;
* C# интегрируется с платформой .NET, что обеспечивает удобные средства разработки и богатую функциональность.

Для интерфейса системы будет использоваться Windows Form.

1. Интерфейс пользователя: Windows Forms

* Windows Forms — это технология, которая позволяет создавать графический интерфейс для приложения
* Windows Forms предоставляет удобные инструменты для разработки графического интерфейса пользователя (GUI) под операционную систему Windows;
* Windows Forms позволяет создавать привлекательные и интуитивно понятные пользовательские интерфейсы.

Все вводимые и выводимые данные будут храниться в СУБД: Microsoft SQL Server.

1. СУБД: Microsoft SQL Server

* Microsoft SQL Server является мощной и надежной системой управления базами данных (СУБД), разработанной для работы в среде Windows;
* SQL Server обладает широкими возможностями по управлению данными, обеспечивает высокую производительность и безопасность хранения информации.

Выбор этих средств и технологий обосновывается следующими преимуществами:

* C# является одним из наиболее распространенных языков программирования для разработки под платформу Windows, обладает богатым набором библиотек и фреймворков, что упрощает разработку и обеспечивает большую гибкость;
* Windows Forms предоставляет простой и эффективный способ создания графического интерфейса пользователя, позволяя быстро разрабатывать удобные и интуитивно понятные приложения;
* Microsoft SQL Server является одной из наиболее популярных и широко используемых СУБД, обладает высокой производительностью, масштабируемостью и надежностью.

Выбор данных средств и технологий позволит эффективно реализовать требуемый функционал приложения, обеспечить удобный интерфейс пользователя и надежное хранение информации для учета размещения материальных ценностей в институте.

## Моделирование предметной области

Концептуальная модель (Рис. 3) используется для проектирования базы данных на самом раннем этапе разработки. ER-диаграмма, созданная с помощью этого подхода, описывает сущности и связи между ними в общих чертах, без учета конкретных атрибутов. Цель концептуальной ER-диаграммы – определить реальные объекты, с которыми будет работать приложение, и описать их связи между собой.

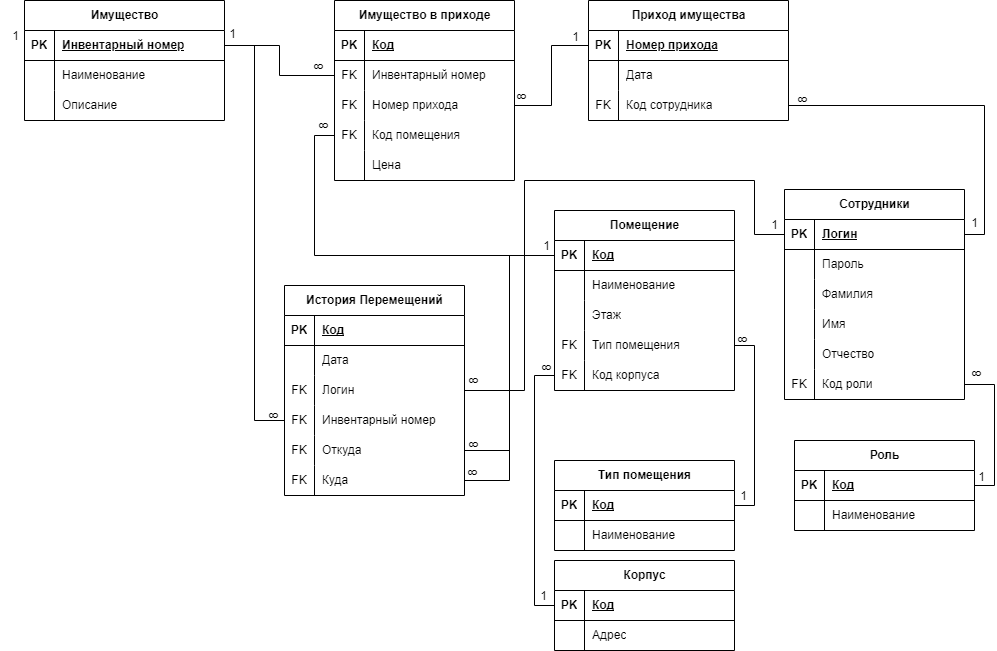


Рис.

Логическая ER-диаграмма (Рис. 4) создается после того, как концептуальная диаграмма была разработана. Её цель — определить атрибуты каждой сущности и определить связи между ними. Логическая ER-диаграмма, таким образом, представляет собой расширенную версию концептуальной диаграммы и включает в себя все необходимые подробности, необходимые разработчикам баз данных для создания приложения (типы данных).

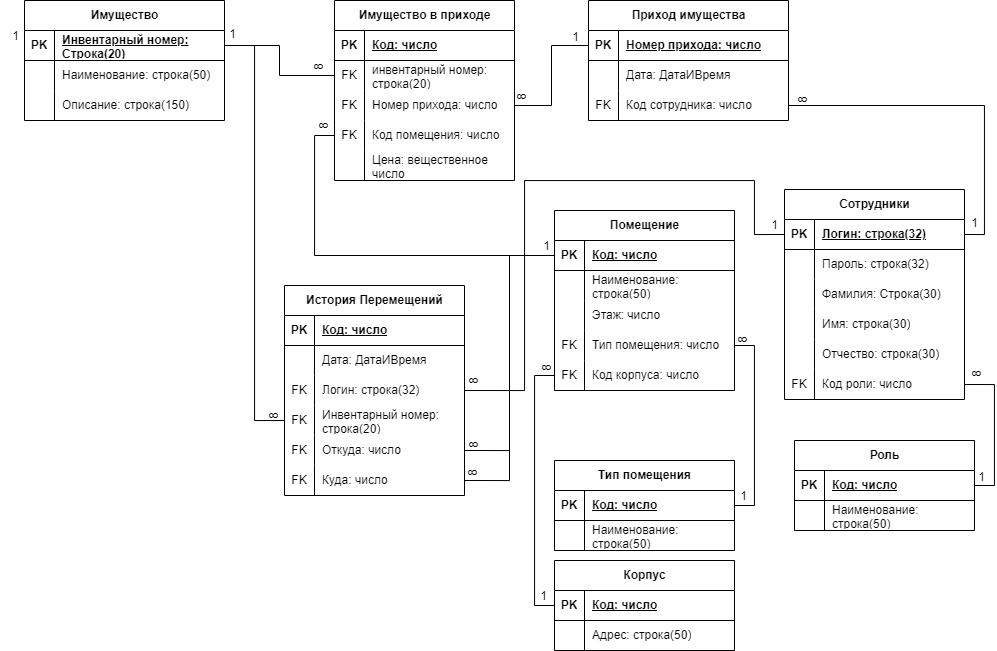


Рис.

Физическая ER-диаграмма (Рис. 5) используется для создания фактической базы данных в выбранной СУБД. Она описывает все объекты из логической диаграммы и включает особенности физической реализации базы данных на конкретной платформе.

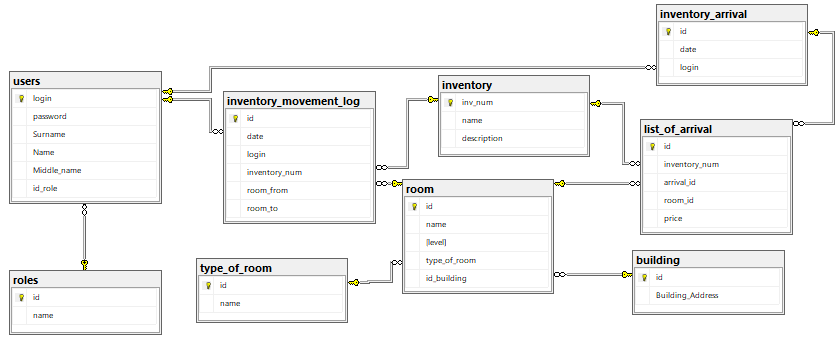


Рис.

## Алгоритм функционирования программного продукта

1. Инициализация программы:

* Запуск приложения и загрузка необходимых компонентов;
* Подключение к базе данных;
* Авторизация.

1. Отображение списка имущества:

* Загрузка информации об имуществе из базы данных;
* Отображение списка имущества в интерфейсе пользователя.

1. Добавление и редактирование информации об имуществе:

* Пользователь выбирает опцию добавления или редактирования информации об имуществе;
* Вводит необходимые данные, такие как наименование, характеристики, количество и местонахождение имущества;
* Происходит сохранение информации в базе данных.

1. Просмотр расположения имущества в аудиториях:

* Пользователь выбирает опцию просмотра расположения имущества;
* Приложение отображает список аудиторий и соответствующее размещение имущества в каждой аудитории;
* Пользователь может выбрать конкретное помещение для подробного просмотра имущества в нем.

1. Перемещение имущества:

* Пользователь выбирает опцию просмотра имущества в помещениях;
* Выбирает имущество (двойным щелчком по нему), которое необходимо переместить, и указывает новое помещение;
* Происходит обновление информации об имуществе в базе данных, отражающее его новое местоположение.

1. Завершение работы:

* Пользователь заканчивает работу с приложением;
* Приложение закрывается.

Этот алгоритм обеспечивает основные функции приложения, включающие в себя добавление, редактирование, просмотр и перемещение имущества.

Для более лучшего понимания была разработана диаграмма деятельности (Рис. 6).

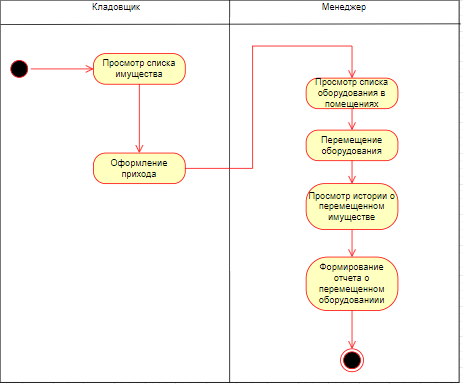
****

Рис.

## Описание работы программного продукта

При запуске приложения сотрудник института попадает на форму авторизации (Рис. 7) с доступными для ввода текстовыми полями «Логин» и «Пароль»

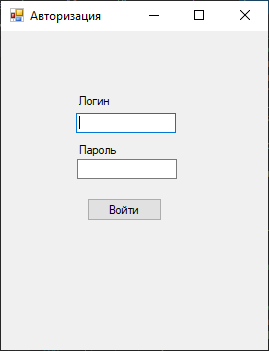


Рис.

Для входа в систему следует использовать данные, выданные администратором/директором института.

Имея данные для авторизации их следует ввести и нажать на кнопку «Войти».

После входа в систему пользователю откроется весь доступный ему функционал (Рис. 8).

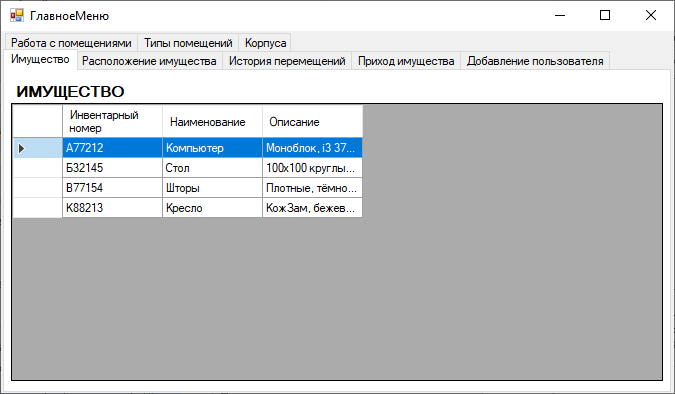


Рис.

На вкладке «Имущество» пользователь может посмотреть полный перечень имущества (Рис. 8).

На вкладке «Расположение имущества» пользователь сможет просматривать, где сейчас находится имущество (Рис. 9).

Также есть функционал для поиска имущества по фильтру.

На форме есть 4 заполняемых поля, которые автономно применяются к фильтру. Присутствует кнопка сброса фильтра которая очищает все поля и восстанавливает исходный список имущества.

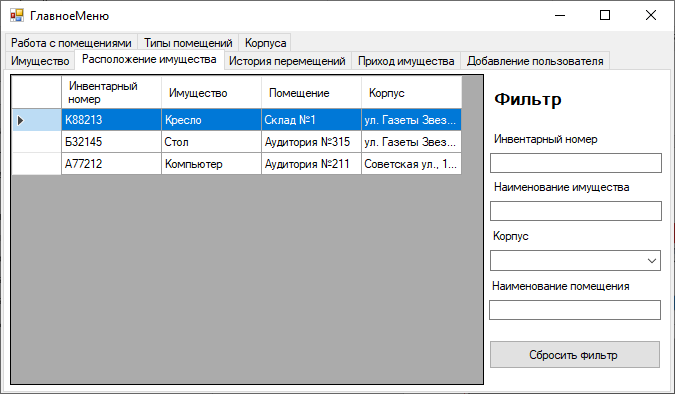


Рис.

По двойному нажатию на какое-либо имущество на этой форме открывается форма «Перемещение» для перемещения выбранного имущества (Рис. 10).

На форме присутствуют поля недоступные для редактирования пользователю в которых указана вся информация по выбранному имуществу.

В правой части формы находится два выпадающих списка в которых необходимо выбрать место назначения перемещаемого имущества.

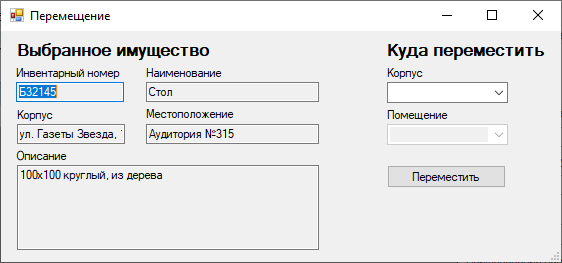


Рис.

На вкладке «История перемещений» сможет просматривать и фильтровать историю перемещения оборудования после чего сформировать отчет в формате excel (Рис. 11).

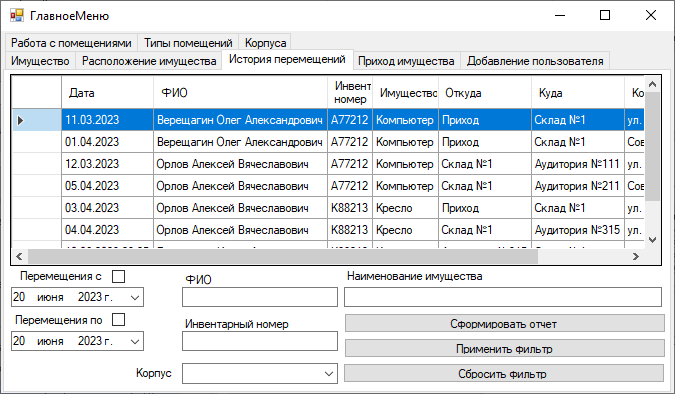


Рис.

На вкладке «Приход имущества» пользователь сможет добавлять новое имущество в систему (Рис. 12). Чтобы оформить приход пользователю необходимо заполнить все поля и добавить имущество в список прихода хотя бы одно имущество, после чего можно будет оформить приход. На форме можно отредактировать имущество из списка. Нажав на имущество в списке поля автоматически заполняется его данными, после чего можно будет отредактировать эти поля и нажать на кнопку «Изменить в приходе»

Так же можно удалить имущество из списка.

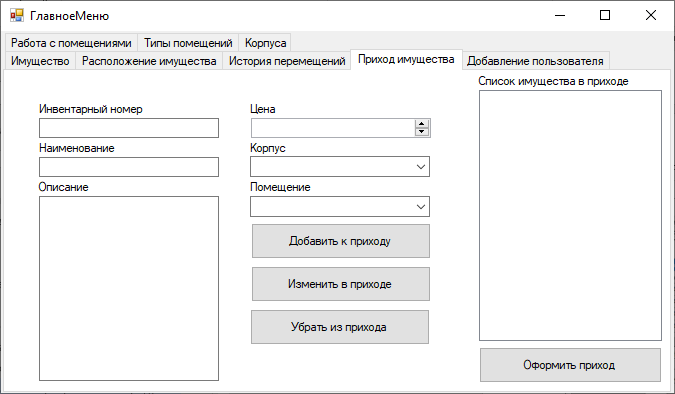


Рис.

На вкладке «Добавление пользователя» пользователь может добавлять новых сотрудников (Рис. 13). Выбрав сотрудника из списка данные под грузятся в поля. Редактирование пользователя происходит по логину сотрудника.

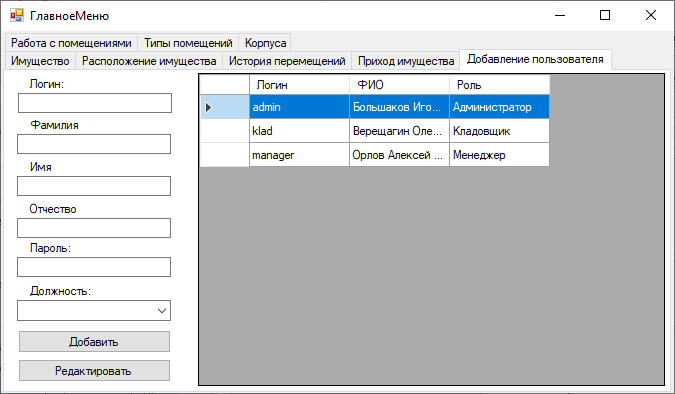


Рис.

На вкладке «Работа с помещениями» пользователь сможет посмотреть список всех помещений в институте (Рис. 14). По кнопке добавить пользователь сможет добавить новое помещение в базу предварительно заполнив поля.

Присутствует возможность редактирования помещений путем выбора необходимого помещения из списка, редактирования полей и нажатием на кнопку «Изменить».

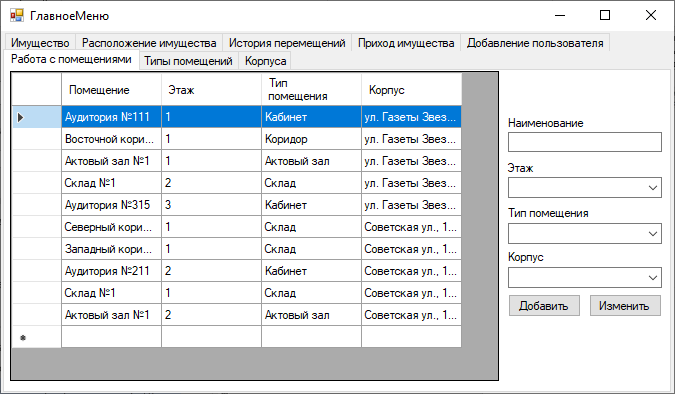


Рис.

На вкладке «Типы помещений» пользователь может просматривать перечень типов помещений в базе данных, а также добавлять новые или изменять старые типы (Рис. 15).

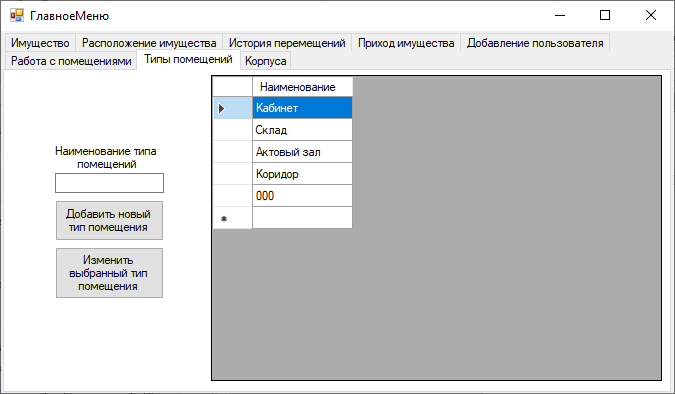


Рис.

На вкладке «Корпуса» пользователь может просматривать перечень адресов института, а так же добавлять и редактировать информацию корпусов (Рис. 16).

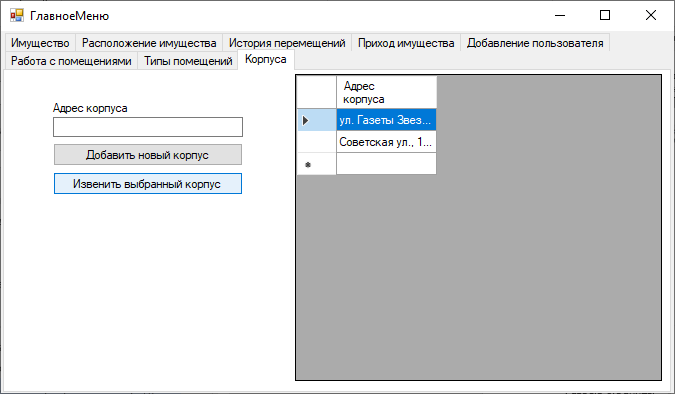


Рис.

## Перспективы развития программного продукта

Программный продукт имеет перспективы развития и расширения функциональности. Некоторые из потенциальных направлений развития включают:

1. Интеграция с другими информационными системами: Продукт может быть доработан для интеграции с другими системами института, такими как система управления учебными планами или система учета кадров. Это позволит обмениваться данными и повысит эффективность работы института в целом;
2. Расширение функциональности отчетности: Добавление возможности генерации различных отчетов и аналитических данных, например, по статистике использования имущества, списанию или обновлению технических характеристик. Это поможет принимать информированные решения по управлению имущества;
3. Мобильное приложение: Разработка мобильного приложения позволит операторам получать доступ к информации о материальных ценностях и управлять ими непосредственно с мобильных устройств, что повысит гибкость и доступность системы;
4. Расширение функциональности интерфейса пользователя: Добавление дополнительных функций и возможностей для удобства работы операторов. Например, создание функции сканирования штрих-кодов для быстрого и точного добавления или перемещения имущества;
5. Реализация системы оповещений и уведомлений: Добавление функционала оповещений и уведомлений, который будет предупреждать операторов об истечении гарантийного срока или необходимости технического обслуживания имущества;
6. Облачная интеграция: Подключение программного продукта к облачным ресурсам для обеспечения более гибкого доступа к данным, улучшения безопасности и совместной работы между различными институтскими подразделениями.

Эти перспективы развития помогут улучшить функциональность и эффективность программного продукта и обеспечат его адаптацию к меняющимся потребностям и требованиям институтской среды.

# ЗАКЛЮЧЕНИЕ

В ходе дипломного проекта было разработано приложение для учета размещения материальных ценностей в помещениях Пермского государственного института культуры

Была проанализирована предметная область

Спроектирована База данных

Был разработан удобный пользовательский интерфейс приложения

Была разработана техническая документация по использованию приложения

# СПИСОК ЛИТЕРАТУРЫ

1. https://metanit.com/;
2. https://stackoverflow.com/;
3. https://habr.com/ru/;

# ПРИЛОЖЕНИЕ 1

**Класс db.cs**

internal class db

{

private static string dbname = @"PC306\_15\SQLEXPRESS";

public static SqlConnection conn = new SqlConnection($@"Data Source={dbname};Initial Catalog=PGIC;Integrated Security=True");

}

**Класс token.cs**

internal class token

{

public static string login;

public static int id\_role;

}

**Форма авторизации**

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace Diploma

{

public partial class Авторизация : Form

{

public Авторизация()

{

InitializeComponent();

}

private string getLogin()

{

token.login = textBox1.Text;

return token.login;

}

private string getPassword()

{

return textBox2.Text;

}

private void button1\_Click(object sender, EventArgs e)

{

string login = getLogin();

string password = getPassword();

db.conn.Open();

SqlDataReader dataReader = new SqlCommand($"SELECT login,[password],[id\_role] FROM Users WHERE Users.login = '{login}'", db.conn).ExecuteReader();

try

{

dataReader.Read();

// Проверка пороля

if (!(dataReader.GetString(1).Equals(password)))

{

dataReader.Close();

throw new Exception("wrong pass");

}

token.id\_role = Convert.ToInt32(dataReader.GetValue(2));

db.conn.Close();

this.Hide();

new ГлавноеМеню().ShowDialog();

this.Show();

}

catch (Exception)

{

MessageBox.Show("Данные введены неверно!");

db.conn.Close();

}

}

}

}

**Форма ГлавноеМеню**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Diagnostics;

using System.Linq;

using System.Windows.Forms;

namespace Diploma

{

public partial class ГлавноеМеню : Form

{

public ГлавноеМеню()

{

InitializeComponent();

applySystemRole();

foreach (TabPage tp in tabControl1.TabPages)

{

foreach (DataGridView dgv in tp.Controls.OfType<DataGridView>())

{

dgv.ReadOnly = true;

dgv.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

}

};

}

private void applySystemRole()

{

if (token.id\_role != 1)

foreach (TabPage tp in tabControl1.TabPages)

{

tp.Parent = null;

};

if (token.id\_role == 2)

{

tab\_inventory\_placement.Parent = tabControl1;

tab\_log\_movement.Parent = tabControl1;

}

if (token.id\_role == 3)

{

tab\_inventory\_arrival.Parent = tabControl1;

tab\_inventory.Parent = tabControl1;

}

}

#region update\_cb

private void update\_CB\_building(ComboBox CB)

{

db.conn.Open();

DataSet ds = new DataSet();

try

{

new SqlDataAdapter(

$@"SELECT ID, [Building\_Address] from [dbo].[building]",

db.conn).Fill(ds);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "\n" + new StackFrame(0, true).GetFileLineNumber());

}

db.conn.Close();

CB.DisplayMember = "Building\_Address";

CB.ValueMember = "ID";

CB.DataSource = ds.Tables[0];

CB.SelectedIndex = -1;

}

#endregion

#region Inventory

private void tab\_inventory\_Enter(object sender, EventArgs e)

{

update\_datagrid\_inventory();

}

private void update\_datagrid\_inventory()

{

db.conn.Open();

DataSet ds = new DataSet();

new SqlDataAdapter(

$@"SELECT [inv\_num] as [Инвентарный номер],[name] as [Наименование],[description] as [Описание] from inventory",

db.conn).Fill(ds);

dataGrid\_inventory.DataSource = ds.Tables[0];

db.conn.Close();

}

#endregion

#region log\_of\_movement

private void tab\_log\_movement\_Enter(object sender, EventArgs e)

{

update\_datagrid\_log(string.Empty);

update\_CB\_building(comboBox\_logMovement\_building);

comboBox\_logMovement\_building.SelectedIndex = -1;

}

private void update\_datagrid\_log(string where)

{

db.conn.Open();

try

{

DataSet ds = new DataSet();

new SqlDataAdapter(

$@"SELECT iml.[date] as [Дата],CONCAT(u.[Surname],' ',u.[Name],' ',u.[Middle\_name]) as [ФИО],i.[inv\_num] as [Инвентарный номер], i.name as [Имущество],

IIF(r1.id is null, 'Приход', r1.name) as [Откуда],

r2.name as [Куда], b.[Building\_Address] as [Корпус]

FROM [dbo].[inventory\_movement\_log] as iml

JOIN inventory as i ON i.inv\_num = iml.inventory\_num

LEFT JOIN room as r1 ON r1.ID = iml.room\_from

JOIN room as r2 ON r2.ID = iml.room\_to

JOIN users as u ON u.login = iml.login

JOIN building as b ON b.id = r2.id\_building

{where}",

db.conn).Fill(ds);

dataGridView\_log.DataSource = ds.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show($"{ex.Message}");

}

db.conn.Close();

}

private void button\_log\_apply\_filter\_Click(object sender, EventArgs e)

{

string inventoryQuerry = textbox\_log\_inventory.Text;

string inventoryNumQuerry = "";

if (textbox\_log\_inventoryNUM.Text != string.Empty)

inventoryNumQuerry = $@"AND i.inv\_num LIKE '%{textbox\_log\_inventoryNUM.Text}%' ";

string loginQuerry = $"AND CONCAT(u.[Surname],' ',u.[Name],' ',u.[Middle\_name]) LIKE '%{textbox\_log\_fio.Text}%'";

string time\_since = "";

string time\_until = $"AND '{DateTime.Now}' > iml.[date]";

if (checkbox\_log\_since.Checked)

time\_since = $"AND '{dateTimePicker\_log\_since.Value}' < iml.[date]";

if (checkbox\_log\_until.Checked)

time\_until = $"AND '{dateTimePicker\_log\_until.Value}' > iml.[date]";

string buildingQuerry = "";

if (!(comboBox\_logMovement\_building.SelectedIndex < 0))

buildingQuerry = $"AND b.id = {comboBox\_logMovement\_building.SelectedValue}";

string FilterQuery = $" WHERE i.name LIKE '%{inventoryQuerry}%' {inventoryNumQuerry} {loginQuerry} {time\_since} {time\_until} {buildingQuerry}";

update\_datagrid\_log(FilterQuery);

}

private void button\_log\_create\_statement\_Click(object sender, EventArgs e)

{

create\_excel();

}

private void create\_excel()

{

// Создаем новый экземпляр класса Excel

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

// Создаем новую книгу Excel

Microsoft.Office.Interop.Excel.Workbook workbook = excel.Workbooks.Add(Type.Missing);

// Создаем новый лист Excel

Microsoft.Office.Interop.Excel.Worksheet worksheet = null;

worksheet = workbook.Sheets["Лист1"];

worksheet = workbook.ActiveSheet;

// Объединяем ячейки первой строки и указываем надпись "Вывод данных найденного оборудования"

worksheet.Range[worksheet.Cells[1, 1], worksheet.Cells[1, dataGridView\_log.Columns.Count + 1]].Merge();

worksheet.Cells[1, 1].Value = "Отчёт по перемещаемому имуществу";

worksheet.Cells[1, 1].HorizontalAlignment = Microsoft.Office.Interop.Excel.XlHAlign.xlHAlignCenter;

worksheet.Cells[1, 1].Interior.Color = System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.Green);

worksheet.Cells[1, 1].WrapText = true;

// Объединяем ячейки второй строки столбцов A, B и C и указываем надпись "Дата вывода: "

worksheet.Range[worksheet.Cells[2, 1], worksheet.Cells[2, 3]].Merge();

worksheet.Range[worksheet.Cells[2, 4], worksheet.Cells[2, 8]].Merge();

worksheet.Range[worksheet.Cells[3, 7], worksheet.Cells[3, 8]].Merge();

worksheet.Cells[2, 4].Interior.Color = System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.LightBlue);

worksheet.Cells[2, 1].Value = "Дата формирования отчета: ";

worksheet.Cells[2, 1].HorizontalAlignment = Microsoft.Office.Interop.Excel.XlHAlign.xlHAlignCenter;

worksheet.Cells[2, 1].Interior.Color = System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.LightBlue);

worksheet.Cells[2, 4].Value = DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss");

// Выводим названия столбцов DataGridView в строке с надписью "Расположение соответствует базе данных"

for (int i = 1; i <= dataGridView\_log.Columns.Count; i++)

{

worksheet.Cells[3, i].Interior.Color = System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.LightBlue);

worksheet.Cells[3, i] = dataGridView\_log.Columns[i - 1].HeaderText;

worksheet.Cells[3, i].WrapText = true;

}

// Выводим данные DataGridView, начиная с клетки B4

int row = 4;

for (int i = 0; i < dataGridView\_log.Rows.Count; i++)

{

for (int j = 0; j < dataGridView\_log.Columns.Count; j++)

{

// Выводим данные в столбец B

worksheet.Cells[row, j + 1] = dataGridView\_log.Rows[i].Cells[j].Value.ToString();

}

row++;

}

// Добавляем тонкие границы во все клетки таблицы

Microsoft.Office.Interop.Excel.Range range = worksheet.UsedRange;

Microsoft.Office.Interop.Excel.Borders borders = range.Borders;

borders.LineStyle = Microsoft.Office.Interop.Excel.XlLineStyle.xlDouble;

borders.Weight = 1d;

borders.Color = System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.Black);

// Сохраняем файл Excel на рабочем столе

string desktopPath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

workbook.SaveAs(desktopPath + "\\Отчёт по перемещаемому оборудованию.xlsx", Microsoft.Office.Interop.Excel.XlFileFormat.xlWorkbookDefault, Type.Missing, Type.Missing, false, false, Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlNoChange, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing);

// Очищаем экземпляр Excel

excel.Quit();

workbook = null;

excel = null;

MessageBox.Show("Файл Excel успешно создан на рабочем столе.", "Уведомление");

}

private void button\_log\_reset\_filter\_Click(object sender, EventArgs e)

{

checkbox\_log\_since.Checked = false;

checkbox\_log\_until.Checked = false;

comboBox\_logMovement\_building.SelectedIndex = -1;

textbox\_log\_fio.Text = string.Empty;

textbox\_log\_inventory.Text = string.Empty;

update\_datagrid\_log(string.Empty);

}

#endregion

#region add/change users

private void tab\_users\_Enter(object sender, EventArgs e)

{

datagrid\_users\_update();

update\_CB\_users\_roles();

}

private void datagrid\_users\_update()

{

try

{

DataSet ds = new DataSet();

new SqlDataAdapter(

$@"SELECT u.login as [Логин], Concat(u.Surname,' ',u.Name,' ',u.Middle\_name) as [ФИО], r.name as [Роль]

FROM users as u

JOIN roles as r ON r.ID = u.id\_role",

db.conn).Fill(ds);

dataGridView\_users.DataSource = ds.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + " CERF");

}

}

private void update\_CB\_users\_roles()

{

db.conn.Open();

DataSet ds = new DataSet();

try

{

new SqlDataAdapter(

$@"SELECT ID,name from [dbo].[roles]",

db.conn).Fill(ds);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "\n" + new StackFrame(0, true).GetFileLineNumber());

}

db.conn.Close();

comboBox\_users\_roles.DisplayMember = "name";

comboBox\_users\_roles.ValueMember = "ID";

comboBox\_users\_roles.DataSource = ds.Tables[0];

comboBox\_users\_roles.SelectedIndex = -1;

}

private bool users\_data\_validation()

{

if (textBox\_users\_login.Text.Length > 32 || textBox\_users\_login.Text.Length < 6)

{

MessageBox.Show("Логин должен содержать от 6 до 32 символов");

return false;

}

if (string.IsNullOrWhiteSpace(textBox\_users\_SurName.Text) ||

string.IsNullOrWhiteSpace(textBox\_users\_Name.Text) ||

string.IsNullOrWhiteSpace(textBox\_users\_MiddleName.Text))

{

MessageBox.Show("Заполните ФИО");

return false;

}

if (textBox\_users\_SurName.Text.Length > 30 ||

textBox\_users\_Name.Text.Length > 30 ||

textBox\_users\_MiddleName.Text.Length > 30)

{

MessageBox.Show("Поля для ФИО не должны превышать 30 символов");

return false;

}

return true;

}

private string users\_data\_validation\_password()

{

string a = $@"[password] = '{textBox\_users\_password.Text}',";

if (string.IsNullOrWhiteSpace(textBox\_users\_password.Text))

a = "";

return a;

}

private void button\_add\_user\_Click(object sender, EventArgs e)

{

if (!users\_data\_validation())

return;

if (string.IsNullOrWhiteSpace(textBox\_users\_password.Text) ||

textBox\_users\_password.Text.Length > 32)

{

MessageBox.Show("Пароль не должен превышать 32 символа");

return;

}

db.conn.Open();

try

{

new SqlDataAdapter(

$@"INSERT INTO users([login],[password],[Surname], [Name], [Middle\_name],[id\_role])

VALUES('{textBox\_users\_login.Text}','{textBox\_users\_password.Text}',

'{textBox\_users\_SurName.Text},'{textBox\_users\_Name}', '{textBox\_users\_MiddleName}',

{comboBox\_users\_roles.SelectedValue})",

db.conn).Fill(new DataSet());

}

catch (Exception)

{

MessageBox.Show("Такой логин уже занят");

return;

}

db.conn.Close();

MessageBox.Show("Пользователь успешно добавлен!");

datagrid\_users\_update();

}

private void button\_change\_user\_Click(object sender, EventArgs e)

{

if (!users\_data\_validation())

return;

db.conn.Open();

try

{

new SqlDataAdapter(

$@"UPDATE users

SET

{users\_data\_validation\_password()}

[Surname] = '{textBox\_users\_SurName.Text}',

[Name] = '{textBox\_users\_Name.Text}',

[Middle\_name] = {textBox\_users\_MiddleName.Text},

[id\_role] = {comboBox\_users\_roles.SelectedValue}

WHERE login = {textBox\_users\_login.Text}",

db.conn).Fill(new DataSet());

}

catch (Exception)

{

MessageBox.Show("Пользователя с таким логином не было найдено в базе данных. \n Изменение не возможно");

return;

}

db.conn.Close();

MessageBox.Show("Данные пользователя успешно изменены!");

datagrid\_users\_update();

}

private void dataGridView\_users\_RowEnter(object sender, DataGridViewCellEventArgs e)

{

db.conn.Open();

try

{

string login = dataGridView\_users.SelectedRows[0].Cells[0].Value.ToString();

SqlDataReader sdr = new SqlCommand(

$@"SELECT u.[Surname], u.[Name], u.[Middle\_name], u.[id\_role]

FROM users as u

WHERE u.login = '{login}'",

db.conn).ExecuteReader();

sdr.Read();

textBox\_users\_login.Text = login;

textBox\_users\_SurName.Text = sdr.GetString(0);

textBox\_users\_Name.Text = sdr.GetString(1);

textBox\_users\_MiddleName.Text = sdr.GetString(2);

comboBox\_users\_roles.SelectedValue = sdr.GetValue(3);

if ((int)comboBox\_users\_roles.SelectedValue == token.id\_role)

comboBox\_users\_roles.Enabled = false;

else

comboBox\_users\_roles.Enabled = true;

sdr.Close();

}

catch (Exception ex)

{

}

db.conn.Close();

}

#endregion

#region types\_of\_room

private void tab\_type\_of\_room\_Enter(object sender, EventArgs e)

{

update\_datagrid\_types\_of\_room();

}

int id\_type\_of\_room;

private void update\_datagrid\_types\_of\_room()

{

DataSet ds = new DataSet();

db.conn.Open();

new SqlDataAdapter("SELECT id, [name] as [Наименование] from [dbo].[type\_of\_room]", db.conn).Fill(ds);

dataGridView\_typesOfRoom.DataSource = ds.Tables[0];

db.conn.Close();

dataGridView\_typesOfRoom.Columns[0].Visible = false;

}

private void button\_add\_typeOfRoom\_Click(object sender, EventArgs e)

{

new SqlCommand($@"INSERT INTO [dbo].[type\_of\_room](name) VALUES('{textBox\_typeOfRoom\_name.Text}')", db.conn).ExecuteNonQuery();

MessageBox.Show("Новый тип помещения успешно добавлен!");

}

private void button\_change\_typeOfRoom\_Click(object sender, EventArgs e)

{

new SqlCommand($@"UPDATE [dbo].[type\_of\_room]

SET name = '{textBox\_typeOfRoom\_name.Text}'

WHERE id ={id\_type\_of\_room}", db.conn).ExecuteNonQuery();

MessageBox.Show("Тип помещения был успешно изменен!");

}

private void dataGridView\_typesOfRoom\_RowEnter(object sender, DataGridViewCellEventArgs e)

{

try

{

id\_type\_of\_room = (int)dataGridView\_typesOfRoom.SelectedRows[0].Cells[0].Value;

}

catch (Exception)

{

}

}

#endregion

#region buildings

private void tab\_buildings\_Enter(object sender, EventArgs e)

{

update\_datagrid\_buildings();

}

int id\_buildings;

private void update\_datagrid\_buildings()

{

DataSet ds = new DataSet();

db.conn.Open();

new SqlDataAdapter("SELECT [Building\_Address] as [Адрес корпуса] from [dbo].[building]", db.conn).Fill(ds);

dataGridView\_buildings.DataSource = ds.Tables[0];

db.conn.Close();

}

private void button\_add\_building\_Click(object sender, EventArgs e)

{

new SqlCommand($@"INSERT INTO [dbo].[building]([Building\_Address]) VALUES('{textBox\_buildings\_adressName.Text}')", db.conn).ExecuteNonQuery();

MessageBox.Show("Корпус успешно добавлен в базу!");

}

private void button\_change\_building\_Click(object sender, EventArgs e)

{

new SqlCommand($@"UPDATE building

SET Building\_Address = '{textBox\_buildings\_adressName.Text}'

WHERE id = {id\_buildings}", db.conn).ExecuteNonQuery();

MessageBox.Show("Адрес выбранного корпуса был изменен!");

}

private void dataGridView\_buildings\_RowEnter(object sender, DataGridViewCellEventArgs e)

{

try

{

id\_buildings = (int)dataGridView\_buildings.SelectedRows[0].Cells[0].Value;

}

catch (Exception)

{

}

}

#endregion

#region rooms

int id\_selected\_room;

private void update\_datagrid\_rooms()

{

DataSet ds = new DataSet();

db.conn.Open();

new SqlDataAdapter(

$@"SELECT r.id, r.name as [Помещение], r.level as [Этаж], t.name as [Тип помещения], b.Building\_address as [Корпус]

FROM room as r

join [dbo].[type\_of\_room] as t ON t.id = r.[type\_of\_room]

join [dbo].[building] as b ON b.id = r.[id\_building]",

db.conn).Fill(ds);

dataGridView\_rooms.DataSource = ds.Tables[0];

db.conn.Close();

dataGridView\_rooms.Columns[0].Visible = false;

}

private void update\_CB\_type\_room()

{

db.conn.Open();

DataSet ds = new DataSet();

try

{

new SqlDataAdapter(

$@"SELECT ID,name from [dbo].[type\_of\_room]",

db.conn).Fill(ds);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "\n" + new StackFrame(0, true).GetFileLineNumber());

}

db.conn.Close();

comboBox\_room\_typeOfRoom.DisplayMember = "name";

comboBox\_room\_typeOfRoom.ValueMember = "ID";

comboBox\_room\_typeOfRoom.DataSource = ds.Tables[0];

}

private void tab\_room\_Enter(object sender, EventArgs e)

{

update\_CB\_type\_room();

update\_CB\_building(comboBox\_room\_building);

update\_datagrid\_rooms();

comboBox\_room\_building.SelectedIndex = -1;

comboBox\_room\_typeOfRoom.SelectedIndex = -1;

}

private bool room\_validation()

{

return false;

}

private void button\_add\_room\_Click(object sender, EventArgs e)

{

if (!room\_validation())

return;

db.conn.Open();

new SqlCommand(

$"INSERT INTO [dbo].[room]([name],[level],[type\_of\_room],[id\_building])" +

$"VALUES('{textBox\_room\_name}',{comboBox\_room\_level.Text},{comboBox\_room\_typeOfRoom},{comboBox\_room\_building})",

db.conn).ExecuteNonQuery();

db.conn.Close();

MessageBox.Show("Помещение успешно добавлено!");

}

private void dataGridView\_rooms\_RowEnter(object sender, DataGridViewCellEventArgs e)

{

try

{

id\_selected\_room = (int)dataGridView\_rooms.SelectedRows[0].Cells[0].Value;

}

catch (Exception)

{

}

}

private void button\_change\_room\_Click(object sender, EventArgs e)

{

if (!room\_validation())

return;

if (!(dataGridView\_rooms.SelectedRows.Count > 0))

{

MessageBox.Show("Выберите помещение которое хотите изменить!");

return;

}

db.conn.Open();

new SqlCommand(

$@"UPDATE [dbo].[room] SET

[name] = '{textBox\_room\_name}',

[level] = {comboBox\_room\_level.Text},

[type\_of\_room] = {comboBox\_room\_typeOfRoom},

[id\_building] = {comboBox\_room\_building}

WHERE id = {id\_selected\_room}",

db.conn).ExecuteNonQuery();

db.conn.Close();

MessageBox.Show("Помещение успешно изменено!");

}

#endregion

#region inventory\_placement ??

private void update\_datagrid\_inventory\_placement(string additional\_querry)

{

DataSet ds = new DataSet();

db.conn.Open();

new SqlDataAdapter(

$@"SELECT iml.inventory\_num as [Инвентарный номер], i.name as [Имущество], r.name as [Помещение], b.[Building\_Address] as [Корпус]

FROM [dbo].[inventory\_movement\_log] as iml

JOIN inventory as i ON i.inv\_num = iml.inventory\_num

JOIN room as r ON r.id = iml.room\_to

JOIN building as b ON r.id\_building = b.id

WHERE iml.date=(select max(date) from inventory\_movement\_log where iml.inventory\_num = inventory\_num)

{additional\_querry}",

db.conn).Fill(ds);

dataGridView\_inventory\_placement.DataSource = ds.Tables[0];

db.conn.Close();

}

private void create\_additional\_querry\_for\_InventoryPlacement\_Datagrid()

{

string inventoryQuerry = $"";

string inventoryNumQuerry = "";

string roomQuerry = "";

string buildingQuerry = "";

if (textBox\_inventoryPlacement\_invNUM.Text != string.Empty)

inventoryNumQuerry = $@"AND i.inv\_num LIKE '%{textBox\_inventoryPlacement\_invNUM.Text}%' ";

if (!(comboBox\_inventoryPlacement\_building.SelectedIndex < 0))

buildingQuerry = $"AND b.id = {comboBox\_inventoryPlacement\_building.SelectedValue} ";

if (textBox\_inventoryPlacement\_inventoryName.Text != string.Empty)

inventoryQuerry = $"AND i.name LIKE '%{textBox\_inventoryPlacement\_inventoryName.Text}%' ";

if (textBox\_inventoryPlacement\_roomName.Text != string.Empty)

roomQuerry = $"AND r.name LIKE '%{textBox\_inventoryPlacement\_inventoryName.Text}%' ";

string FilterQuery = $"{inventoryQuerry} {inventoryNumQuerry} {buildingQuerry} {roomQuerry}";

update\_datagrid\_inventory\_placement(FilterQuery);

string filter = "";

if (!String.IsNullOrEmpty(textBox\_inventoryPlacement\_roomName.Text))

{

filter += string.Format("[Помещение] LIKE '%{0}%'", textBox\_inventoryPlacement\_roomName.Text);

}

(dataGridView\_inventory\_placement.DataSource as System.Data.DataTable).DefaultView.RowFilter = filter;

}

private void button\_inventoryplacement\_filterReset\_Click(object sender, EventArgs e)

{

textBox\_inventoryPlacement\_inventoryName.Clear();

comboBox\_inventoryPlacement\_building.SelectedIndex = -1;

textBox\_inventoryPlacement\_roomName.Clear();

update\_datagrid\_inventory\_placement(string.Empty);

}

private void tab\_inventory\_placement\_Enter(object sender, EventArgs e)

{

update\_CB\_building(comboBox\_inventoryPlacement\_building);

update\_datagrid\_inventory\_placement(string.Empty);

}

private void dataGridView\_inventory\_placement\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

new Перемещение(dataGridView\_inventory\_placement.SelectedRows[0].Cells[0].Value).ShowDialog();

}

private void textBox\_inventoryPlacement\_inventoryName\_TextChanged(object sender, EventArgs e)

{

create\_additional\_querry\_for\_InventoryPlacement\_Datagrid();

}

private void comboBox\_inventoryPlacement\_building\_SelectedValueChanged(object sender, EventArgs e)

{

create\_additional\_querry\_for\_InventoryPlacement\_Datagrid();

}

private void textBox\_inventoryPlacement\_roomName\_TextChanged(object sender, EventArgs e)

{

create\_additional\_querry\_for\_InventoryPlacement\_Datagrid();

}

private void textBox\_inventoryPlacement\_invNUM\_TextChanged(object sender, EventArgs e)

{

create\_additional\_querry\_for\_InventoryPlacement\_Datagrid();

}

#endregion

#region arrival

private void tab\_inventory\_arrival\_Enter(object sender, EventArgs e)

{

try

{

update\_CB\_building(comboBox\_arrival\_building);

arrival\_clean\_form();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

struct inventory\_item

{

public string inventory\_num;

public string inventory\_name;

public string inventory\_description;

public double inventory\_price;

public int room\_id;

}

Dictionary<string, inventory\_item> inventory\_list = new Dictionary<string, inventory\_item>();

private void update\_CB\_arrival\_Room()

{

db.conn.Open();

DataSet ds = new DataSet();

try

{

new SqlDataAdapter(

$@"SELECT ID, [name] from [dbo].[room] where id\_building = {comboBox\_arrival\_building.SelectedValue}",

db.conn).Fill(ds);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "\n" + new StackFrame(0, true).GetFileLineNumber());

}

db.conn.Close();

comboBox\_arrival\_room.DataSource = ds.Tables[0];

comboBox\_arrival\_room.DisplayMember = "Name";

comboBox\_arrival\_room.ValueMember = "ID";

}

private void arrival\_add\_To\_List\_and\_ListBox()

{

inventory\_item ii = new inventory\_item();

ii.inventory\_num = textBox\_arrival\_invNum.Text;

ii.inventory\_name = textBox\_arrival\_invName.Text;

ii.inventory\_price = (double)numericUpDown\_arrival\_invPrice.Value;

ii.inventory\_description = textBox\_arrival\_invDescr.Text;

ii.room\_id = (int)comboBox\_arrival\_building.SelectedValue;

inventory\_list.Add(ii.inventory\_num, ii);

listBox1.Items.Add(ii.inventory\_num);

}

private void button\_add\_to\_Arrival\_List\_Click(object sender, EventArgs e)

{

arrival\_add\_To\_List\_and\_ListBox();

}

private void button\_change\_in\_arrival\_list\_Click(object sender, EventArgs e)

{

string key = listBox1.SelectedItem.ToString();

inventory\_list.Remove(key);

listBox1.Items.Remove(key);

arrival\_add\_To\_List\_and\_ListBox();

MessageBox.Show(key);

}

private void button\_delete\_from\_arrival\_list\_Click(object sender, EventArgs e)

{

arrival\_clean\_form();

}

private void arrival\_write\_data\_in\_DataBase(int arrival\_id, string key, DateTime date)

{

// create [dbo].[inventory]

new SqlCommand(

$@"INSERT INTO [dbo].[inventory]([inv\_num],[name],[description])

VALUES('{key}','{inventory\_list[key].inventory\_name}','{inventory\_list[key].inventory\_description}')",

db.conn).ExecuteNonQuery();

// create [dbo].[list\_of\_arrival]

new SqlCommand(

$@"INSERT INTO [dbo].[list\_of\_arrival]([inventory\_num],[arrival\_id],[room\_id],[price])

VALUES('{key}',{arrival\_id},{inventory\_list[key].room\_id},{inventory\_list[key].inventory\_price})",

db.conn).ExecuteNonQuery();

// create [dbo].[inventory\_movement\_log]

new SqlCommand(

$@"INSERT INTO [dbo].[inventory\_movement\_log]([date],[login],[inventory\_num],[room\_to])

VALUES('{date}','{token.login}','{key}',{inventory\_list[key].room\_id})",

db.conn).ExecuteNonQuery();

}

private void button\_create\_arrival\_Click(object sender, EventArgs e)

{

try

{

DateTime date = DateTime.Now;

db.conn.Open();

// create [dbo].[inventory\_arrival]

new SqlCommand(

$@"INSERT INTO [dbo].[inventory\_arrival](date,login) VALUES('{date}','{token.login}')",

db.conn).ExecuteNonQuery();

// take last id from inventory arrival to add this in other tables

SqlDataReader sdr\_id = new SqlCommand(

$@"SELECT id FROM [dbo].[inventory\_arrival] WHERE date = '{date}'",

db.conn).ExecuteReader();

sdr\_id.Read();

int arrival\_id = sdr\_id.GetInt32(0);

sdr\_id.Close();

db.conn.Close();

foreach (string inv\_item\_key in inventory\_list.Keys)

{

db.conn.Open();

arrival\_write\_data\_in\_DataBase(arrival\_id, inv\_item\_key, date);

db.conn.Close();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "\n" + new StackFrame(0, true).GetFileLineNumber());

return;

}

arrival\_clean\_form();

listBox1.Items.Clear();

inventory\_list.Clear();

MessageBox.Show("Приход оформлен успешно!");

}

private void arrival\_clean\_form()

{

try

{

foreach (Control item in tab\_inventory\_arrival.Controls)

{

if (item.GetType() == typeof(System.Windows.Forms.TextBox) || item.GetType() == typeof(NumericUpDown))

{

item.Text = "";

}

}

comboBox\_arrival\_building.SelectedIndex = -1;

comboBox\_arrival\_room.SelectedIndex = -1;

DataSet ds = new DataSet(); ds.Tables.Add();

comboBox\_arrival\_room.DataSource = ds.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void comboBox\_arrival\_building\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox\_arrival\_building.SelectedIndex != -1)

update\_CB\_arrival\_Room();

}

#endregion

}

}

**Форма Перемещение**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Diagnostics;

using System.Windows.Forms;

namespace Diploma

{

public partial class Перемещение : Form

{

public Перемещение(object data)

{

InitializeComponent();

update\_CB\_building();

update\_cb\_room();

update\_information\_selected\_inventory(data);

comboBox1.SelectedIndex = -1;

comboBox2.SelectedIndex = -1;

}

private int id\_room\_from;

private void update\_CB\_building()

{

db.conn.Open();

DataSet ds = new DataSet();

try

{

new SqlDataAdapter(

$@"SELECT ID, [Building\_Address] from [dbo].[building]",

db.conn).Fill(ds);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "\n" + new StackFrame(0, true).GetFileLineNumber());

}

db.conn.Close();

comboBox1.DisplayMember = "Building\_Address";

comboBox1.ValueMember = "ID";

comboBox1.DataSource = ds.Tables[0];

}

private void update\_cb\_room()

{

db.conn.Open();

DataSet ds = new DataSet();

try

{

new SqlDataAdapter(

$@"SELECT ID, [name] from [dbo].[room]

Where room.id\_building = {comboBox1.SelectedValue}",

db.conn).Fill(ds);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "\n" + new StackFrame(0, true).GetFileLineNumber());

}

db.conn.Close();

comboBox2.DisplayMember = "name";

comboBox2.ValueMember = "ID";

comboBox2.DataSource = ds.Tables[0];

}

private void update\_information\_selected\_inventory(object data)

{

db.conn.Open();

SqlDataReader sdr = new SqlCommand(

$@"SELECT iml.inventory\_num, i.name, i.description, r.name, b.Building\_Address, r.id

FROM [dbo].[inventory\_movement\_log] as iml

JOIN inventory as i ON i.inv\_num = iml.inventory\_num

JOIN room as r ON r.id = iml.room\_to

JOIN building as b ON b.id = r.id\_building

WHERE iml.inventory\_num = '{data.ToString()}' and

iml.date=

(select max(date)

from inventory\_movement\_log

where iml.inventory\_num = inventory\_num)",

db.conn).ExecuteReader();

sdr.Read();

textBox\_building.Text = sdr.GetValue(4).ToString();

textBox\_invDescr.Text = sdr.GetValue(2).ToString();

textBox\_invName.Text = sdr.GetValue(1).ToString();

textBox\_invNum.Text = sdr.GetValue(0).ToString();

textBox\_room.Text = sdr.GetValue(3).ToString();

id\_room\_from = (int)sdr.GetValue(5);

sdr.Close();

db.conn.Close();

}

private void comboBox1\_DropDownClosed(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex >= 0)

{

comboBox2.Enabled = true;

try

{

update\_cb\_room();

}

catch (Exception)

{

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex < 0 || comboBox2.SelectedIndex < 0)

{

MessageBox.Show("Выберите куда хотите переместить имущество");

return;

}

db.conn.Open();

new SqlDataAdapter(

$@"INSERT INTO [dbo].[inventory\_movement\_log]

([date],[login],[inventory\_num],[room\_from],[room\_to])

VALUES('{DateTime.Now}','{token.login}','{textBox\_invNum.Text}',

{id\_room\_from},{comboBox2.SelectedValue})",

db.conn).

Fill(new DataSet());

db.conn.Close();

}

}

}

# ПРИЛОЖЕНИЕ 2

**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ для «Кладовщик»**

Пользователю с ролью «Кладовщик» необходимо после запуска приложения ввести свои данные для авторизации (Рис. 17).

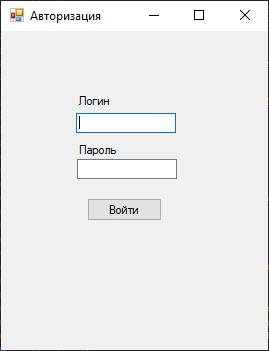


Рис.

После авторизации пользователь может просмотреть перечень всего имущество находящегося в институте (Рис. 18).

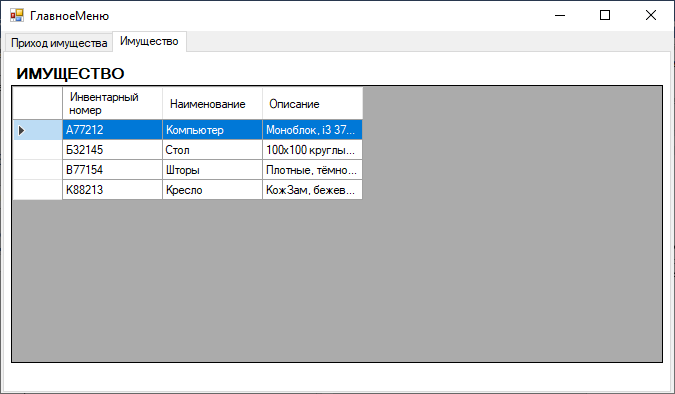


Рис.

На вкладке «Приход имущества» пользователь видит несколько полей для заполнения (Рис. 19). На данной вкладке пользователь после заполнения полей добавляет имущество в список прихода путем нажатия кнопки «Добавить в приход».

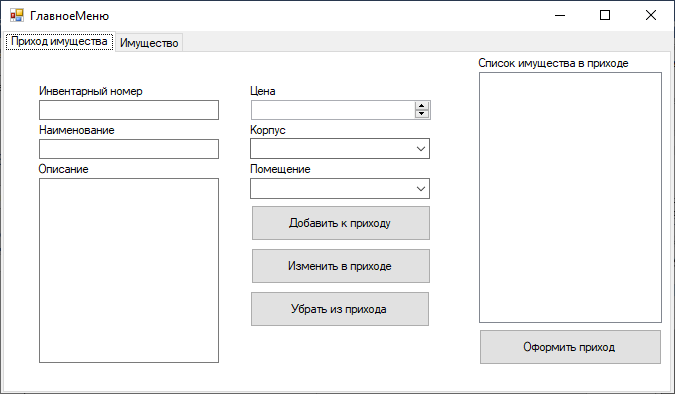


Рис.

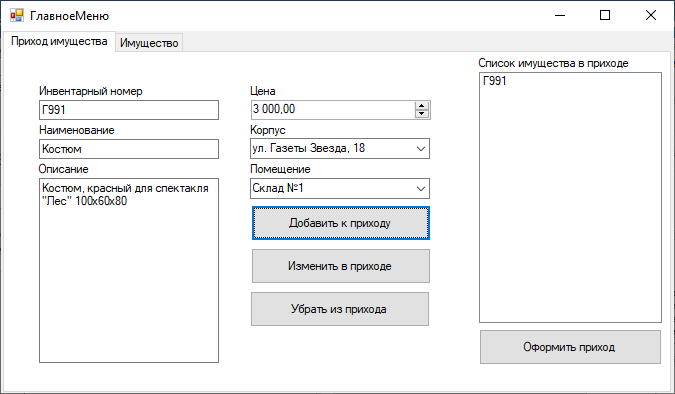
После нажатия на кнопку «Добавить в приход» имущество добавляется в список прихода (Рис. 20). 

Рис.

После добавления всего нужного имущества пользователь должен нажать на кнопку «Оформить приход». После чего всё имущество из списка будет добавлено в базу и пользователь получит сообщение обратной связи (Рис. 21).

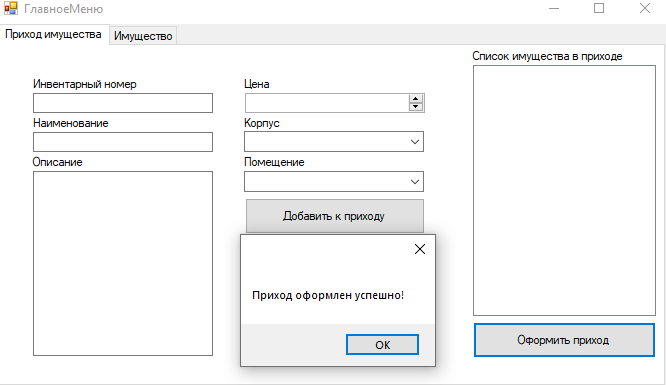


Рис.

**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ для «Менеджер»**

Пользователю с ролью «Кладовщик» необходимо после запуска приложения ввести свои данные для авторизации (Рис. 22).

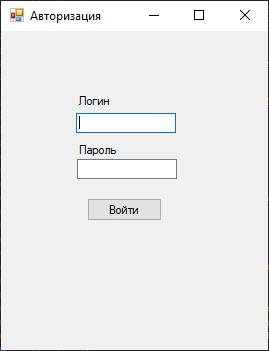


Рис.

После авторизации пользователь попадает на вкладку «Расположение имущества» (Рис. 23).

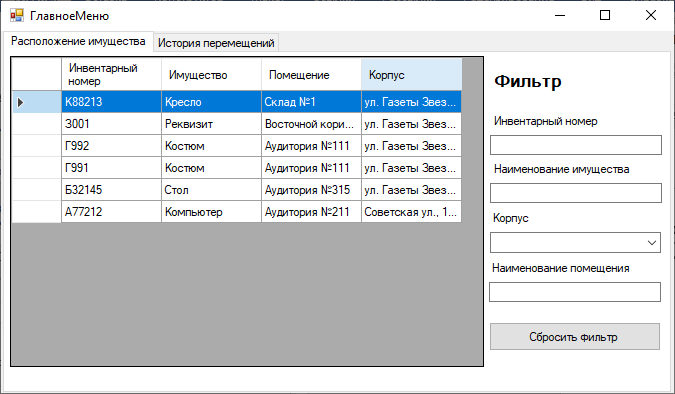


Рис.

На данной вкладке пользователь видит список имущества, который он может отфильтровать. По двойному нажатию на имущество из списка, откроется форма по перемещению имущества (Рис. 24).

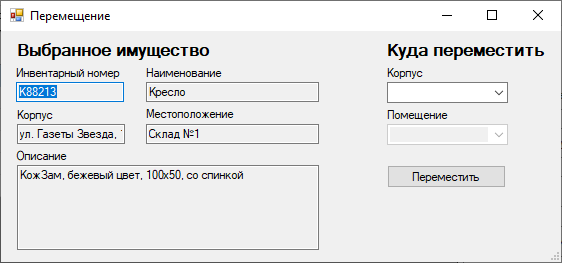


Рис.

На этой форме пользователь видит всю информацию о выбранном имуществе и два выпадающих списка в которых необходимо выбрать место назначения перемещаемого имущества. После выбора места назначения пользователю необходимо нажать на кнопку «переместить».

На вкладке «История перемещений» пользователь сможет посмотреть историю перемещений имущества за всё время (Рис. 25). Эту историю можно отфильтровать и сформировать отфильтрованный отчёт по нажатию на кнопку «Сформировать отчет» (Рис. 26). Отчет формируется с расширением excel на рабочем столе (Рис. 27)

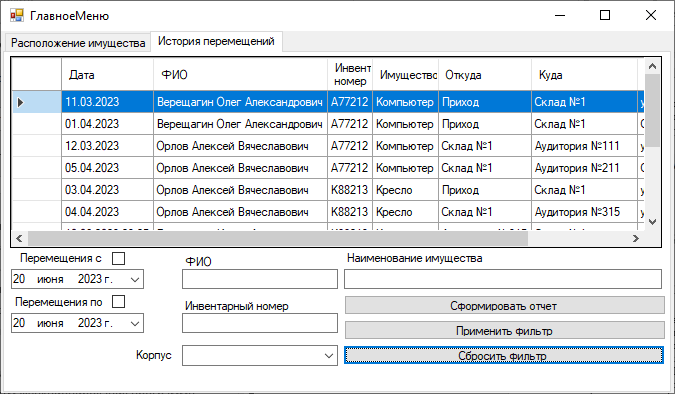


Рис.

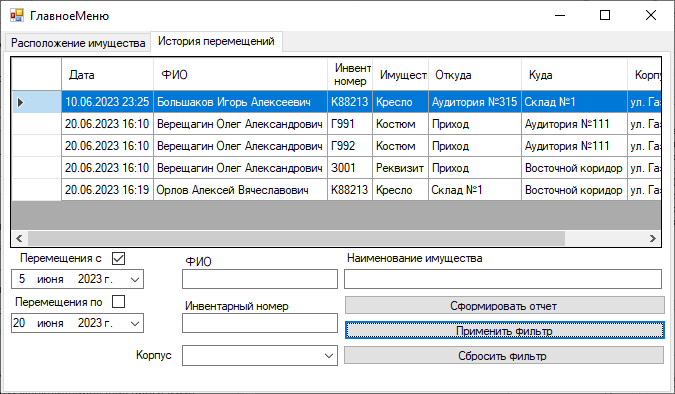


Рис.

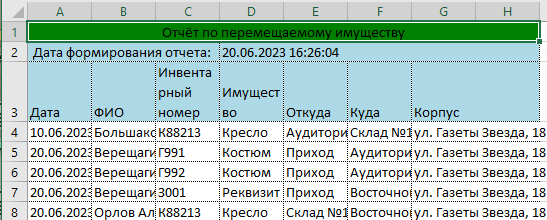


Рис.

**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ для «Администратор»**

Пользователю с ролью «Кладовщик» необходимо после запуска приложения ввести свои данные для авторизации (Рис. 28).

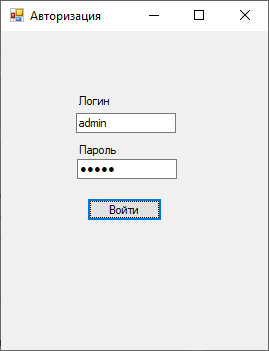


Рис.

Пользователь под ролью «Администратор» может выполнять те же функции, что «Менеджер» и «Кладовщик». Так же «Администратор» может добавлять новых пользователей в систему на вкладке «Добавление пользователя»(Рис. 29).

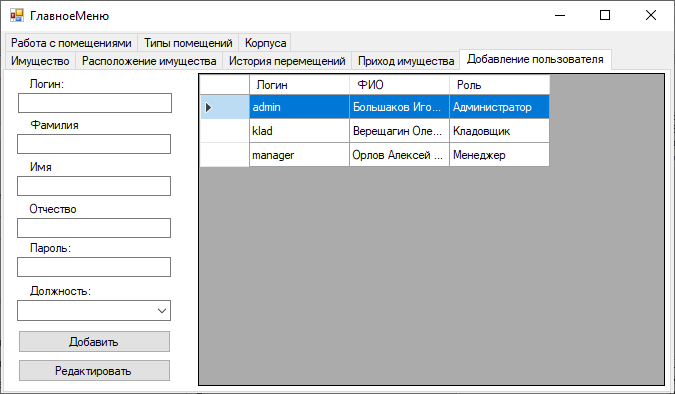


Рис.

На вкладке «Работа с помещениями» пользователь может просматривать список всех помещений в базе данных (Рис. 30). Администратор сможет добавлять новые помещения в базу данных или редактировать старые путем нажатия на соответствующие кнопки «Добавить» и «Редактировать».

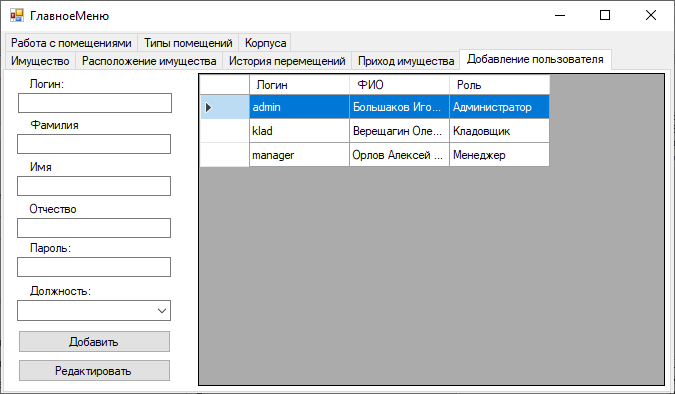


Рис.

На вкладке «Типы помещений» пользователь сможет просмотреть перечень всех типов помещений (Рис. 31). Так же пользователь сможет добавить новый тип помещений в базу данных или отредактировать старый путем нажатия на соответствующие кнопки «Добавить новый тип помещения» и «Изменить выбранный тип помещения».

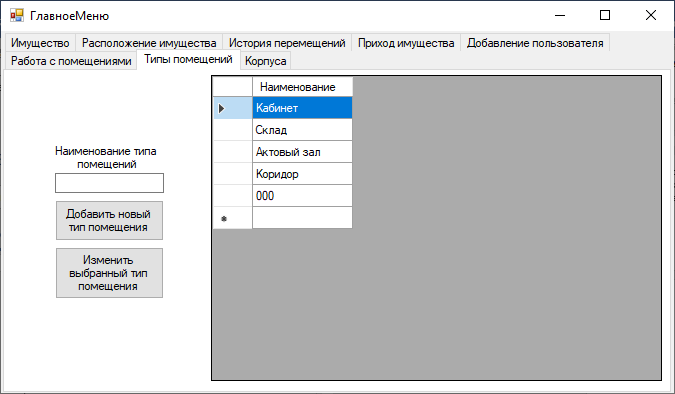


Рис.

На вкладке «Корпуса» пользователь может видеть все доступные корпуса института (Рис. 32). Так же можно добавить новый корпус или изменить адрес существующего путём нажатия на соответствующие кнопки «Добавить новый корпус» и «Изменить выбранный корпус».

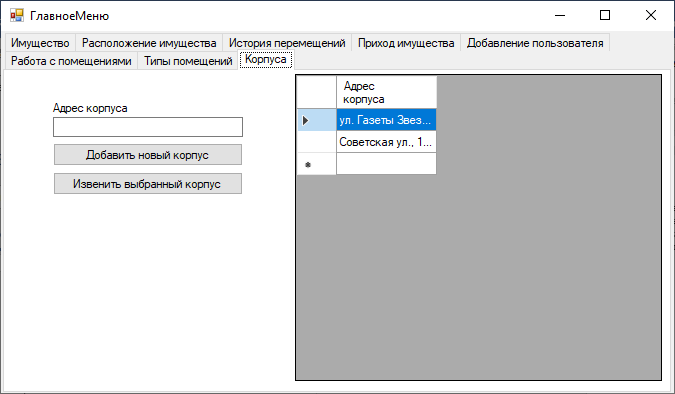


Рис.