

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**   
**«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»**

Факультет Физико-технический

Кафедра Компьютерных технологий (КТ)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | КТ |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Т. В. Ермоленко |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе бакалавра 2 курса

на тему:

|  |
| --- |
| РАЗРАБОТКА ПРИЛОЖЕНИЯ УЧЕТА |
| СТУДЕНТОВ В ОБЩЕЖИТИИ |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| Автор работы |  | И.В. Михайский |  |

подпись

|  |  |  |
| --- | --- | --- |
| Направление | 09.03.01 | Информатика и вычислительная техника |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преп. В.Н. Котенко

подпись

Консультанты по разделам:

Сетевые технологии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель А.Е. Гукай

подпись

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. лаборант В.Г. Медведева

подпись

Курсовая работа защищена \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата итоговая оценка комиссия

Подписи членов комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

Донецк  
2017

ГОУ ВПО «Донецкий национальный университет»

кафедра Компьютерных технологий

Утверждаю

Зав. кафедрой КТ

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

подпись

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

дата

**ЗАДАНИЕ**

на курсовую работу студента 2 курса Михайский И.В.

*Тема курсовой работы*: Разработка приложения учета студентов в общежитии

*Краткая постановка задачи*: 1. Изучить последовательность действий при поселении в общежитие, провести анализ аналогичных программных продуктов 2. Разработать техническое задание. 3. Описать функциональную модель проекта 5. Разработать проект программного обеспечения приложения. 6. Разработать функционирующую программу. 7. Протестировать программное обеспечение и оценить его качество. 8. Оформить отчёт.

*Исходные данные*: 1. Инструкция по поселению в общежитие. 2. Документация по Visual Studio. 3. Документация по Windows Forms. 4. Документация по C#.

*Ожидаемые результаты*: Приложение для учета студентов в общежитии.

*Календарный план работы:*

|  |  |  |
| --- | --- | --- |
| Даты консультаций | Этапы выполнения работы | Отметки о выполнении |
| 30.02.2017 | Постановка задачи и обсуждение литературы | выполнено |
| 13.03.2017 | Предварительное утверждение содержания отчёта | выполнено |
| 20.03.2017 | Утверждение проекта, алгоритмов, методов, технологий | выполнено |
| 27.03.2017 | Ход реализации проекта | выполнено |
| 01.04.2017 | Обсуждения организации тестирования программы | выполнено |
| 13.04.2017 | Демонстрация программного продукта руководителю | выполнено |
| 21.04.2017 | Оформление отчёта | выполнено |
| 22.04.2017 | Предоставление отчёта руководителю | выполнено |

Дата выдачи задания 30.01.2017 года

Студент И.В. Михайский

Руководитель В.Н. Котенко

**АННОТАЦИЯ**

Отчёт о курсовой работе: 51 с., 18 рис., 5 табл., 2 приложения,   
6 источников.

Объект исследования – процесс поселения и проживания в общежитии.

Цель работы – разработка приложения учета проживания студентов в общежитии, разработка базы данных для приложения.

Метод исследования – изучение принципов разработки приложений средствами фреймворка .NET, с помощью Windows Form.

В результате решения задачи было разработано работоспособное приложение для учета проживания студентов в общежитии. Данное приложение может быть использовано в сфере предоставления коммунальных услуг.

Дальнейшее развитие программы связано с расширением ее возможностей, увеличением числа опций.

ПРИЛОЖЕНИЕ, ПОСЕЛЕНИЕ В ОБЩЕЖИТИИ, FRAMEWORK, C#, WINDOWS FORM, VISUAL STUDIO

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc482570995)

[1 Анализ предметной области 6](#_Toc482570996)

[1.1 Состояние вопроса 6](#_Toc482570997)

[1.2 Процесс поселения в общежитие 7](#_Toc482570998)

[1.3 Актуальность и цель работы 9](#_Toc482570999)

[2 Техническое задание 10](#_Toc482571000)

[2.1 Описание области применения и исходных данных приложения 10](#_Toc482571001)

[2.2 Требования к пользовательским интерфейсам 10](#_Toc482571002)

[2.3 Требования к аппаратным и программным интерфейсам 14](#_Toc482571003)

[2.4 Требования к пользователям продукта 14](#_Toc482571004)

[2.5 Функции продукта 15](#_Toc482571005)

[2.6 Ограничения 15](#_Toc482571006)

[2.7 Сценарии использования 16](#_Toc482571007)

[2.7.1 Сценарии использования окна авторизации 16](#_Toc482571008)

[2.7.2 Сценарии использования окна справки 16](#_Toc482571009)

[2.7.3 Сценарии использования главного окна приложения 17](#_Toc482571010)

[3 Обоснование выбора инструментальных средств для реализации поставленной задачи 18](#_Toc482571011)

[3.1 Язык программирования С# 18](#_Toc482571012)

[3.2 .NET Framework 18](#_Toc482571013)

[3.3 Windows Forms 20](#_Toc482571014)

[3.4 Visual Studio 20](#_Toc482571015)

[4 Разработка ПО 22](#_Toc482571016)

[4.1 Входные и выходные данные 22](#_Toc482571017)

[4.2 Структура приложения 24](#_Toc482571018)

[5 Тестирование программного продукта 32](#_Toc482571019)

[5.1 Аппаратные, системные и программные требования 32](#_Toc482571020)

[5.2 Руководства установки и использования 32](#_Toc482571021)

[5.3 Описание контрольных примеров 33](#_Toc482571022)

[ЗАКЛЮЧЕНИЕ 35](#_Toc482571023)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc482571024)

[ПРИЛОЖЕНИЕ А 37](#_Toc482571025)

[ПРИЛОЖЕНИЕ Б 40](#_Toc482571026)

[Листинг Б.1 – Файл MineForm.cs 40](#_Toc482571027)

[Листинг Б.2 – Фрагмент файла ExtendedBitmap.cs 47](#_Toc482571028)

[Листинг Б.3 – Фрагмент файла UserClass.cs 51](#_Toc482571029)

# ВВЕДЕНИЕ

Многие из студентов ДонНУ являются иногородними, поэтому при поступлении в университет становится актуальным вопрос о поселении в общежитие. Данный процесс занимает отнюдь немало времени вследствие огромных очередей на подачу документов, получение подписей и печатей. Также существует ряд проблем и вопросов, связанных с проживанием в общежитии и у студентов старших курсов: оплата проживания, печать нормативной документации, обратная связь с персоналом, получение информации о штрафах и задолженностях, связь с оперотрядом.

Именно поэтому на сегодняшний день как никогда важна задача автоматизации бумажного документооборота, взаимодействие между студентами и управляющим звеном студгородка. Ее оперативное и эффективное решение позволит экономить время студентов и работников студенческого городка, ускорить процесс поселения, обмена информацией, а также уменьшить затраты средств на расходные материалы.

Целью курсовой работы является разработка приложения учета проживания студентов в общежитии. Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Изучить последовательность действий при поселении в общежитие, провести анализ аналогичных программных продуктов.
2. Изучить технологии: .NET, Windows Forms, C#, Visual Studio 2015.
3. Разработать структуру и функционал программы, соответствующий техническому заданию.

# 1 Анализ предметной области

## 1.1 Состояние вопроса

На данный момент задача автоматизации бумажного документооборота важна и актуальна, для ее решения предлагается множество способов. Поэтому перед созданием собственного программного продукта я проанализировал похожую систему учета студентов с целью выявления недостатков и достоинств, уточнения терминологии. На основе данных полученных при анализе будет составлено техническое задание для данного курсового проекта, а также поставлены частные задачи.

Анализировалась программа учета проживания студентов в Саратовском государственном университете (таблица 1).

Таблица 1 – Анализ программы учета проживания студентов в Саратовском государственном университете

|  |  |
| --- | --- |
| Критерий | Комментарий |
| Название | Учет проживания студентов университета в общежитиях |
| Страна | Россия |
| Функционал | 1. Ввод и редактирование информации в справочниках системы;  2. Ввод и редактирование информации в справочниках системы, общей информации о студенте, а также информации о проживании студента в период обучения в университете (номера и даты соответствующих служебных записок); |

Окончание таблицы 1

|  |  |
| --- | --- |
|  | 3. Осуществление автоматизированного управления проживанием студентов: вселение, выселение, переезд в другие комнаты; |
| Недостатки | Отсутствие обратной связи;  Отсутствие оповещений о нарушениях;  Отсутствие связи с платежными системами; |
| Достоинства | Автоматизированный механизм переезда в другую комнату; |
| Стадия внедрения | Программа готова к внедрению |

В ходе анализа были обнаружены недостатки системы, связанные с отсутствием некоторого функционала, а также с организацией обратной связи. Также нет связи с платежными системами для оплаты проживания в общежитии и некоторая часть работы, связанной с учетом активности студентов все еще осуществляется в бумажном виде. Система требует значительной доработки и тестирования на месте.

## 1.2 Процесс поселения в общежитие

Моделирование бизнес-процессов является крайне важным этапом разработки, так как позволяет оценить работу предприятия и в дальнейшем оптимизировать его работу.

Для описания бизнес-процессов для данного курсового проекта была выбрана нотация IDEF0, которая позволяет описать процессы в их функциональном аспекте. Главными компонентами моделей являются блоки и дуги. В свою очередь дуги и их вхождение обуславливают тип интерфейса:

* Если дуга входит в блок сверху – это управляющая информация
* Если дуга входит в блок слева – это входная информация
* Если дуга выходит из блока справа – это результаты
* Если дуга выходит из блока снизу – это механизм

В основе выбранной нотации лежит принцип о том, что каждый компонент модели может быть разложен на мелкие детали (декомпозирован) с целью детализации процесса, наглядного представления.

На рисунке 1 представлена контекстная диаграмма. Ее задача – отображение в целом процессов, которые описывают структуру и функции проекта.



Рисунок 1 – Контекстная диаграмма

На диаграмме (рисунок 1) реализованы следующие интерфейсы:

* Управляющая информация – правила общежития, нормативная документация
* Входная информация – необходимый пакет документов, заявление на поселение, соглашение на проживание, соглашение на использование данных
* Результаты – выданные ключи от комнаты, пропуск
* Механизм – паспортный стол, военкомат №2

## 1.3 Актуальность и цель работы

Автоматизация процесса учета проживания студентов в общежитии является актуальной, так как позволяет решить ряд проблем, возникающих при поселении и дальнейшем проживании. Внедрение разрабатываемого программного продукта позволит:

* Организовать процесса поселения и выселения без очередей.
* Обеспечить обратной связью между руководством, персоналом студенческого городка и студентами.
* Экономить время студентов и работников студенческого городка.
* Уменьшить затраты на расходные материалы.

Целью курсовой работы является разработка приложения учета проживания студентов в общежитии. Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Изучить последовательность действий при поселении в общежитие, провести анализ аналогичных программных продуктов.
2. Изучить технологии: Spring MVC, AJAX, jQuery, HTML5, CSS3, PostgreSQL, IDE NetBeans.
3. Разработать структуру и функционал программы, соответствующий техническому заданию.

# 2 Техническое задание

## 2.1 Описание области применения и исходных данных приложения

Данное приложение предназначено для учета студентов в общежитии. Его можно использовать в сферах обслуживания и коммунальных услуг, оно может использоваться для:

* Регистрации студентов общежития и персонала.
* Управления учетными записями студентов и персонала.
* Записи основных данных студентов общежития и персонала, а также составлению заметок к ним.
* Создания файловой структуры с данными о пользователях.

Исходными данными для приложения являются данные о студентах и персонале общежития:

* Логин (должен быть уникальным).
* Пароль.
* Полное имя.
* Номер комнаты.
* Дата регистрации.
* Уровень доступа.
* Заметки (не обязательно).

## 2.2 Требования к пользовательским интерфейсам

Интерфейс программы должен быть интуитивно понятным, чего можно достигнуть, использовав стандартные элементы управления Windows. Приложение должно иметь минимум три окна фиксированных размеров: окно для авторизации и главное окно приложения. Среди элементов интерфейса обязательно должны быть стандартные кнопки закрытия и свертывания приложения, но не должно быть кнопки свертывания.

Окно авторизации должно иметь структуру, изображенную на рисунке 2.

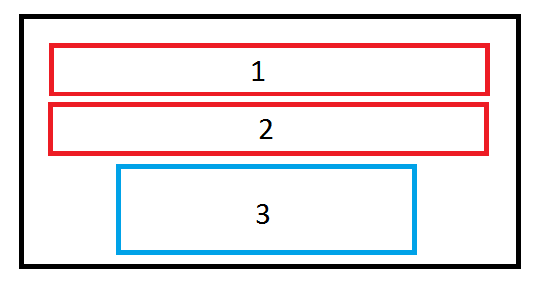


Рисунок 2 – Схема интерфейса окна авторизации

На рисунке 2 изображены следующие блоки интерфейса:

1. Элемент для ввода логина и элемент с подписью к нему.
2. Элемент для ввода пароля, элемент с подписью к нему и элемент checkbox с установкой флага для показа пароля.
3. Кнопки для авторизации и вызова справки по приложению.

Главное окно приложения должно отображать как данные о конкретном пользователе приложения, так и список пользователей приложения, в связи с чем следует отделить друг от друга элементы управления, отвечающие за эти функции. Однако, есть информация, которую пользователь должен видеть всегда: данные об авторизированном пользователе (ФИО, уровень доступа) и кнопки управления приложением. Следует вынести в интерфейс приложения элементы управления со следующим функционалом:

* Открытие вкладки с подробной информацией о текущем пользователе.
* Кнопка открытия регистрации.
* Кнопка просмотра списка всех пользователей.
* Кнопка сохранения внесенных изменений

Схему интерфейса главного окна приложения, с учетом особенностей, можно рассмотреть на рисунке 3. На ней видно всего два блока. Первый блок схемы – панель с информацией и управляющими элементами интерфейса, описанными высшее. Второй блок содержит элемент с вкладками.

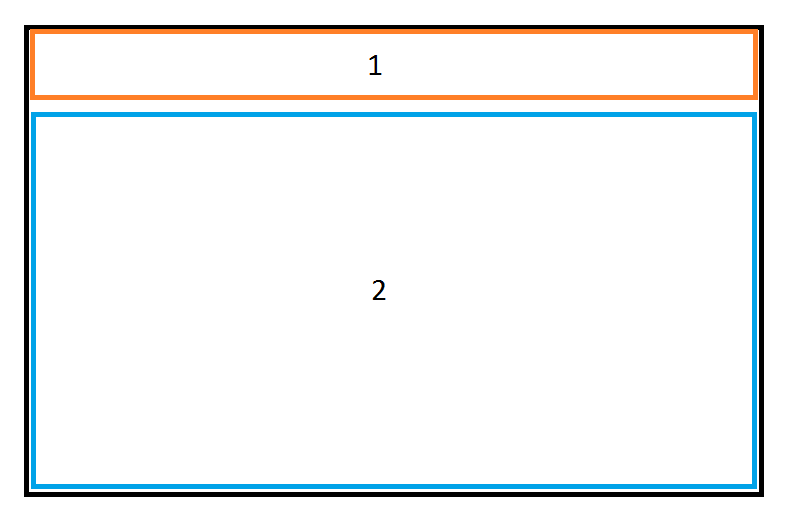


Рисунок 3 - Схема интерфейса главного окна

Всего на главном окне две вкладки, каждая из которых имеет свой интерфейс. Первая вкладка содержит всего один элемент интерфейса – таблицу с зарегистрированными пользователями. Данная таблица не должна позволять пользователю редактировать данные в ней, но должна позволять сортировать данные по каждому из полей. Столбцы данной таблицы должны отображать следующие данные:

* Полное имя пользователя.
* Уровень доступа.
* Комната.
* Дата регистрации.

При клике на полное имя пользователя должна открываться вкладка с его подробными данными.

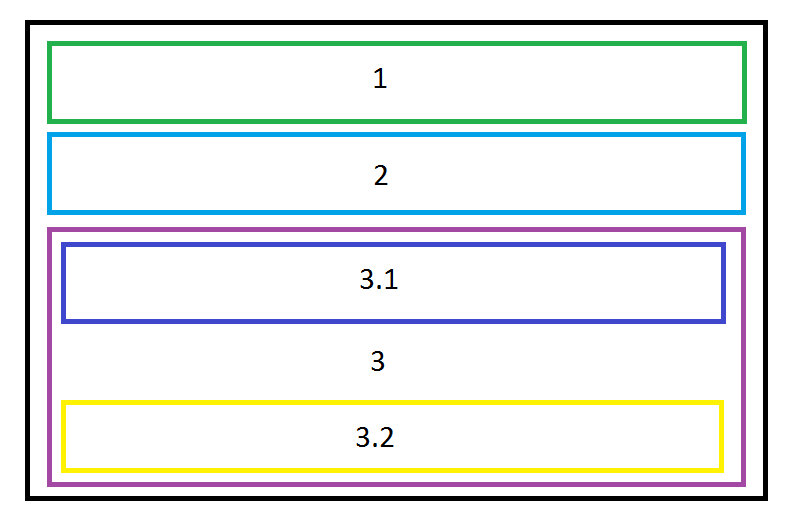
Вторая вкладка главного окна содержит элементы, содержащие подробные данные о пользователе. Данная вкладка открывается при: просмотре собственного аккаунта пользователя, просмотре чужого аккаунта, создании нового пользователя. Схема интерфейса данной вкладки изображена на рисунке 4.

Рисунок 4 - Схема интерфейса вкладки с информацией об пользователе

Внимательнее рассмотрим блоки интерфейса вкладки, изображенные на рисунке 4:

1. Блок с подписью к логину и текстовый элемент ввода логина
2. Блок с подписью к паролю, текстовый элемент ввода пароля и checkbox для установки состояния видимости пароля
3. Блок с основной информацией пользователя
   1. Блок, содержащий элементы для ввода и отображения следующих данных: полное имя пользователя, номер комнаты, уровень доступа, дата регистрации
   2. Блок с элементом для ввода и отображения заметок об пользователе.

Для каждого элемента из блока 3.1 (рисунок 4) должен соответствовать свой элемент интерфейса для удобного и корректного ввода данных. То есть, в элементе для ввода номера комнаты не должно быть возможности ввести текст или число, превышающее диапазон комнат. Элементы должны быть доступны только, если у текущего пользователя есть права для их редактирования.

## 2.3 Требования к аппаратным и программным интерфейсам

Для установки и работы программы необходимо иметь вычислительную систему следующей минимальной конфигурации:

* процессор: 1.8 ГГц
* оперативная память: от 512 мб
* видеокарта от 256 мб памяти
* более 50 мб памяти на диске для установки

Необходимо обеспечить программное взаимодействие системы с операционными системами Windows XP/7/8/8.1/10 и необходимо наличие на компьютере установленного Microsoft .NET Framework 4.5.1.

## 2.4 Требования к пользователям продукта

Пользователями программы могут быть разнообразные группы людей, в различном возрастном диапазоне, начиная от студентов, заканчивая заведующими общежития. Пользователи программы должны:

* иметь базовые навыки работы с компьютером
* понимать базовые принципы работы файловой системы
* уметь изучать справочную документацию
* иметь доступ к данным о пользователях (для регистрации)

## 2.5 Функции продукта

Системой должны иметь возможность пользоваться три категории пользователей: администраторы, модераторы и студенты. У каждой группы пользователей есть свои права.

* Администраторы могут редактировать любые данные всех пользователей. К данному типу пользователей можно отнести заведующих общежитием и т.д.
* Студенты имеют права просматривать свои и чужие профили, а также менять пароль доступа.
* Модераторами могут быть секретари или, например, члены оперотряда. Данная группа пользователей имеет право редактировать номера комнат других пользователей и редактировать их описание (чтобы написать жалобу, например).

## 2.6 Ограничения

* продукт будет поддерживать только русский язык пользовательского интерфейса
* продукт не содержит подсказок к элементам интерфейса
* продукт не предусматривает автоматического перехода на платформы, не перечисленные в данном документе
* скорость работы приложения будет зависеть только от производительности машины и степени нагрузки ЦП
* данные внутри приложения прикреплены только к компьютеру пользователя (данные хранятся на диске пользователя и не синхронизируются с данными других экземпляров приложений, установленных на других машинах)

## 2.7 Сценарии использования

Пользователь запускает приложение. Появляется окно авторизации.

## 2.7.1 Сценарии использования окна авторизации

Сценарий 1: Пользователь вводит свой логин и пароль и нажимает на кнопку входа.

Сценарий 1.1: Такого пользователя не существует. Пользователю выведено сообщение с данной ошибкой.

Сценарий 1.2: Такой пользователь существует, но пароль не верный. Пользователю будет показано сообщение с данной ошибкой.

Сценарий 1.3: Такой пользователь существует и пароль верный. Окно авторизации закрывается и открывается главное окно с учетной записью и личными данными пользователя.

Сценарий 2: Пользователь снимает/ставит галочку с элемента напротив элемента для ввода пароля. Пароль показывается/скрывается.

Сценарий 3: Пользователь нажимает кнопку вызова справки. Открывается окно со справкой.

Сценарий 4: Пользователь нажимает стандартную кнопку закрытия окна. Окно закрывается.

## 2.7.2 Сценарии использования окна справки

Окно справки открывается на вкладке

Сценарий 1: Пользователь нажимает на кнопку выбора вкладки. Текущая вкладка изменяется на выбранную.

Сценарий 2: Пользователь нажимает стандартную кнопку закрытия окна. Окно закрывается.

## 2.7.3 Сценарии использования главного окна приложения

Главное окно открывается на вкладке подробной информации о текущем пользователе. На панели отображаются кнопки: «Регистрация пользователя» (Администратор), «Пользователи», «Профиль».

Сценарий 1: Пользователь нажимает на кнопку «Пользователи». Открывается вкладка с таблицей пользователей.

Сценарий 1.1: Пользователь нажимает на заголовок столбца с датами регистрации, данные сортируются по возрастанию дат.

Сценарий 1.2: Пользователь нажимает на одну из записей таблицы в зону столбца с полным именем пользователя. Открывается вкладка с подробной информацией о пользователе, на полное имя которого нажал текущий пользователь.

Сценарий 2: Пользователь нажимает на кнопку «Профиль». Открывается вкладка с подробной информацией о текущем пользователе.

Сценарий 3: Пользователь нажимает на кнопку «Регистрация студента» (Администратор). Открывается вкладка, отображающая подробную информацию о студенте, но все текстовые поля пустые, в поле даты текущая дата, а в элементе выбора роли установлена роль «Студент».

Сценарий 4: Пользователь изменяет данные в одном из доступных полей для редактирования. В верхней панели появляется кнопка «Сохранить».

Сценарий 4.1: Пользователь нажимает на кнопку «Сохранить», которая сразу же пропадает, а данные сохраняются.

Сценарий 4.2 Пользователь открывает другую вкладку, данные не сохраняются.

# 3 Обоснование выбора инструментальных средств для реализации поставленной задачи

## 3.1 Язык программирования С#

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

## 3.2 .NET Framework

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код Common Intermediate Language (CIL). В терминах .NET получается сборка, англ. assembly. Затем код либо исполняется виртуальной машиной Common Language Runtime (CLR), либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Данные процессы видны на рисунке 5.

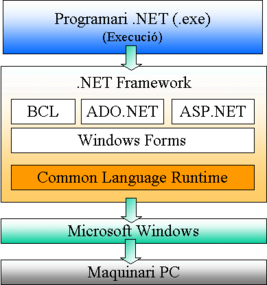


Рисунок 5 – Выполнение программы в .NET

Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR встроенный в неё JIT-компилятор «на лету» (just in time) преобразует промежуточный байт-код в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также сама заботится о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы.

## 3.3 Windows Forms

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

## 3.4 Visual Studio

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения (рисунок 6), веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

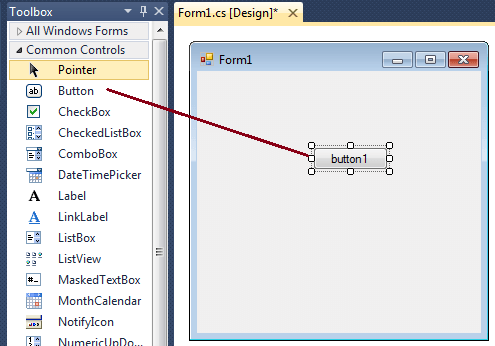


Рисунок 6 – Графический редактор форм

Данный набор библиотек поможет создать понятный для пользователя интерфейс, так как в Windows Form элементы, использованные по умолчанию уже знакомы всем пользователям Windows. Также Windows Form ускорит разработку, не отвлекаясь на разработку графических элементов интерфейса.

# 4 Разработка ПО

## 4.1 Входные и выходные данные

Входными данными для приложения являются данные о пользователях приложения, которыми являются студенты и работники общежития. Данные пользователей рассмотрены в таблице 2:

Таблица 2 - Данные пользователей приложения

|  |  |  |
| --- | --- | --- |
| Данные | Тип данных | Описание/Особенности |
| Логин | Строка | Уникальное непустое значение |
| Пароль | Строка | Любая строка |
| Полное имя | Строка | Любая строка |
| Дата регистрации | Дата | Формат «дд/мм/гггг» |
| Уровень доступа | Целое число | Числам соответствуют уровни:  0 – Студент,  1 – Модератор,  2 – Администратор. |
| Номер комнаты | Целое число | Диапазон от 0 до 1299 включительно. |
| Заметки | Строка | Набор символов, может содержать и перенос строк |

Выходными данными приложения является набор файлов и папок, которые содержат данные приложения. Файловая структура, построенная приложением, имеет вид, изображенный на рисунке 7.

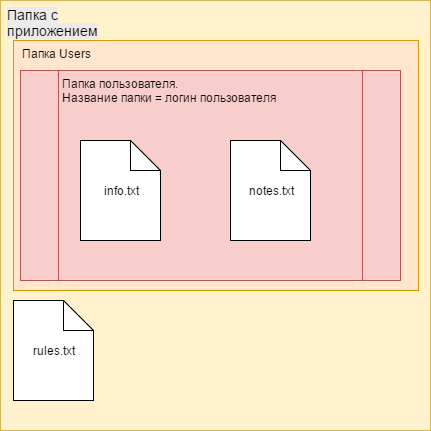


Рисунок 7 Схема файловой структуры, с которой работает приложение

Под «папка с положением» (рисунок 7) имеется ввиду то, что в этой папке находится exe файл приложения. Вся иерархия файлов и папок приложения построена относительно этого пути. Файл rules.txt содержит информацию, которая будет отображена в окне справки. Для каждого пользователя существует своя папка, в которой содержится два файла: файл с информацией о пользователе и отдельный файл с заметками о пользователе.

## 4.2 Структура приложения

### 4.2.1 Файловая структура приложения

Файловая структура проекта приложения изображена на рисунке 8.

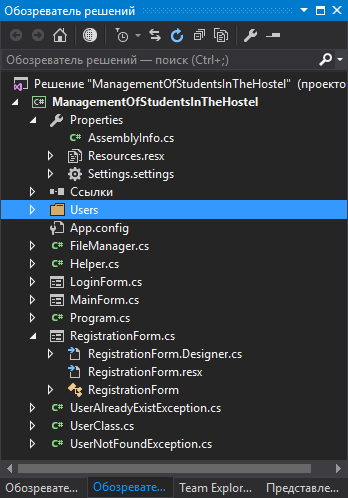


Рисунок 8 - Файловая структура проекта

Рассмотрим детальнее компоненты проекта с приложением:

1. Properties содержит основные настройки приложения (целевая версия платформы .NET Framework, тип сборки и т.д)
   1. AssemblyInfo.cs – файл содержит в себе информацию о сборке
   2. Resourses.resx – файл, описывающий ресурсы
   3. Settings.settings – файл, в котором можно хранить некоторые настройки проекта для динамического изменения
2. References - cодержит все ссылки на внешние компоненты в проекте C#
3. Users – папка, которая копируется к exe-файлу во время компиляции. Данная папка содержит информацию об уже созданных пользователях (с логинами student, moderator, admin), которые позволяют быстро протестировать функционал приложения во время разработки.
4. App.config – конфигурационный файл XML, который может содержать, например строку подключения к БД и т.д.
5. <ИмяОкна>Form.cs – файл, содержащий класс одного из окон приложения, отличается от других файлов классов тем, что к данному файлу прикреплены файлы Windows Form (res с ресурсами окна и designer c инициализацией и настройкой элементов окна)
6. <ИмяКласса>.cs – файлы с классами C#

### 4.2.2 Описание обьектов и их взаимодействия

Структура взаимодействия классов приложения приведена на рисунке 9. Стрелки символизируют, что класс, из которого исходит стрелка содержит экземпляр класса или обращается к статическим методам класса, к которому указывает стрелка. Форма элемента схемы просто разделяет классы по их функционалу. Подробнее про классы, содержащиеся на рисунке 9, можно узнать из таблицы 3.

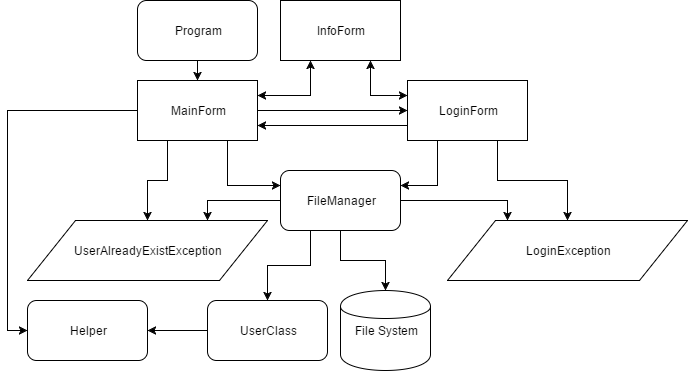


Рисунок 9 - Структура взаимодействия классов приложения

Таблица 3 - Описание классов разрабатываемого приложения

|  |  |
| --- | --- |
| Название класса | Описание |
| Program | Класс, сгенерированный при создании проекта. Данный класс запускает приложение, создав новый экземпляр класса MainForm и передав его в статический метод Run класса Application (библиотеки Windows Form). |
| MainForm, LoginForm, InfoForm | Класс, наследуемый от класса Form, библиотеки Windows Form. Данный класс описан сразу в нескольких файлах (\*.cs, \*.Designer.cs), что позволяет разбить логику, управляемую дизайнером Windows Form (содержащиеся элементы интерфейса и их начальные значения, параметры окна, прописанные в свойствах графического редактирования окна) и логику поведения элементов при разных событиях, которую указывает пользователь. |
| UserClass | Класс, описывающий пользователя. Реализован для простоты передачи параметров. |
| Helper | Класс, содержащий дополнительные (облегчающие разработку) функции приложения. |
| UserAlreadyExistException,  LoginException | Классы пользовательских исключений. |

Окончание таблицы 3

|  |  |
| --- | --- |
| FileManager | Класс, реализующий логику работы с файловой системой. Данный класс агрегирует экземпляр класса UserClass, который отвечает за текущего пользователя, что позволяет значительно облегчить использование этого класса. Единственный экземпляр данного класса создается при создании окна входа, после чего, при удачной авторизации передается в класс Main Form. |

По сути, основными классами приложения являются MainForm, который отвечает за интерфейсную часть приложения и логику работы графических элементов и класс FileManager, который реализует взаимодействие с файловой системой. В таблицах 4 и 5 подробно описаны методы классов MainForm и FileManager соответственно.

Таблица 4 – Методы класса MainForm

|  |  |
| --- | --- |
| Метод | Описание |
| public MainForm() | Конструктор класса, вызывает инициализацию компонентов. |
| private void MainForm\_Load(object sender, EventArgs e) | Метод главного окна приложения, который вызывается при загрузке всех элементов окна. Данный метод сначала прячет главное окно приложения, затем создает и отображает диалог авторизации, а затем, в зависимости от результатов работы окна авторизации, закрывает главное окно, либо отображает его. |

Продолжение таблицы 4

|  |  |
| --- | --- |
| private void linkLabel1\_LinkClicked  (object sender, LinkLabelClickedEventArgs e) | Метод, вызываемый при клике на кнопку «Профиль». Вызывает метод ОpenUser с параметром текущего пользователя. |
| private void OpenAccount  (string login) | Один из основных методов класса. Заполняет вкладку с подробной информацией о пользователе данными, основываясь на переданном в метод параметре login. Данные для заполнения данный метод берет из метода GetUser класса FileManager. Если логин равен пустой строке, то элементы заполняются данными по умолчанию. Так же этот метод устанавливает значения атрибутов Enable элементов интерфейса в зависимости от прав текущего пользователя. |
| private void SaveChangesBtn\_LinkClicked  (object sender,  LinkLabelClickedEventArgs e) | Метод выполняется при нажатии на кнопку «Сохранить». В данном методе все данные с элементов собираются в экземпляр класса UserClass, а затем, в зависимости от того редактируем или создаем пользователя, вызываются методы класса FileManager ChangeUser или CreateUser соответственно. |

Окончание таблицы 4

|  |  |
| --- | --- |
| private void RefreshUsersList() | Метод обновляющий таблицу с пользователями. Получает список пользователей с помощью метода GetAllUsers класса FileManager, а затем устанавливает его как источник данных для таблицы. |
| private void Account\_EditChanged  (object sender, EventArgs e) | Метод, который стоит у всех элементов вкладки с подробной информацией о пользователе на событиях изменения. Данный метод устанавливает свойство Visible кнопки «Сохранить» в значение true. |
| private void dataGridView1\_CellContentClick  (object sender, DataGridViewCellEventArgs e) | Событие нажатия на таблицу с пользователями запускает данный метод. В данном методе учитываются только клики по первому столбцу (с полными именами пользователей), после чего открывается профиль пользователя с помощью метода OpenUser (login нажатого пользователя содержится в скрытом от пользователя поле таблицы). |

Таблица 5 – Методы класса FileManager

|  |  |
| --- | --- |
| Метод | Описание |
| public UserClass GetUser  (string login) | Метод, возвращающий данные о пользователе по его логину. Поочередно открывает файлы с информацией о пользователе и файл с заметками с проверкой на существование файлов. При отсутствии файла с информацией о пользователе выкидывает исключение типа LoginException. |
| public string  GetTextInstr() | Метод, считывающий инструкцию для поселения из файла. При отсутствии файла возвращает пустую строку. |
| public void Login  (string login,  string password) | Метод, практически идентичный методу GetUser, за исключением того, что данный метод ничего не возвращает, так как считывает данные в атрибут user (класса UserClass). Также, проверяет совпадение пароля, переданного в метод с паролем профиля, при несовпадении которых выкидывает исключение типа LoginException. |
| public void CreateUser  (UserClass newUser) | Данный метод сначала проверяет наличие пользователя с таким логином и если такого пользователя нет, то создает директорию для пользователя, а дальше вызывает метод ChangeUser. |
| public void ChangeUser (UserClass newUser) | Поочередно открывает файлы с информацией о пользователе и файл с заметками для их перезаписи. Если файлы не существуют, то создаются. |

# 5 Тестирование программного продукта

## 5.1 Аппаратные, системные и программные требования

Аппаратные требования для работы приложения:

* процессор с тактовой частотой 1.0 ГГц;
* оперативная память 512 Мб и более;
* видеокарта с объёмом памяти 64 Мб и выше;
* монитор 800х600 или с более высоким разрешением.

Для функционирования приложения следует, чтобы на компьютере пользователя были установлены:

* Операционная система —Windows XP или Windows 7/8/8.1/10;
* Microsoft .NET Framework 4.5.1

## 5.2 Руководства установки и использования

### 5.2.1 Руководство пользователю

Для установки приложения на компьютер пользователя необходимо скопировать папку с приложением на диск.

### 5.2.2 Руководство разработчику

Для разработки приложения и его компонентов необходимо:

* иметь установленную Visual Studio 2015 или боле позднюю версию программы с установленными компонентами для работы c языком C#
* наличие на компьютере установленного Microsoft .NET Framework 4.6

## 5.3 Описание контрольных примеров

После запуска приложения появляется окно авторизации, изображенное на рисунке 12.

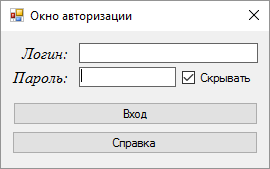


Рисунок 10 – Вид окна авторизации.

При нажатии на кнопку «Справка» открывается новое диалоговое окно (рисунок 13), в котором есть три вкладки: общая информация, о приложении и информация о заселении. При открытии выбрана вкладка «общая информация». При нажатиях на названия вкладок содержимое окна изменяется. После закрытия окна снова открывается окно авторизации.

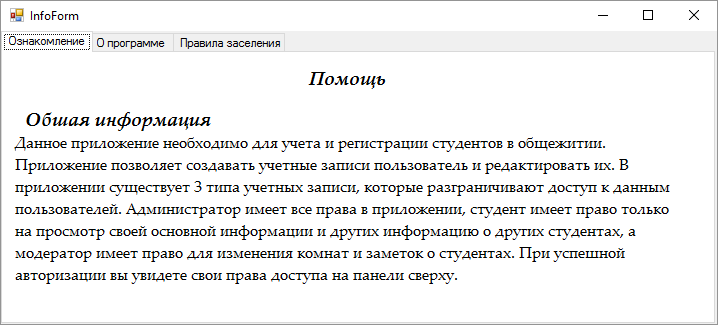


Рисунок 11 - Вид окна справки

Вводится логин и пароль. Вместо символов в элементе ввода пароля видны одинаковые символы, чтобы проверить правильность написания пароля нажимается переключатель «Скрыть» и текст в элементе ввода пароля изменяется на введенный текст. Нажимается кнопка входа, после чего моментально появляется главное окно приложения (рисунок 14).

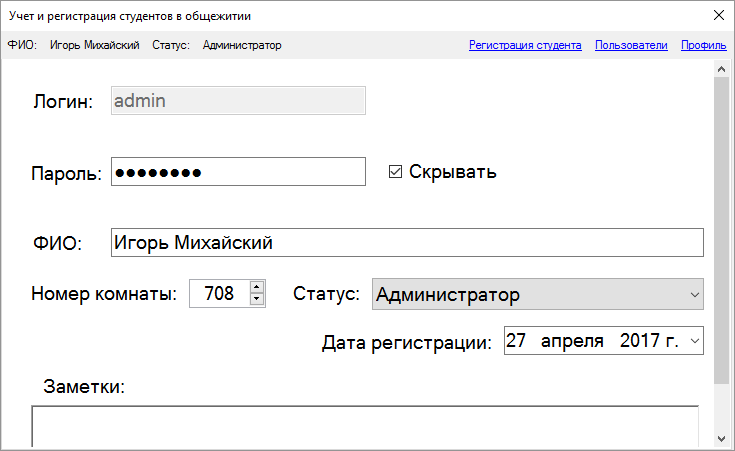


Рисунок 12 - Главное окно приложения

После изменения полного имени в панели сверху появляется кнопка «сохранить», после ее нажатия вкладка перезагружается, а кнопка «сохранить» исчезает. При изменении других элементов программа реагирует аналогичным образом.

При нажатии на кнопку «Пользователи» во вкладке появляется список всех зарегистрированных пользователей. Фамилия каждого пользователя выделена как ссылка. При ее нажатии открывается вкладка с подробными данными о пользователе.

При нажатии на кнопку «Зарегистрировать пользователя» открывается окно подробной информации о пользователе, но с пустыми значениями.

# ЗАКЛЮЧЕНИЕ

В результате работы над курсовой работой выполнены все поставленные задачи. Разработано рабочее приложение для учета студентов в общежитии. Был изучен ряд технологий при создании проекта: .NET, Windows Forms, C#, Visual Studio 2015. Также были изучены бизнес-процессы поселения студентов в общежитии. Написано обоснование выбора данных технологий в курсовом проекте. Функционал приложения был утвержден согласно разработанной структуре программы. На основе разработанной функциональной схемы проекта составлен программный продукт, обладающий интерфейсной частью и работой с файловой системой.

На данный момент при работе с приложением доступны следующие возможности: авторизация нового аккаунта, редактирование информации об аккаунте, разбиение аккаунтов по правам доступа, просмотр списка аккаунтов, просмотр справки.

Дальнейшее развитие системы связанно с расширением функционала, переносом хранимых данных в базу, внедрением. Тестирование программы показало ее работоспособность.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Draw.io – онлайн конструктор диаграмм:

URL: https://www.draw.io/ (дата обращения: 27.03.2017)

2. ProfessionalC# 5.0 and .NET 4.5 / Автор: НейгелК.,ИвьенБ. / Издательство - М.: Вильямс, Год: 2014.

3. Хейлсберг А. Язык программирования C#. Классика Computers Science: 4-е изд. / А.Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд.

4. Стиллмен Э. Изучаем C#: 3-е изд. / Э. Стиллмен, Дж. Грин/

5. DependencyInjectionin .NET Букинистическое издание / Автор: Марк Симан, А. Барышнев, Евгений Зазноба, издательство: Питер.

6. Wikipedia – информационный ресурс:

URL: https://ru.wikipedia.org/wiki/.NET\_Framework

(дата обращения: 06.04.2017).

URL: https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio

(дата обращения: 12.04.2017).

URL: https://ru.wikipedia.org/wiki/Windows\_Forms

(дата обращения: 13.04.2017).

# ПРИЛОЖЕНИЕ А

**Экранные формы**

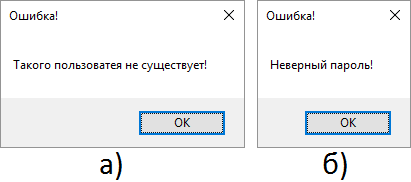


Рисунок А.1 – Модальные окна ошибок авторизации:

а – неверный логин;

б – неверный пароль

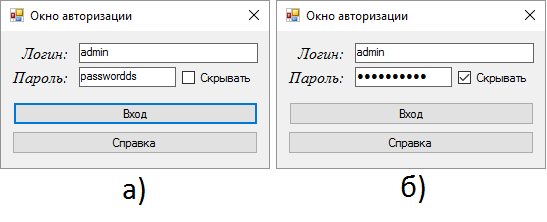


Рисунок А.2 – Вид окна авторизации:

а – видимый пароль;

б – скрытый пароль

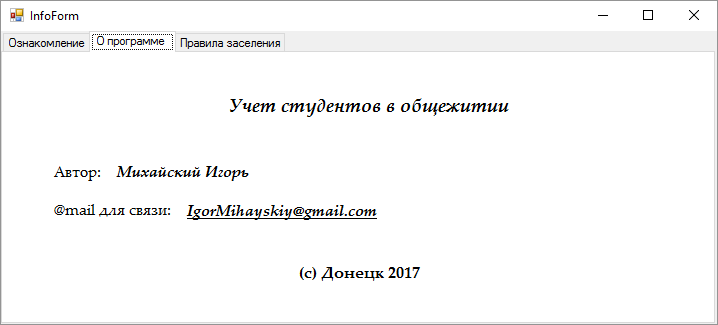


Рисунок А.3 – Вид окна справки

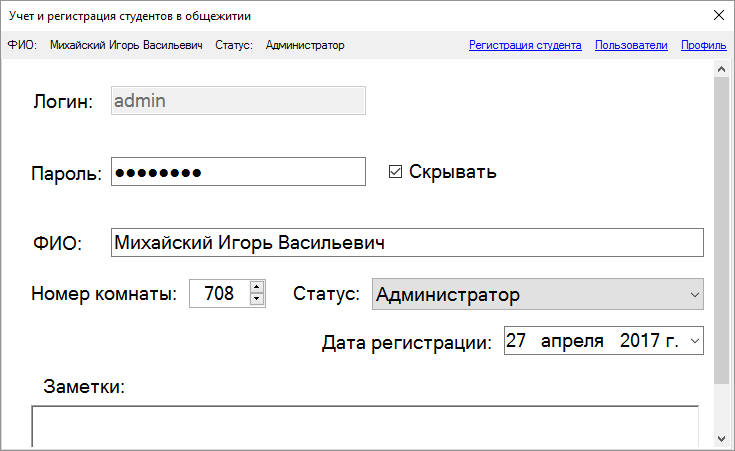


Рисунок А.4 – Вид вкладки данных пользователя от пользователя с правами администратора

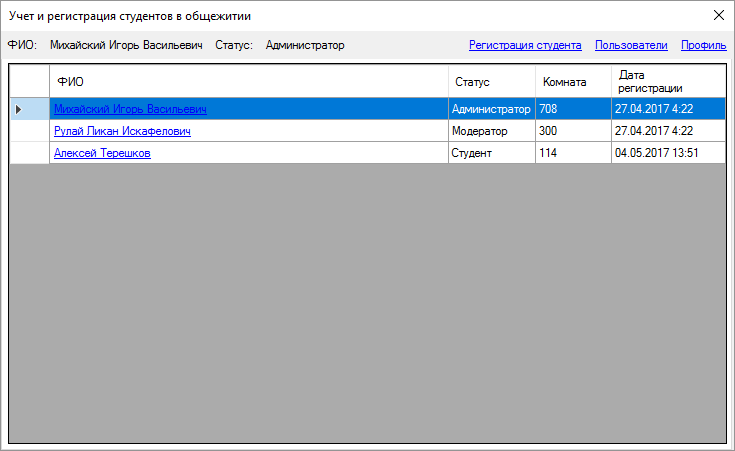


Рисунок А.5 – Вид окна со списком пользователей

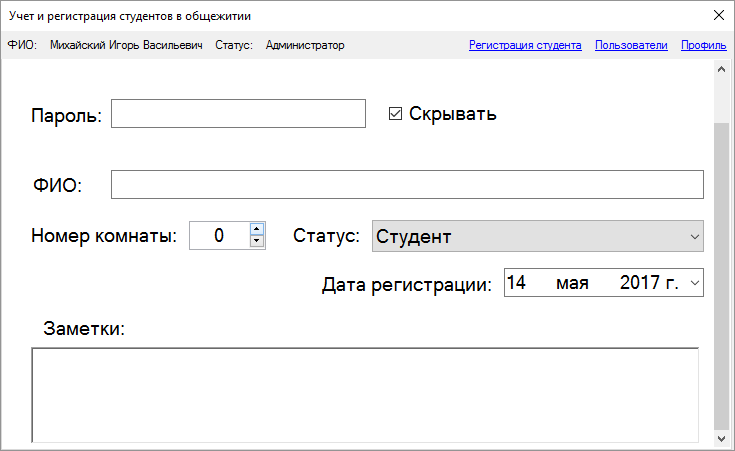


Рисунок А.6 – Вид окна при авторизации пользователя

# ПРИЛОЖЕНИЕ Б

**Фрагменты листинга**

## Листинг Б.1 – Файл MineForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ManagementOfTheStudentsHostel

{

public partial class MainForm : Form

{

public FileManager session;

public UserClass currentOpenUser;

public MainForm()

{

InitializeComponent();

}

private void MainForm\_Load(object sender, EventArgs e)

{

Hide();

LoginForm lf = new LoginForm();

DialogResult r = lf.ShowDialog(this);

switch (r)

{

case (DialogResult.Cancel):

{

Close();

break;

}

case (DialogResult.OK):

{

session = lf.fm;

FullNameLabel.Text = session.user.FullName;

AccesLvlLabel.Text = session.user.AccessLevelString;

if (session.user.AccessLevelString == "Администратор")

{

RegistrationBtn.Visible = true;

}

else

{

RegistrationBtn.Visible = false;

}

OpenAccount(session.user.Login);

Show();

break;

}

default:

{

MessageBox.Show(r.ToString(), "Неизвестный результат работы программы");

break;

}

}

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

OpenAccount(session.user.Login);

}

private void OpenAccount(string login)

{

if (login == "")

{

currentOpenUser = new UserClass("", "", "", 0, 0, DateTime.Now, "");

}

else

{

currentOpenUser = session.GetUser(login);

}

// заносим данные

FullNameTB.Text = currentOpenUser.FullName;

AccessLevelCB.SelectedIndex = currentOpenUser.AccessLevel;

RoomNB.Value = currentOpenUser.Room;

NotesRTB.Text = currentOpenUser.Notes;

dateTimePicker2.Value = currentOpenUser.CreateDate;

PasswordTB.Text = currentOpenUser.Password;

LoginTB.Text = currentOpenUser.Login;

// устанавливаем права

// панель с логином видно только

if (login == "")

{

LoginPanel.Visible = true;

}

else

{

LoginPanel.Visible = false;

}

// если админ или это профиль текущего пользователя, то можно показать пароль

if (session.user.AccessLevelString == "Администратор"

|| session.user.Login == currentOpenUser.Login)

{

PasswordPanel.Visible = true;

}

else

{

PasswordPanel.Visible = false;

}

// настройка панели с информацией

switch (session.user.AccessLevelString)

{

case "Администратор":

{

FullNameTB.Enabled = true;

AccessLevelCB.Enabled = true;

RoomNB.Enabled = true;

NotesRTB.Enabled = true;

dateTimePicker2.Enabled = true;

LoginPanel.Visible = true;

// если создаем новый акк, то включаем редактирование логина

if (login == "")

{

LoginTB.Enabled = true;

}else

{

LoginTB.Enabled = false;

}

}

break;

case "Модератор":

{

FullNameTB.Enabled = false;

AccessLevelCB.Enabled = false;

RoomNB.Enabled = true;

NotesRTB.Enabled = true;

dateTimePicker2.Enabled = false;

LoginPanel.Visible = false;

}

break;

case "Студент":

{

FullNameTB.Enabled = false;

AccessLevelCB.Enabled = false;

RoomNB.Enabled = false;

NotesRTB.Enabled = false;

dateTimePicker2.Enabled = false;

LoginPanel.Visible = false;

}

break;

default:

// если сюда дошли, то что-то не так

throw new ArgumentOutOfRangeException();

}

SaveChangesBtn.Visible = false;

tabControl1.SelectTab(1);}

private void SaveChangesBtn\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

bool newUser = currentOpenUser.Login == "";

currentOpenUser.AccessLevel = Helper.AccessLevelTextToInt(AccessLevelCB.Text);

currentOpenUser.FullName = FullNameTB.Text;

currentOpenUser.Room = Decimal.ToInt32(RoomNB.Value);

currentOpenUser.Notes = NotesRTB.Text;

currentOpenUser.CreateDate = dateTimePicker2.Value;

currentOpenUser.Login = LoginTB.Text;

currentOpenUser.Password = PasswordTB.Text;

try

{

if (newUser)

{ session.CreateUser(currentOpenUser); OpenAccount(currentOpenUser.Login);}

else

{session.ChangeUser(currentOpenUser); }

SaveChangesBtn.Visible = false;

}

catch (UserAlreadyExistException ex)

{MessageBox.Show(ex.Message, "Ошибка!");}

}

private void Account\_EditChanget(object sender, EventArgs e)

{

SaveChangesBtn.Visible = true;

}

private void linkLabel1\_LinkClicked\_1(object sender, LinkLabelLinkClickedEventArgs e)

{

RefreshUsersList();

tabControl1.SelectTab(0);

}

private void RefreshUsersList()

{

var source = new BindingSource();

List<UserClass> list = session.GetAllUsers();

source.DataSource = list;

dataGridView1.AutoGenerateColumns = false;

dataGridView1.DataSource = source;

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

var senderGrid = (DataGridView)sender;

if (senderGrid.Columns[e.ColumnIndex] is DataGridViewLinkColumn &&

e.RowIndex >= 0)

{

OpenAccount(senderGrid.Rows[e.RowIndex].Cells[senderGrid.Columns["Login"].Index].Value.ToString());

}

}

private void ShowPassord\_CheckedChanged(object sender, EventArgs e)

{

PasswordTB.UseSystemPasswordChar = ShowPassord.Checked;

}

private void RegistrationBtn\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

OpenAccount("");

}

}

}

## Листинг Б.2 – Фрагмент файла ExtendedBitmap.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ManagementOfTheStudentsHostel

{

public class FileManager

{

public UserClass user = new UserClass();

public FileManager() { }

public UserClass GetUser(string login)

{

// папка пользователя

string userFolder = Directory.GetCurrentDirectory() + "\\Users\\" + login;

// файл пользователя

string infoFilePath = userFolder + "\\info.txt";

UserClass userToReturn = new UserClass();

// проверка сущестования файла пользователя

if (!File.Exists(infoFilePath))

{

throw new LoginException("Такого пользоватея не существует!");

}

// файл есть, открываем его

using (StreamReader sr = File.OpenText(infoFilePath))

{

userToReturn.Login = login;

userToReturn.Password = sr.ReadLine();

userToReturn.FullName = sr.ReadLine();

userToReturn.Room = int.Parse(sr.ReadLine());

userToReturn.AccessLevel = int.Parse(sr.ReadLine());

userToReturn.CreateDate = DateTime.Parse(sr.ReadLine());

}

// файл c заметками о пользователе

string notesFilePath = userFolder + "\\notes.txt";

// пытаемся открыть его, иначе ничего страшного

if (File.Exists(notesFilePath))

{

using (StreamReader sr = File.OpenText(notesFilePath))

{

userToReturn.Notes = sr.ReadToEnd();

}

}

else

{

userToReturn.Notes = "";

}

return userToReturn;

}

public string GetTextInstr()

{

// файл c инструкцией для поселения

string InstructionsFilePath = Directory.GetCurrentDirectory() + "\\instructions.txt";

// пытаемся открыть его, иначе ничего страшного

if (File.Exists(InstructionsFilePath))

{

using (StreamReader sr = File.OpenText(InstructionsFilePath))

{

return sr.ReadToEnd();

}

}

else

{

return "";

}

}

public void Login(string login, string password)

{

// папка пользователя

string userFolder = Directory.GetCurrentDirectory() + "\\Users\\" + login;

// файл пользователя

string infoFilePath = userFolder + "\\info.txt";

// проверка сущестования файла пользователя

if (!File.Exists(infoFilePath))

{

throw new LoginException("Такого пользоватея не существует!");

}

// файл есть, открываем его

using (StreamReader sr = File.OpenText(infoFilePath))

{

// если пароли совпадают, то считываем файл дальше

if (password == sr.ReadLine())

{

user.Login = login;

user.Password = password;

user.FullName = sr.ReadLine();

user.Room = int.Parse(sr.ReadLine());

user.AccessLevel = int.Parse(sr.ReadLine());

user.CreateDate = DateTime.Parse(sr.ReadLine());

}

// иначе ошибка

else

{

throw new LoginException("Неверный пароль!");

}

}

// файл c заметками о пользователе

string notesFilePath = userFolder + "\\notes.txt";

// пытаемся открыть его, иначе ничего страшного

if (File.Exists(notesFilePath))

{

using (StreamReader sr = File.OpenText(notesFilePath))

{

user.Notes = sr.ReadToEnd();

}

}

else

{

user.Notes = "";

}

}

public void CreateUser(UserClass newUser)

{

// папка c информацией пользователя (если он уже есть)

string userFolder = Directory.GetCurrentDirectory() + "\\Users\\" + newUser.Login;

// проверка сущестования файла пользователя

if (File.Exists(userFolder))

{

throw new UserAlreadyExistException("Пользователь с логином "+ newUser.Login+" уже существует!");

}

else

{

Directory.CreateDirectory(userFolder);

ChangeUser(newUser);

}

}

public void ChangeUser(UserClass newUser)

{

// текущая папка проекта

string startPath = Directory.GetCurrentDirectory();

// файл c информацией пользователя

string infoFilePath = startPath + "\\Users\\" + newUser.Login + "\\info.txt";

// открываем его

using (StreamWriter sw = File.CreateText(infoFilePath))

{

sw.WriteLine(newUser.Password);

sw.WriteLine(newUser.FullName);

sw.WriteLine(newUser.Room);

sw.WriteLine(newUser.AccessLevel);

sw.WriteLine(newUser.CreateDate);

}

// файл c заметками о пользователе

string notesFilePath = startPath + "\\Users\\" + newUser.Login + "\\notes.txt";

// открываем его

using (StreamWriter sw = File.CreateText(notesFilePath))

{

sw.Write(newUser.Notes);

}

}

public List<UserClass> GetAllUsers()

{

List<UserClass> userstToReturn = new List<UserClass>();

// получение всех пользователей в директории

// текущая папка проекта

string usersFolderPath = Directory.GetCurrentDirectory() + "\\Users\\";

string[] usersFolders = Directory.GetDirectories(usersFolderPath);

foreach (string folder in usersFolders)

{

// folderShortName == login

string folderShortName = Path.GetFileName(folder);

userstToReturn.Add(GetUser(folderShortName));

}

return userstToReturn;

}

}

}

## Листинг Б.3 – Фрагмент файла UserClass.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ManagementOfTheStudentsHostel

{

public class UserClass

{

public string Login { get; set; }

public string Password { get; set; }

public string FullName { get; set; }

public int Room { get; set; }

public int AccessLevel { get; set; }

public string AccessLevelString {

// Только для чтения

get {return Helper.AccessLevelIntToText(AccessLevel);}

}

public DateTime CreateDate { get; set; }

public string Notes { get; set; }

public UserClass() { }

public UserClass(string login,string password, string fullName, int room, int accessLevel, DateTime createDate, string notes) { Login = login;

Password = password;

FullName = fullName;

Room = room;

AccessLevel = accessLevel;

CreateDate = createDate;

Notes = notes;

}

}

}