МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-47 01 02 «Дизайн электронных и веб-изданий»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированное программирование»

Тема Программное средство «Общежитие»

Исполнитель

Студента 2 курса группы 12 Худницкий Д. А.

(Ф.И.О.)

Руководитель работы Марченко А. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

# СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Введение | 3 |
| 1. Постановка задачи | 4 |
| * 1. Формулировка проблемы и ее решения | 4 |
| * 1. Обзор прототипов | 4 |
| 1. Разработка архитектуры проекта | 6 |
| * 1. Проектирование структуры приложения и UML-диаграммы | 6 |
| * 1. Проектирование дизайна приложения | 7 |
| * 1. Схема базы данных | 8 |
| 1. Разработка приложения | 10 |
| * 1. Описание функций приложения | 10 |
| * 1. Реализация работы с базой данных | 10 |
| * 1. Основные окна приложения | 12 |
| * + 1. Окно авторизации | 12 |
| * + 1. Основное окно | 13 |
| * + 1. Страницы отображения модели и коллекции данных | 14 |
| * + 1. Окно регистрации нового пользователя | 16 |
| 1. Руководство пользователя | 17 |
| 1. Тестирование | 24 |
| Заключение | 26 |
| Список использованных источников | 27 |
| Приложение 1 | 28 |
| Приложение 2 | 29 |
| Приложение 3 | 30 |
| Приложение 4 | 31 |

# ВВЕДЕНИЕ

Разработка приложения происходила в программной среде Microsoft Visual Studio 2019 на платформе WPF — это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность.

Данные для работы будут храниться в базе данных. Её создание происходит с помощью Microsoft SQL Server — это система управления реляционными базами данных, которая подключается к приложению.

Для работы с самим SQL Server используется Entity Framework, который представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Он позволяет работать напрямую с объектами, данные для которых хранятся в таблицах базы данных.

# Постановка задачи

Основной задачей курсового проекта является разработка приложения, позволяющего упросить работу сотрудников общежития с жильцами.

Функционально ПС должно выполнять следующие задачи:

* расчёт показателей потребления коммунальных услуг;
* учёт записей вызова работников общежития;
* хранение учётных данных жильцов и сотрудников;

## Формулировка проблемы и алгоритма ее решения

Приложение представляет из себя общежитие — место временного проживания для приезжих студентов и учащихся на время учёбы, сезонных рабочих на время работы, службы или ставшее постоянным для работников учреждения. Соответственно, следует различать студенческие и рабочие общежития. Данное приложение реализует возможности для студенческих общежитий. Следовательно, приложение должно осуществлять решение вопросов, связанных с заселением, выселением, а также давать жильцу всю необходимую информацию.

Основной функционал приложения имеет следующий вид:

* заселение нового жильца в общежитие;
* регистрация сотрудников общежития;
* авторизация уже существующих пользователей приложения;
* просмотр списка жильцов и сотрудников;
* поиск и фильтрация жильцов и сотрудников;
* просмотр информации о комнате;
* просмотр записей, оставленные сотрудниками общежития;
* просмотр показателей потребления коммунальных услуг;

## Обзор прототипов

В качестве объекта обзора и анализа взято программное средство «Общежитие» (Рисунок 1.1), которое представляет из себя десктопное приложение, позволяющие ввести учёт жильцов студенческого общежития.

Программное средство позволяет создавать, изменять и удалять записи в базе данных. Предоставляет доступ к просмотру жильцов выбранной комнаты (Рисунок 1.2).Также, позволяет искать в базе данных по заданному критерию. Разработано на платформе .Net Framework с использование WPF.

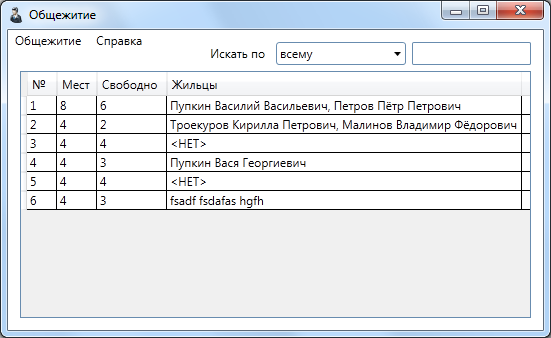


Рисунок 1.1 — Основное окно ПС «Общежитие»

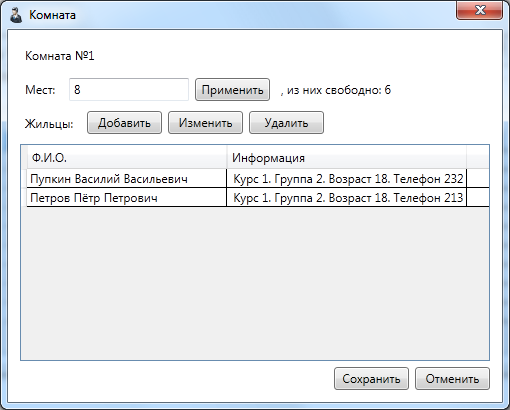


Рисунок 1.2 — Окно просмотра комнаты ПС «Общежитие»

# Разработка архитектуры проекта

## Проектирование структуры приложения и UML-диаграммы

Решение представлено одним проектом «DormitoryApp», имеющим структуру, представленную на рисунке:

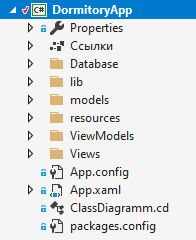


Рисунок 2.1 — Структура проекта

**App.config**

Файл конфигурации приложения, который находится в корневой папке приложения.

**App.xaml**

Файл, в котором определены ресурсы, общие для приложения.

**lib**

Папка, в которой содержатся так называемые «технические» файлы, необходимые для корректной работы приложения.

**resources**

Папка, в которой содержатся изображения, файлы-локализации, и шрифты, которые используются в приложении, а также файлы, содержащие стили и шаблоны для элементов управления соответственно.

**models**

Папка, в которой содержатся модели данных, как производные от моделей базы данных, так и непроизводные.

**ViewModels**

Папка, в которой содержатся модели представления, которые связывают модель и представление через механизм привязки данных.

**Views**

Папка, в которой содержатся представления определяющие визуальный интерфейс, через который пользователь взаимодействует с приложением.

**Database**

Папка, в которой содержатся файлы связанные с базой данных, например: файл модели данных, файл связей таблицы и дриугие.

**ClassDiagram.cd**

Диаграмма классов

На данной UML-диаграмме (Рисунок 2.2) изображены классы программы, методы, поля и свойства классов.

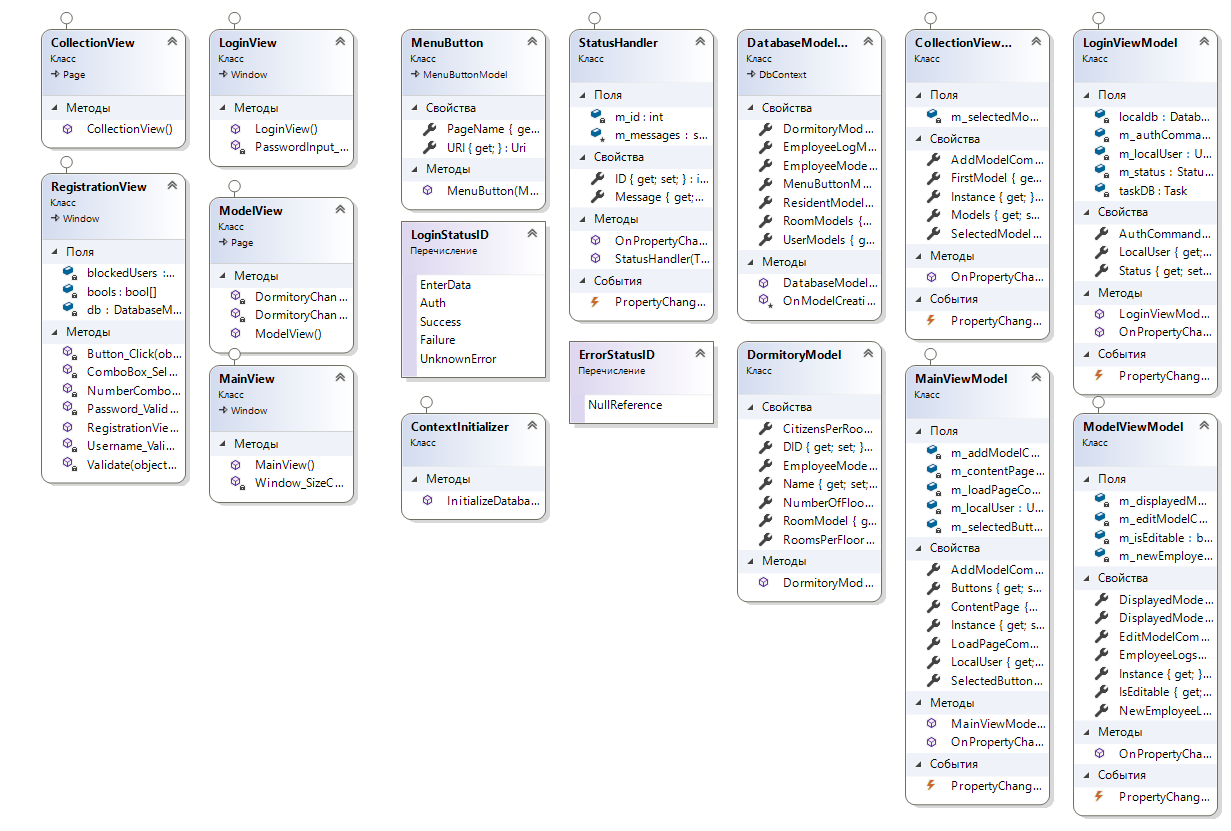


Рисунок 2.2 — Диаграмма классов

## Проектирование дизайна приложения

При разработке дизайна приложения главной целью было сделать его соответствующим тематике. Для этого в качестве основных были выбраны нейтральные пастельные (Рисунок 2.4). Основным направлением в дизайне был выбран тренд Glassmorphism.



Рисунок 2.4 – Цвета приложения.

## Схема базы данных

Для реализации поставленной для приложения задачи была создана база данных «dormitorydb». Для ее создания использовался подход Model First фреймворка Entity Framework. Данный подход подразумевает, что сначала делается модель базы данных, а на основе её создаётся сама база данных и модели данных. Для взаимодействия базы данных с приложением был также использован Entity Framework, который предоставляет функционал для работы с базами данных.

База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Система управления базами данных (СУБД) — это комплекс программных средств, необходимых для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации.

В данном приложении база данных состоит из 9 таблиц, и ее схема изображена на рисунке 2.5.

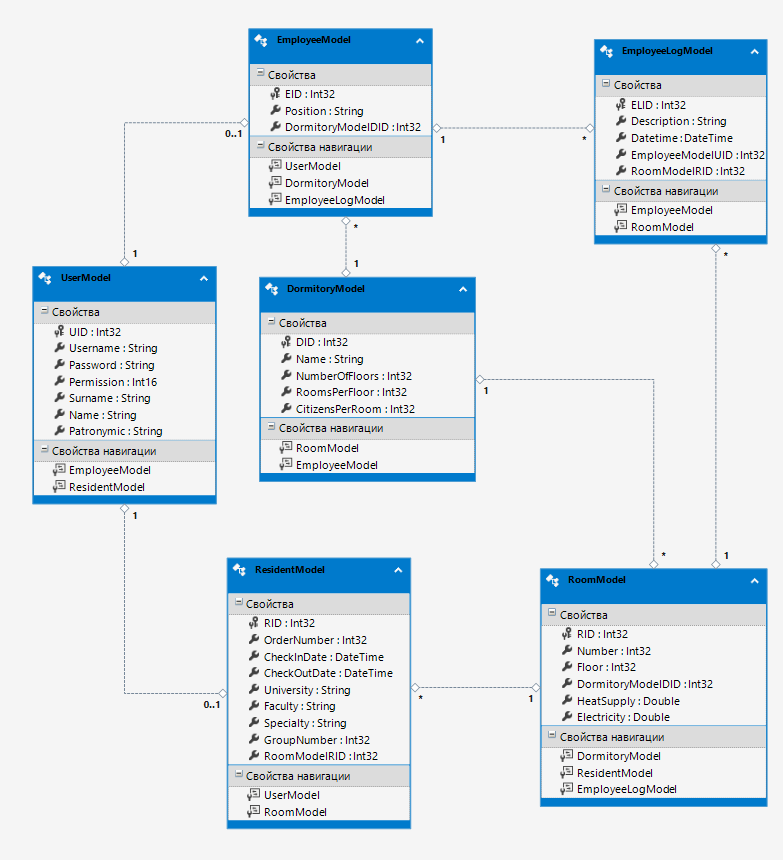


Рисунок 2.5 — Схема базы данных

Для хранения данных пользователей используется таблица UserModel с полями:

* UID — уникальный идентификатор пользователя (первичный ключ);
* Username — имя пользователя;
* Password — пароль от аккаунта;
* Permission —уровень прав пользователя;
* Surname — фамилия пользователя;
* Name — имя пользователя;
* Patronymic — отчество пользователя;

Для хранения информации о жильцах используется таблица ResidentModel c полями:

* RID — уникальный идентификатор жильца (первичный ключ);
* OrderNumber — номер ордера;
* CheckInDate — дата заселения;
* CheckOutDate — дата выселения;
* University — наименование университета;
* Faculty — наименование факультета;
* Specialty — наименование специальности;
* GroupNumber — номер группы;
* RoomModelID — уникальный идентификатор комнаты (внешний ключ RID в таблице RoomModel);

Для хранения информации о сотрудниках используется таблица EmployeeModel с полями:

* EID — уникальный идентификатор сотрудника (первичный ключ);
* Position — должность сотрудника;
* DormitoryModelDID — уникальный идентификатор (внешний ключ на DID в таблице Dormitory);

Для хранения информации о комнатах используется таблица RoomModel с полями:

* RID— уникальный идентификатор комнаты (первичный ключ);
* Floor — этаж комнаты;
* Number — номер комнаты;
* DormitoryModelDID — уникальный идентификатор (внешний ключ на DID в таблице Dormitory);
* HeatSupple — итоговый показать за отопление;
* Electricity — итоговый показатель за электричество;

Для хранения информации об общежитиях используется таблица DormitoryModel с полями:

* DID — уникальный идентификатор общежития (первичный ключ);
* Name — наименование общежития;
* NumberOfFloors — количество этажей в общежитии;
* RoomsPerFloor — количество комнат на этаже;
* CitizensPerRoom — количество жильцов в комнате;

Для хранения информации об записях сотрудников используется таблица EmployeeLog с полями:

* ELID — уникальный идентификатор записи (первичный ключ);
* Description — описание записи;
* Datetime — время внесения записи;
* RoomModelRID — уникальный идентификатор комнаты (внешний ключ на RID в таблице RoomModel);
* EmployeeModelUID — уникальный идентификатор сотрудника (внешний ключ на EID в таблице EmployeeModel)

Все эти таблицы соответствуют одноименным классам в проекте, которые были созданы с помощью Entity Framework. Классы содержат такие же поля, как и таблицы и используются для работы с базой данных.

# Разработка приложения

## Описание функций приложения

Данное приложение представляет собой ритуальное агентство и выполняет следующие функции:

* регистрация и авторизация пользователей;
* возможность смены профиля;
* регистрация общежитий и сотрудников;
* предоставление списка жильцов или сотрудников;
* предоставление списка комнат и общежитий;
* предоставление списка записей каждой комнаты;
* предоставление профиля пользователей, комнат;
* предоставление показателей коммунальных услуг;

## Реализация работы с базой данных

Для взаимодействия базы данных с приложением был использован Entity Framework, который предоставляет функционал для работы с базами данных. Для подключения к базе данных dormitory в файл App.config была добавлена строка подключения. Так как использовался подход Model First, то, при создании модели базы данных, данная строка появилась сама (Рисунок 3.1). Также для первичной настройки базы данных была добавлена строка с настройками приложения.

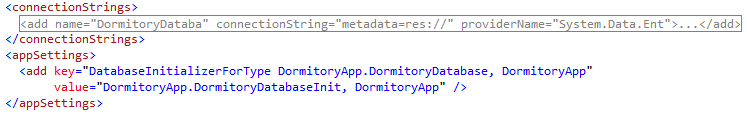


Рисунок 3.1 — Настройка подключения базы данных

При подходе Model First IDE автоматически генерирует модели данных, и на их основе создаёт классы. (Рисунок 3.2).

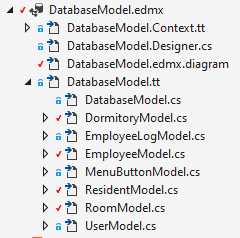


Рисунок 3.2 — Файлы созданные с помощью метода Model First

В этих классах содержатся свойства (Рисунок 3.3) с такими же именами и типами, как у полей соответствующих им таблиц (Рисунок 3.4).

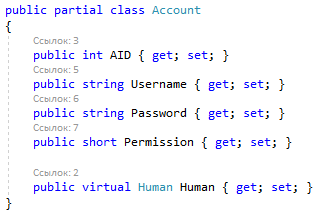


Рисунок 3.3 — Класс Account

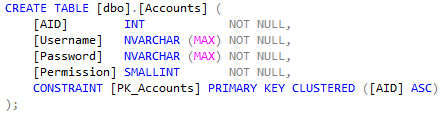


Рисунок 3.4 — Таблица Account

## Основные окна приложения

Приложение построено по такому принципу, что все дополнительные окна вызываются из главного, путем нажатия на соответствующие кнопки. Во всех окнах отключен стандартный стиль Windows-приложения.

### **Окно авторизации**

Окно авторизации пользователя появляется первым при запуске приложения. В нем пользователь должен имя пользователя и пароль. После нажатия на кнопку «Войти» срабатывает команда «AuthCommand» (Рисунок 3.5), сравнивает введенные пользователем данные с данными в базе данных и, в случае совпадения, открывает следующее окно программы. Если пользователь с таким именем пользователя и паролем не содержится в базе данных, то выводится сообщение о том, что пользователь не найден.

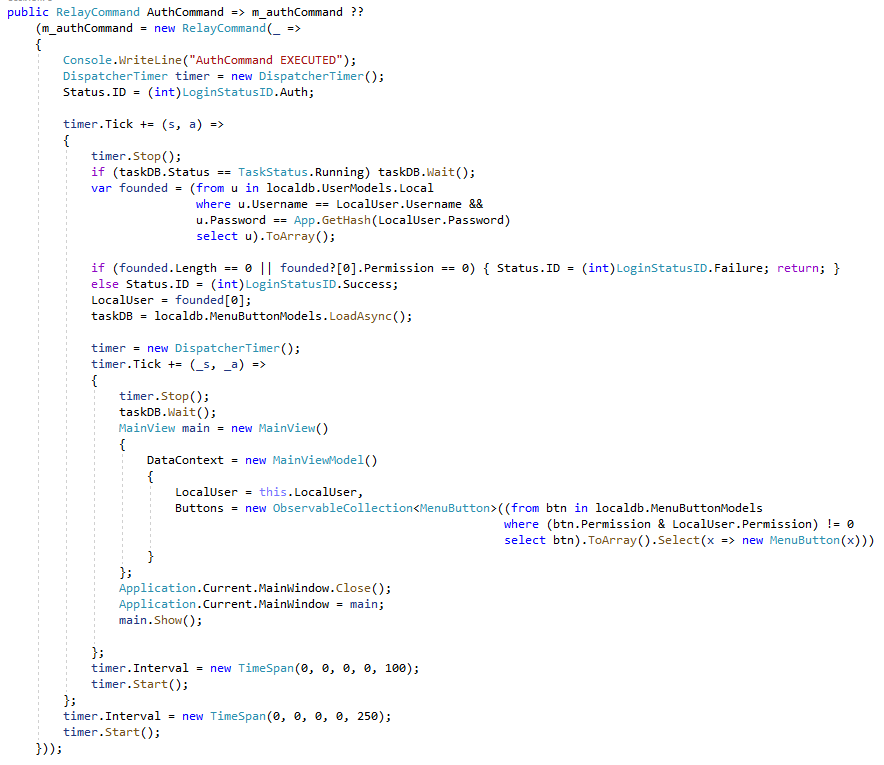


Рисунок 3.5 — Комманда «AuthCommand»

### **Основное окно**

В основном окне пользователь может выбирать необходимую ему вкладку. После соответствующая страница будет подгружена в основное окно и доступно для взаимодействия. Загрузка страницы происходит благодаря механизму привязки WPF на свойство «ContentPage». За изменение содержимого свойства отвечает команда «LoadPageCommand» (Рисунок 3.6).

Также, для обращения к главному окну используется шаблон проектирования «Singleton». Данный шаблон предоставляется доступ к единственному экземпляру класса, и гарантирует, что данный экземпляр будет один.

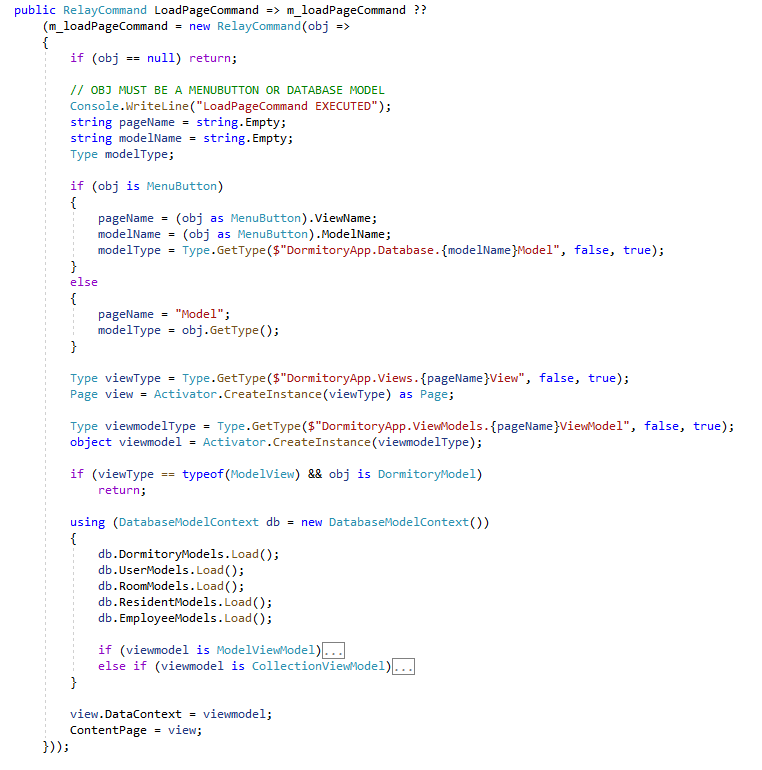


Рисунок 3.6 — Команда «LoadPageCommand»

В приложении реализован шаблон проектирования MVVM. Данный шаблон позволяет распределить разработку приложения на компоненты: View — компонент представления, отвечает за отображения модели данных; ViewModel — компонент-«связка», обрабатывает полученные данные и передаёт их компоненту View; Model — модели данных, полученные из какого-либо источника, например — база данных.

Компоненты View поделены на два основных типа: для отображения одной сущности и для отображения коллекции сущностей. Соответственно для каждого View был создан свой ViewModel. Для загрузки View создана ранее упомянутая команда «LoadPageCommand», а для загрузки данных используется функционал Entity Framework и «LINQ to Entities» для фильтрации данных.

Команда «AddModelCommand» отвечает за добавление модели данных в свою таблицу в базу данных (Рисунок 3.7)



Рисунок 3.7 — Команда «AddModelCommand»

### **Страницы отображения модели и коллекции данных**

Обе страницы сделаны по одному и тому же принципу. Благодаря функционалу XMAL, в частности DataTemplate — View определяет, что за тип данных ему передали. И в соответствии с типом отображает модели данных (Рисуноки 3.7–3.8).



Рисунок 3.7 — Код внешнего вида страницы «ModelView»



Рисунок 3.8 — Код внешнего вида страницы «CollectionView»

### **Окно регистрации нового пользователя**

Под регистрацией нового пользователя подразумевается либо заселение нового жильца в общежитие, либо добавление сотрудника общежития в программу.

За регистрацию нового пользователя отвечает ранее упомянутая команда «AddModelCommand». А за ввод данных окно — «RegistrationView» (Рисунок 3.9)



Рисунок 3.9— Код внешнего вида окна «RegistrationView»

# Руководство пользователя

Сразу после запуска приложения пользователь попадает в окно авторизации, где ему необходимо ввести своё имя пользователя и пароль для входа в систему и доступа ко всему функционалу (Рисунок 4.1). В случае неудачной авторизации приложение оповестит пользователя о том, что авторизация безуспешна.

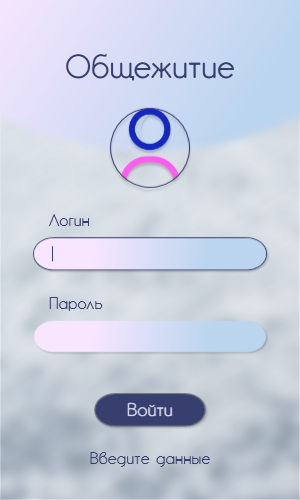


Рисунок 4.1 — Окно авторизации

После успешной авторизации появляется основное окно приложения, (Рисунок 4.2). В данном окне, а в частности в правой его части в дальнейшем буду загружаться соответствующие страницы.



Рисунок 4.2 — Основное окно

В левой части окна расположено меню, оно содержит вкладки доступные данному пользователю. То есть, в зависимости от уровня доступа пользователя, его набор вкладок меню будет изменяться соответствующим образом. Весь набор вкладок меню: «Домашняя страница», «Общежития», «Комнаты», «Жильцы», «Сотрудники».

При нажатии на вкладку «Домашняя страница» в основное окно подгружается страница с основной информацией об пользователе. Если данный пользователь является администратором приложения, то ему будет предложен выбор загрузить соответствующий профиль (Рисунок 4.3–4.5). Также у администратора есть возможность изменить общежитие, к которому он принадлежит.

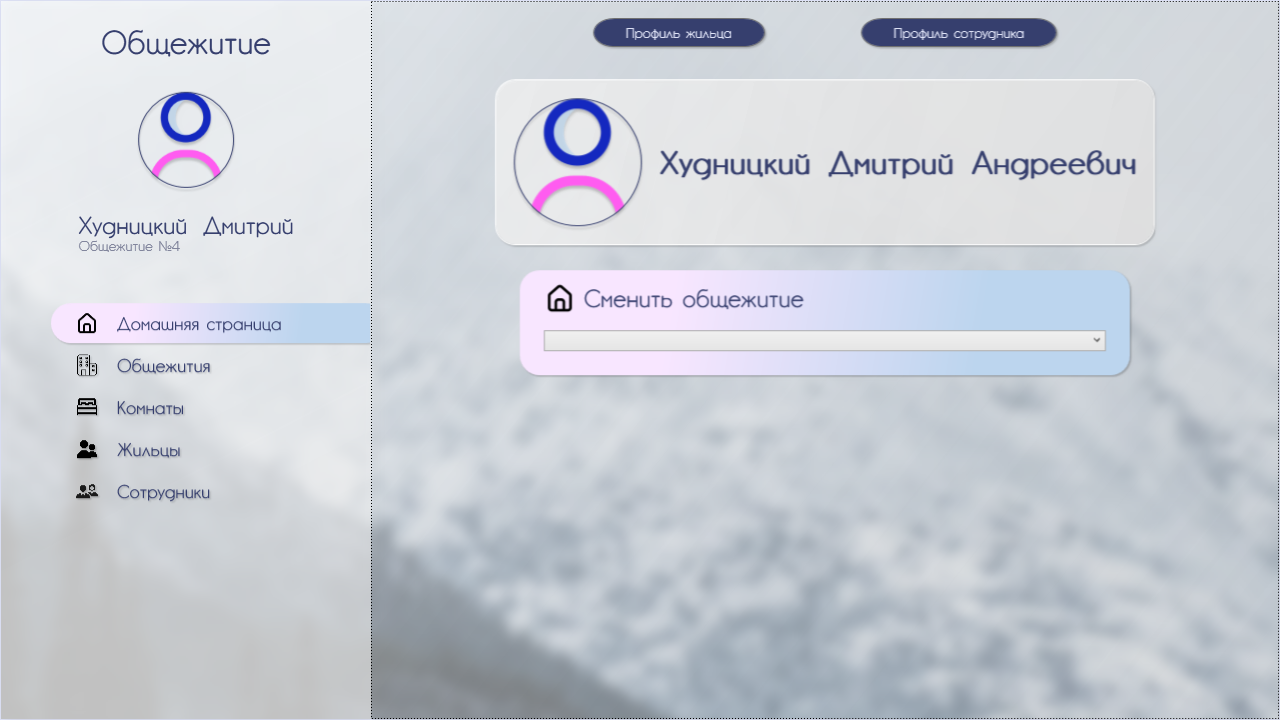


Рисунок 4.3 — Страница профиля администратора

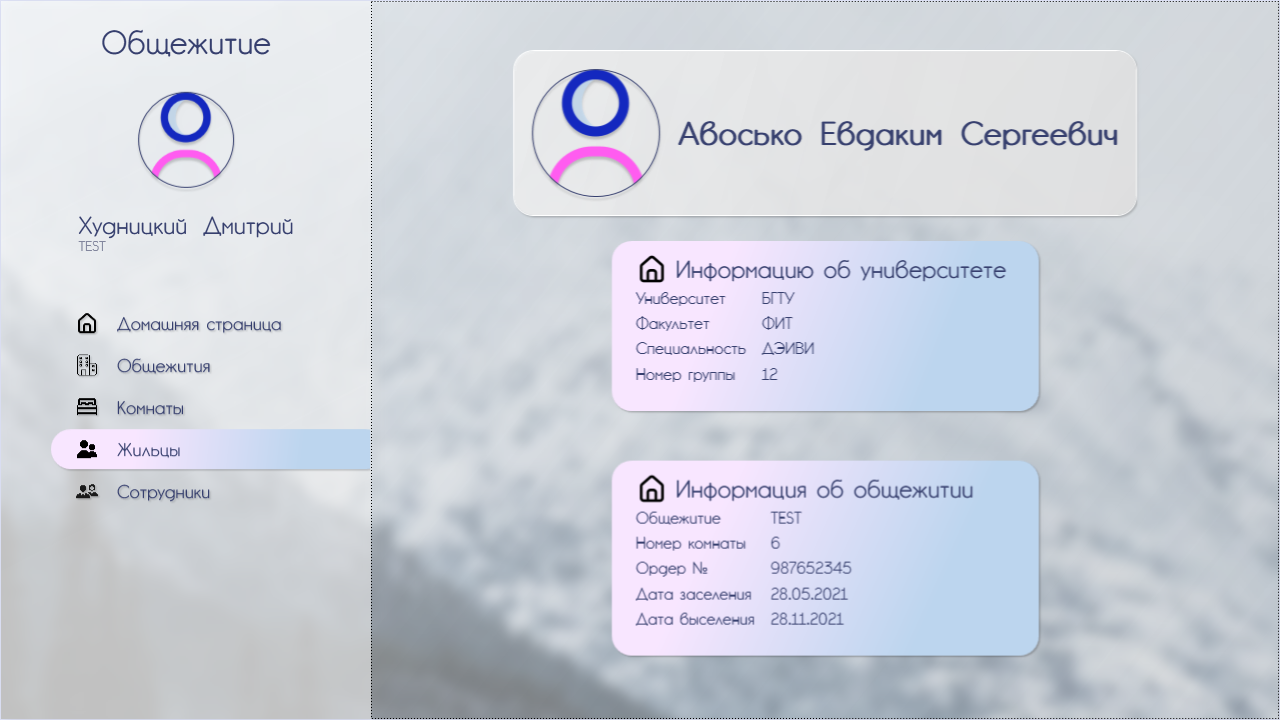


Рисунок 4.4 — Страница профиля жителя

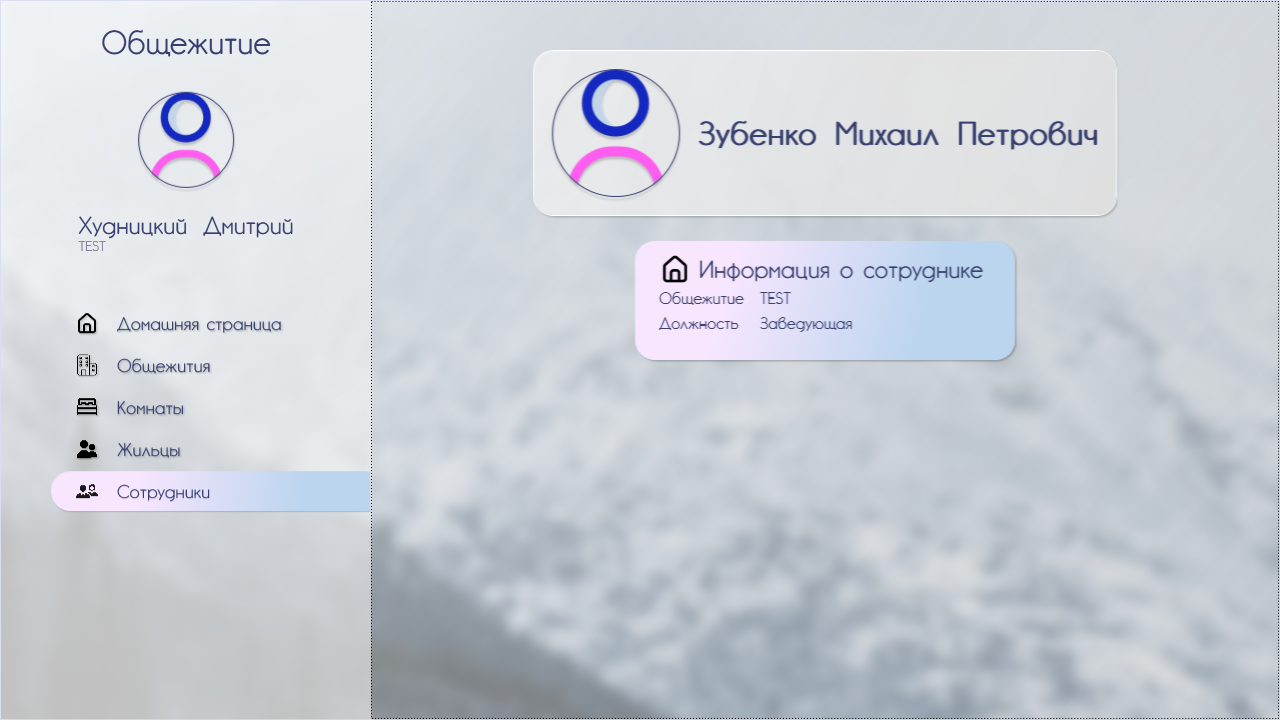


Рисунок 4.5 — Страница профиля сотрудника

Вкладка «Общежития» содержит список общежитий, хранимых в базе данных приложения. Также, доступна форма регистрации нового общежития в приложение, но только для администратора приложения (Рисунок 4.6).

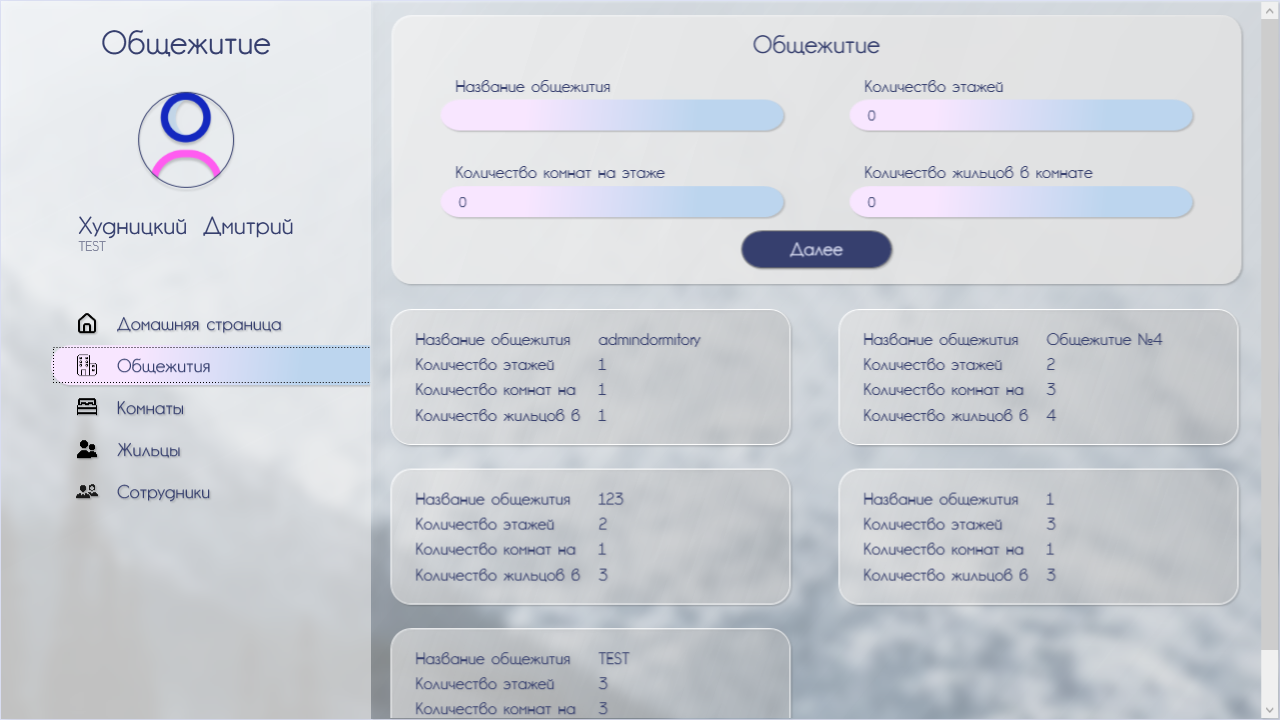


Рисунок 4.6 — Список общежитий

Вкладка «Комнаты» содержит список комнат доступных в данном общежитии (Рисунок 4.7). После выбора нужной комнаты подгружается информация о данной комнате (Рисунок 4.8). В «профиле» комнаты содержится список записей сотрудников, а также расчёт показателей коммунальных услуг за определённый период и поля для ввода записи сотрудника общежития.

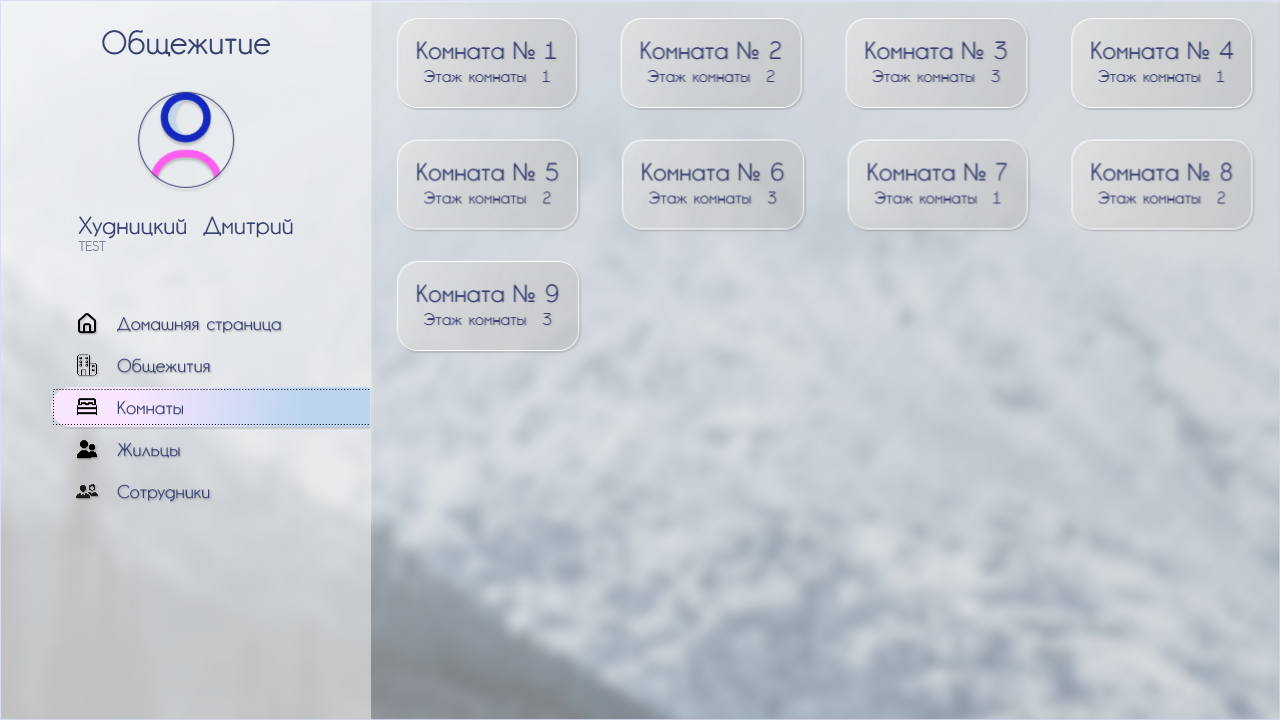


Рисунок 4.7 — Список комнат

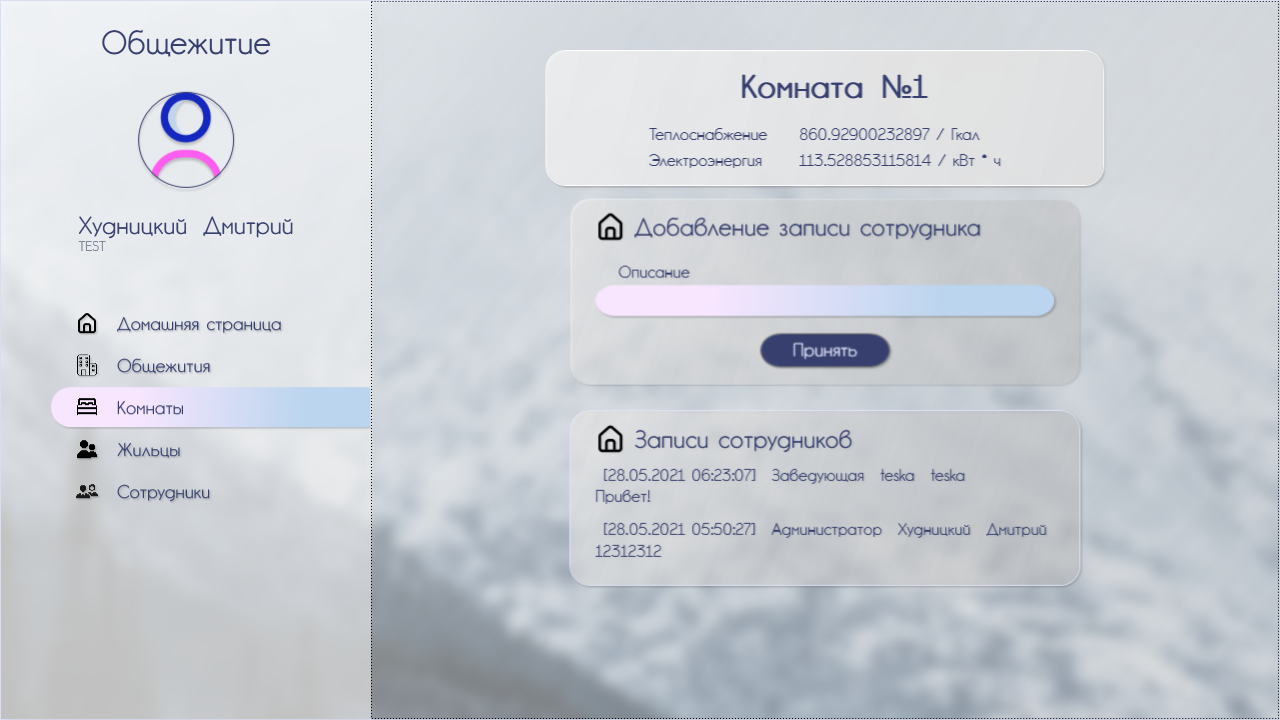


Рисунок 4.8 — «Профиль» комнаты

Вкладка «Жильцы» содержит список жильцов доступных в данном общежитии (Рисунок 4.9). После выбора жильца подгружается страница с информацией о данном жильце

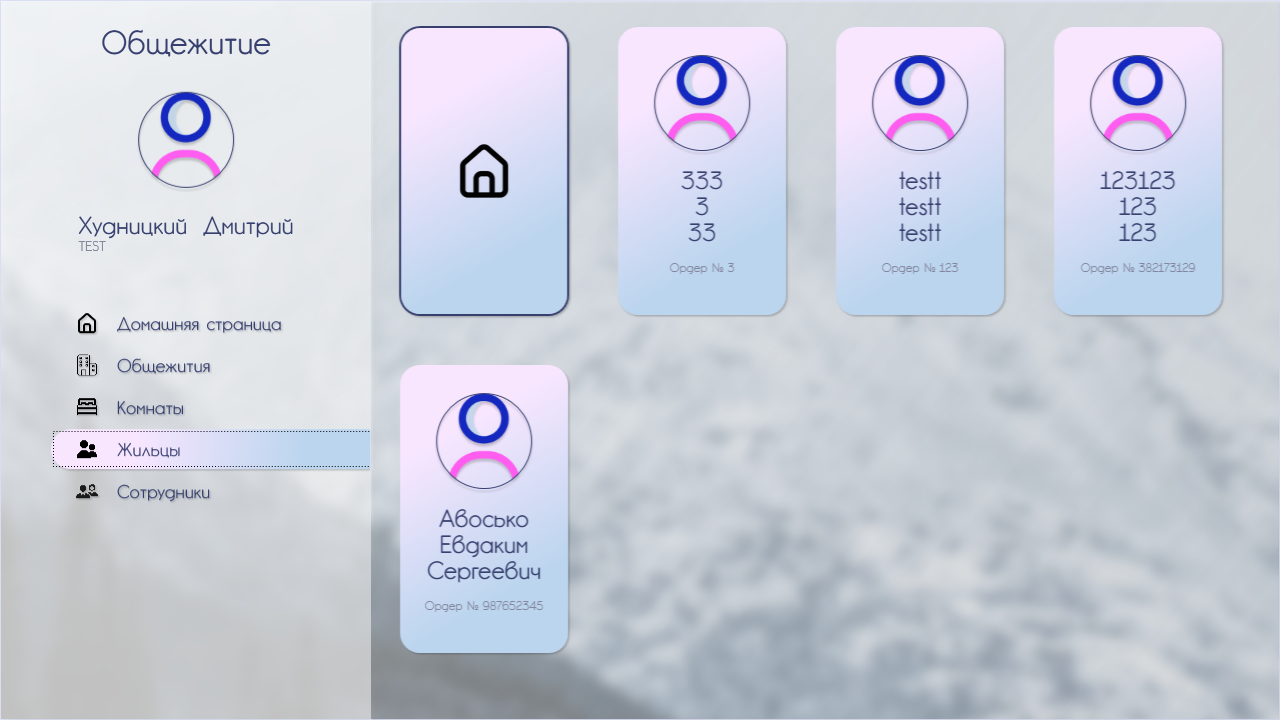


Рисунок 4.9 — Список жильцов

Вкладка «Сотрудники» содержит список сотрудников доступных в данном общежитии, и работает по тому же самому принципу, что и вкладка жильцы, но для сотрудников (Рисунок 4.10)

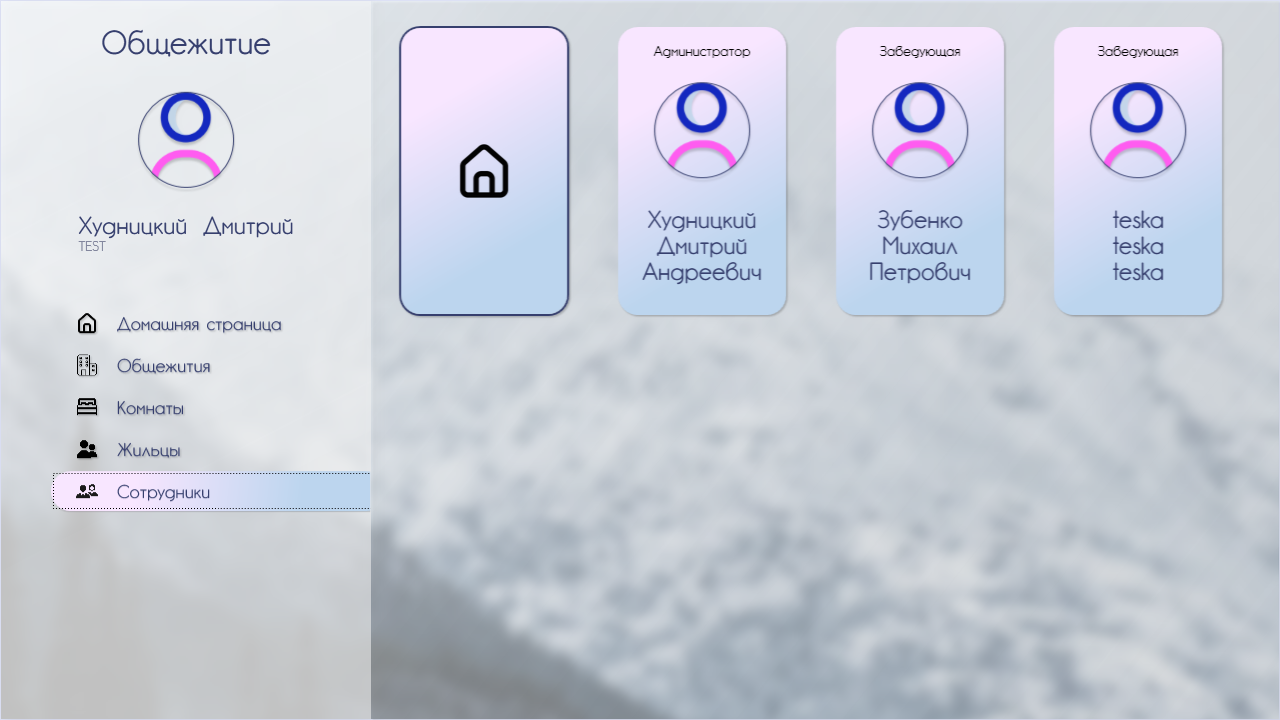


Рисунок 4.10 — Список жильцов

При выборе пустой ячейки на странице списка жильцов или сотрудников — появляется окно ввода данных для соответствующей сущности (Рисунок 4.11–4.12)

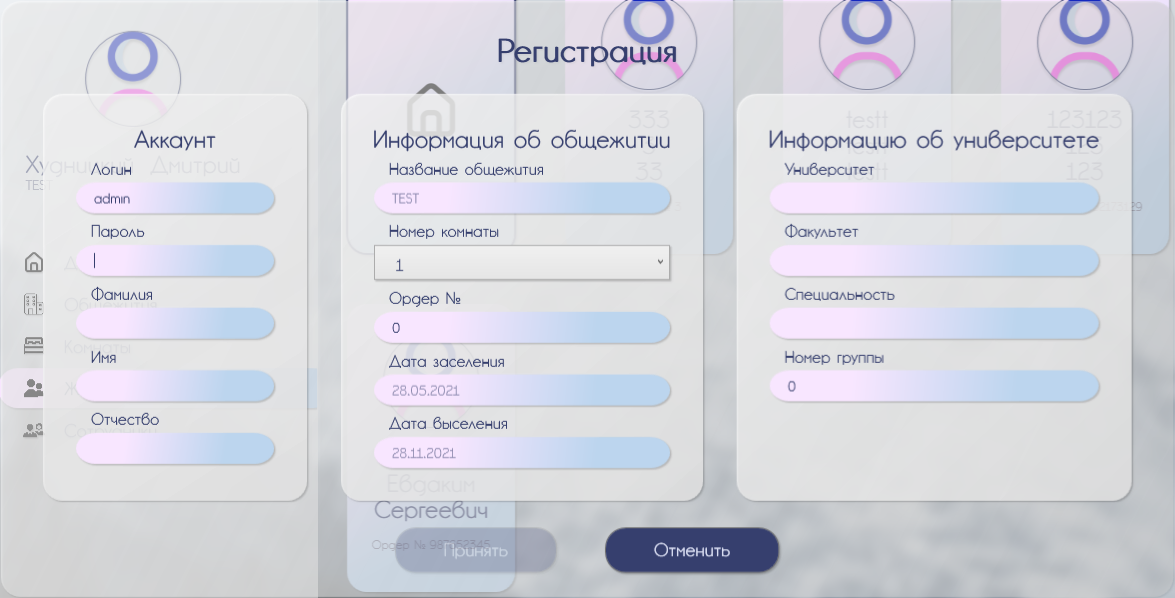


Рисунок 4.10 — Регистрация жильца

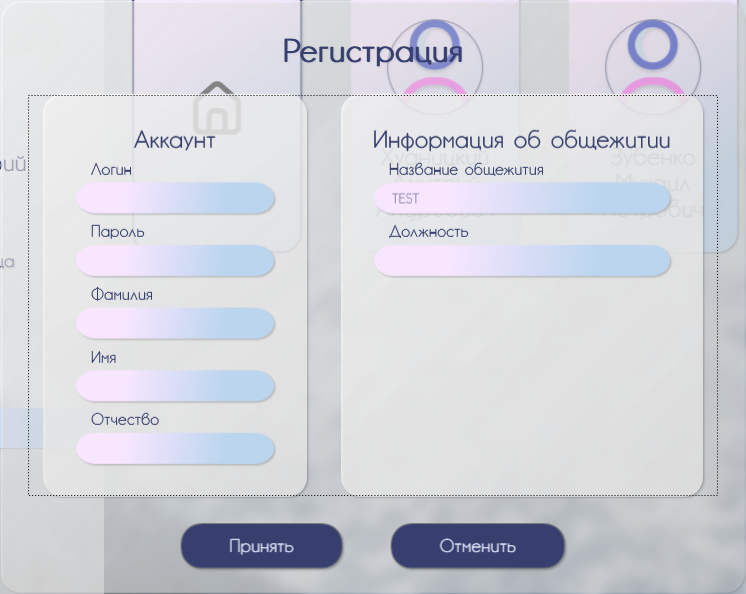


Рисунок 4.11 — Регистрация сотрудника

# Тестирование

Для обеспечения корректной работоспособности приложения необходимо обрабатывать возникающие ошибки. Для этого можно использовать различные средства, например, валидацию. Так как приложение работает с базой данных, то некорректные значения могут вызвать непредвиденные трудности с его работоспособностью, а также важно, чтобы данные не противоречили их логическому назначению. В данном приложении валидация выполняется при помощи регулярных выражений.

Для предотвращения регистрация пользователей программа проверяет введённое имя пользователя на наличие его в базе данных (Рисунок 5.1). Также запрещено оставлять поле «Пароль» пустым.

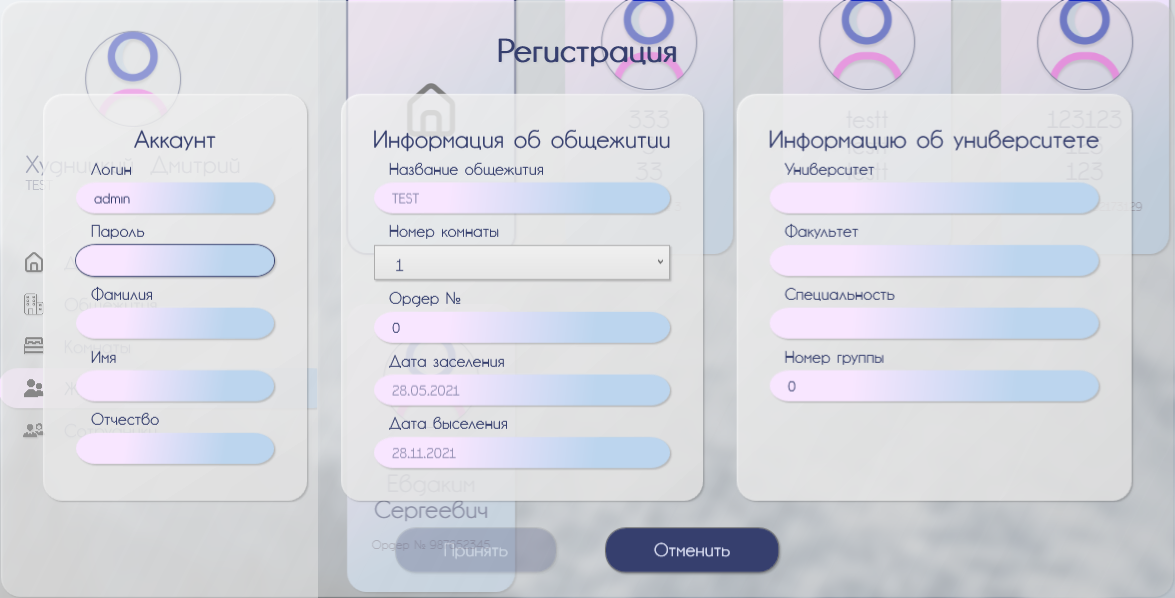


Рисунок 5.1 — Валидация имени пользователя и пароля

Для корректного ввода номера комнаты приложение предлагает сотруднику выбрать номер из выпадающего списка (Рисунок 5.2)

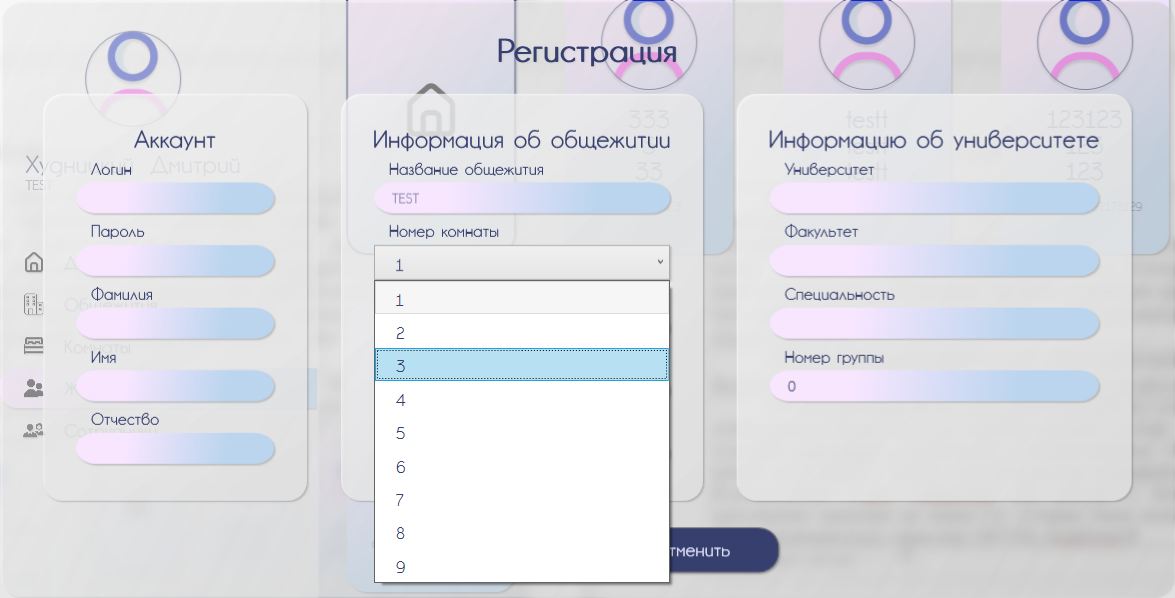


Рисунок 5.2 — Выпадающий список

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта было разработано приложение «Общежитие», содержащее в себе процессы работы с заселением жильцов общежития, просмотр показателей коммунальных услуг каждой комнаты, список жильцов и сотрудников каждого зарегистрированного общежития. По ходу разработки проекта была создана база данных для хранения информации, разработан интерфейс и написан код для приложения.

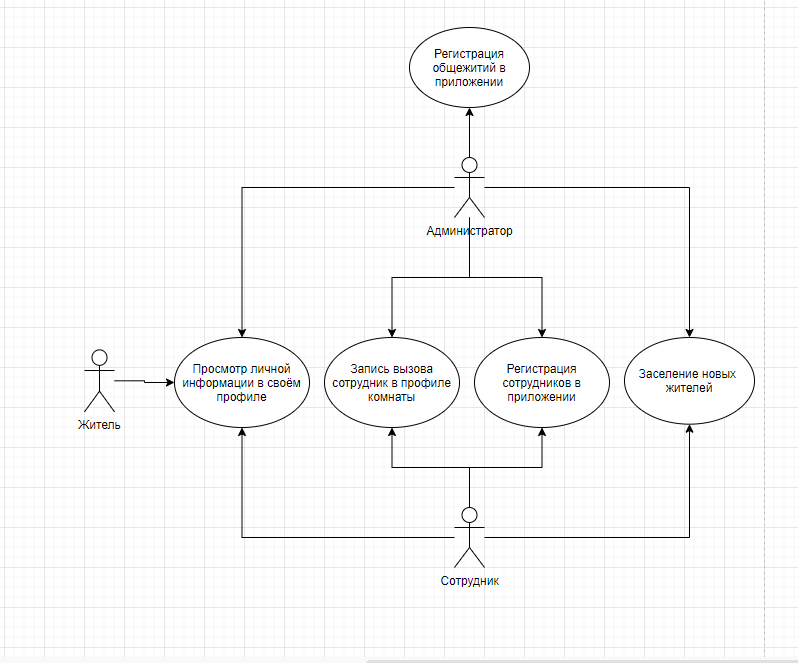
Требования для курсового проекта были выполнены в полном объеме. Были реализованы основные задачи, прописанные в листе заданий.

За период выполнения курсового проекта по созданию приложения «Общежитие» были изучены способы работы с «SQL Server», принципы создания декстопного приложения с использованием .NET и WPF, изучен новый язык верстки XAML для построения интерфейса и работы с ним. Использовался Entity Framework для работы с базой данный. Само приложение написано на языке С#. А также были использованы шаблоны проектирования кода, такие как: MVVM, Singletone.

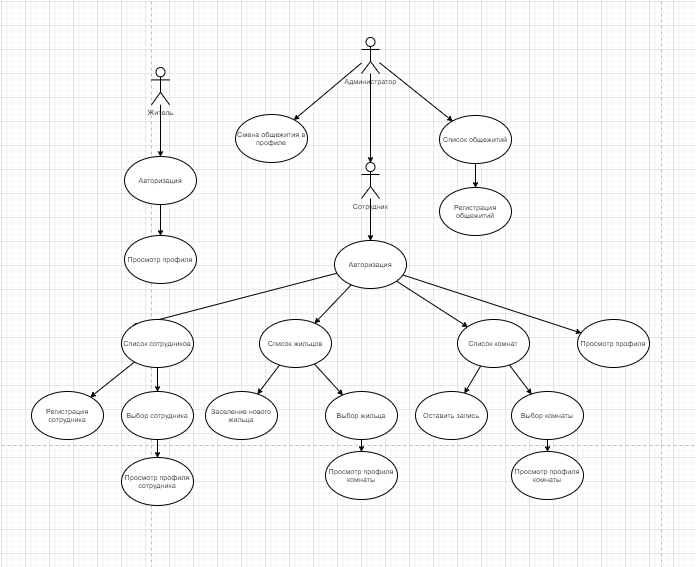
# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Википедия [Электронный ресурс] / Режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio>
2. Полное руководство по C#8 [Электронный ресурс] / Режим доступа: <https://metanit.com/sharp/tutorial/1.1>\_
3. Руководство по WPF [Электронный ресурс] / Режим доступа: <https://metanit.com/sharp/wpf/1>
4. Руководство по WPF [Электронный ресурс] / Режим доступа: <https://metanit.com/sharp/wpf/2>
5. Что такое material design [Электронный ресурс] / Режим доступа: <https://darksiteofmarketing.com/stati/chto-takoe-material-design.html>
6. Руководство по Entity Framework [Электронный ресурс] / Режим доступа: <https://metanit.com/sharp/entityframework/1.1>

# ПРИЛОЖЕНИЕ 1



# ПРИЛОЖЕНИЕ 2



# ПРИЛОЖЕНИЕ 3



# ПРИЛОЖЕНИЕ 4

