**Цель:**

Разработка приложения электронной телефонной книги

**Исходные данные:**

* База данных с таблицами:
  + Users
  + Telephone\_numb
  + Telephone\_users
* Текущие требования и функциональность приложения.

**Модельные представления:**

Создание связей между таблицами для обеспечения целостности данных. Разработка схемы базы данных для оптимального хранения информации. Реализация механизма добавления, удаления, редактирования контента в приложении. Разработка функционала управления контентом пользователя. Интеграция существующего приложения с базой данных.

**Ожидаемый результат:**

Разработанное приложение, интегрированное с базой данных на языке *C#,* для персональных компьютеров, на базе операционной системы *Windows*.  
Графический интерфейс. Возможность добавлять/удалять/редактировать информацию связанную с телефонной справочной информацией.

**Критерии оценки результата:**

* Функциональность приложения.
* Интеграция приложения с базой данных без ошибок.

**Средства реализации:**

* *Visual* *Studio* 2022 для создания приложения.
* *WPF* для разработки графического интерфейса.
* Язык программирования *C#* для написания логики приложения.
* Библиотека *NpgSql* для взаимодействия с базой данных *PostgreSQL*.

CREATE OR REPLACE FUNCTION track\_telephonenumb\_changes()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

-- Записываем вставленные данные в таблицу ARCHIVE

INSERT INTO telephonenumb\_archive (N\_Z ,Numb\_Country,Numb\_City,Numb,action,change\_date )

VALUES (new.n\_z ,new.numb\_country,new.numb\_city,new.numb, 'Insert', CURRENT\_TIMESTAMP);

ELSIF TG\_OP = 'UPDATE' THEN

-- Записываем исходные данные перед обновлением в таблицу ARCHIVE

INSERT INTO telephonenumb\_archive (N\_Z ,Numb\_Country,Numb\_City,Numb, action, change\_date)

VALUES (old.n\_z ,old.numb\_country,old.numb\_city,old.numb, 'Update (Original)', CURRENT\_TIMESTAMP);

-- Записываем измененные данные после обновления в таблицу ARCHIVE

INSERT INTO telephonenumb\_archive (N\_Z ,Numb\_Country,Numb\_City,Numb, action, change\_date)

VALUES (new.n\_z ,new.numb\_country,new.numb\_city,new.numb, 'Update (New)', CURRENT\_TIMESTAMP);

ELSIF TG\_OP = 'DELETE' THEN

-- Записываем удаленные данные в таблицу ARCHIVE

INSERT INTO telephonenumb\_archive (N\_Z ,Numb\_Country,Numb\_City,Numb, action, change\_date)

VALUES (old.n\_z ,old.numb\_country,old.numb\_city,old.numb, 'Delete', CURRENT\_TIMESTAMP);

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

Этот код представляет собой функцию триггера на языке PL/pgSQL для отслеживания изменений в таблице telephonenumb и записи этих изменений в таблицу telephonenumb\_archive.

Давайте разберемся с основными моментами кода:

CREATE OR REPLACE FUNCTION track\_telephonenumb\_changes(): Эта строка определяет функцию триггера с именем track\_telephonenumb\_changes.

RETURNS TRIGGER: Указывает, что функция триггера возвращает тип TRIGGER.

BEGIN...END;: Блок кода, который содержит логику триггера.

IF TG\_OP = 'INSERT' THEN...ELSIF TG\_OP = 'UPDATE' THEN...ELSIF TG\_OP = 'DELETE' THEN: Это условные операторы, проверяющие тип операции триггера (INSERT, UPDATE или DELETE).

IF TG\_OP = 'INSERT' THEN: Если происходит вставка данных, то их исходные значения (new) записываются в таблицу telephonenumb\_archive с меткой 'Insert' и текущей временной меткой.

ELSIF TG\_OP = 'UPDATE' THEN: Если данные обновляются, то исходные значения (old) записываются с меткой 'Update (Original)', а новые значения (new) с меткой 'Update (New)'.

ELSIF TG\_OP = 'DELETE' THEN: Если данные удаляются, то старые значения (old) записываются с меткой 'Delete'.

RETURN NULL;: В конце каждого условия, триггер возвращает NULL, так как эта функция триггера не должна возвращать результат.

$$ LANGUAGE plpgsql;: Эта строка указывает язык функции триггера (PL/pgSQL).

Таким образом, функция track\_telephonenumb\_changes служит для отслеживания операций INSERT, UPDATE и DELETE в таблице telephonenumb и записи соответствующих данных в таблицу telephonenumb\_archive.

public void quarry(string s, string[] param)

{

NpgsqlDataReader reader1;

NpgsqlCommand cmd = new NpgsqlCommand();

cmd.Connection = conn;

cmd.CommandText = s;

for (int i = 0; i < param.Length; i++)

{

cmd.Parameters.AddWithValue($"@value{i+1}", param[i]);

}

reader1 = cmd.ExecuteReader();

reader = reader1;

}

Этот код представляет собой метод quarry (возможно, было бы лучше использовать правильное название query), который выполняет SQL-запрос к базе данных с использованием параметров. Давайте разберемся с основными моментами кода:

public void quarry(string s, string[] param): Объявление метода с именем quarry, который принимает два параметра: строку s для SQL-запроса и массив строк param для параметров.

NpgsqlDataReader reader1;: Объявление объекта NpgsqlDataReader с именем reader1, который будет использоваться для чтения результатов запроса.

NpgsqlCommand cmd = new NpgsqlCommand();: Создание нового объекта NpgsqlCommand с именем cmd, который представляет собой команду SQL для выполнения на базе данных.

cmd.Connection = conn;: Установка соединения для команды. Предполагается, что conn - это ранее инициализированный объект NpgsqlConnection, представляющий соединение с базой данных.

cmd.CommandText = s;: Установка текста команды, который представляет собой SQL-запрос.

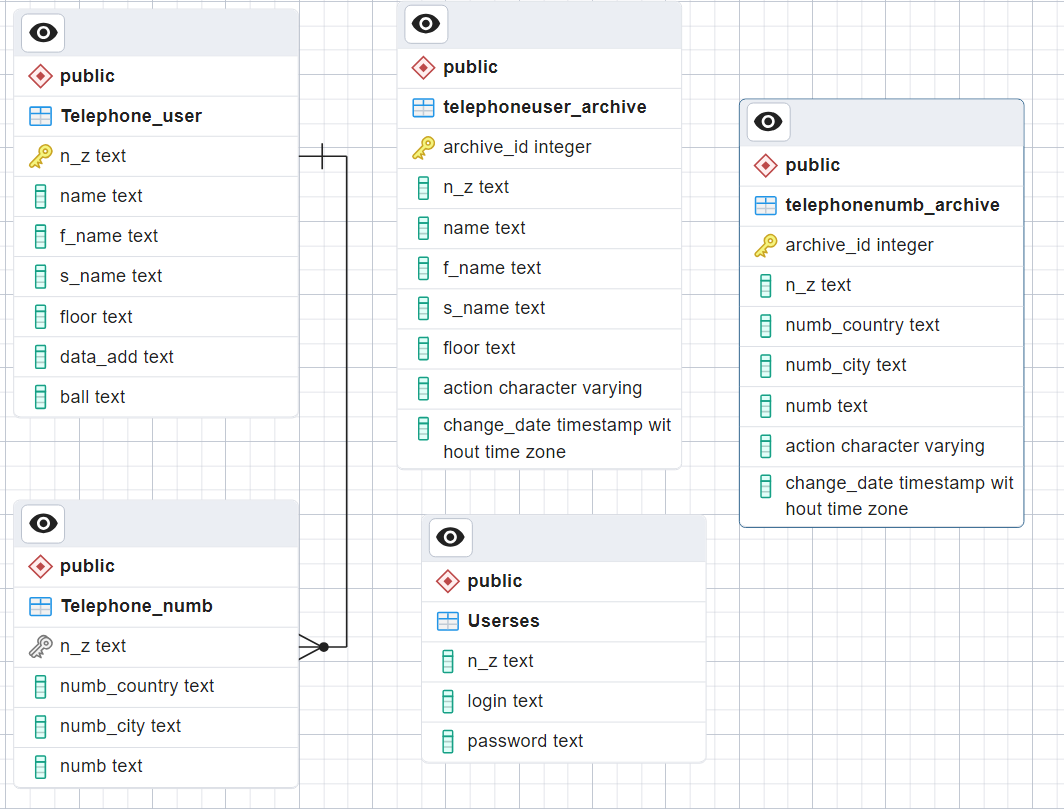
for (int i = 0; i < param.Length; i++): Цикл по параметрам, где каждый параметр добавляется в команду с использованием cmd.Parameters.AddWithValue.

cmd.Parameters.AddWithValue($"@value{i+1}", param[i]);: Добавление параметра с именем вида @value1, @value2, и так далее, в команду. Значения параметров берутся из массива param.

reader1 = cmd.ExecuteReader();: Выполнение команды и получение объекта NpgsqlDataReader для чтения результатов запроса.

reader = reader1;: Предположительно, результаты запроса сохраняются в глобальной переменной reader для дальнейшего использования в коде.

Этот метод позволяет выполнять SQL-запросы к базе данных с использованием параметров, что важно для предотвращения SQL-инъекций.



База данных включает в себя четыре основные таблицы: *users*, *contents*, *videoinfo* и *contents\_archive*.

**Таблица "*userses*":**

Столбцы:

* + Логин (*login*)
  + Номер записи (n\_z)
  + Пароль (*password*)

**Таблица "Telephone\_*numb*":**

Столбцы:

* + Номер записи (*n\_z*)
  + Номер страны (*numb\_country*)
  + Код города(*numb\_city)*
  + *Номер телефона (numb)*

**Таблица "*Telephone\_users*":**

Столбцы:

* + Номер записи (*n\_z*)
  + Имя (*name*)
  + Фамилия (f\_*name*)
  + Отчество (s\_*name*)
  + Пол(floor)
  + Дата добавления (data\_add)
  + Оценка (ball)

**Таблица " Telephone\_*numb \_archive*":**

Столбцы:

* + Номер записи (*n\_z*)
  + Номер страны (*numb\_country*)
  + Код города(*numb\_city)*
  + *Номер телефона (numb)*
  + Действие (*action*) - операция удаления/добавления/изменения
  + Дата изменения (*change\_date*) - дата и время последнего изменения

**Таблица " *Telephone\_users \_archive*":**

Столбцы:

* + Номер записи (*n\_z*)
  + Имя (*name*)
  + Фамилия (f\_*name*)
  + Отчество (s\_*name*)
  + Пол(floor)
  + Действие (*action*) - операция удаления/добавления/изменения
  + Дата изменения (*change\_date*) - дата и время последнего изменения

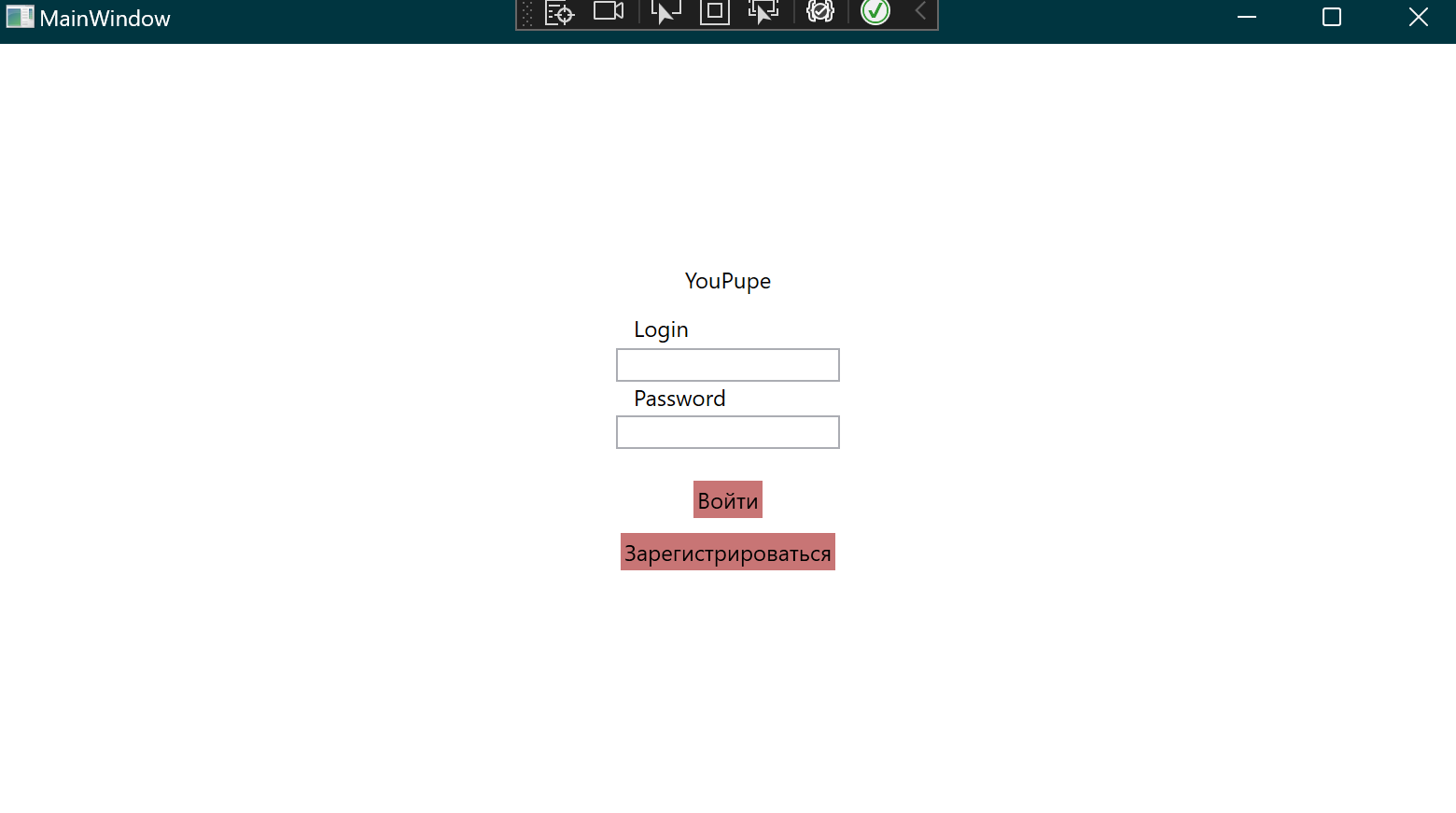


Рисунок 1. Окно авторизации.

Окно авторизации содержит поля для ввода логина и пароля, а также кнопки "Войти" и "Зарегистрироваться". При корректном вводе пароля и логина:

* Кнопка "Войти" перенаправляет на окно главного экрана.
* Если введенные данные неверны, выводится уведомление "Неверный логин или пароль."

Кнопка "Зарегистрироваться" перенаправляет на окно регистрации, где можно добавить нового пользователя.

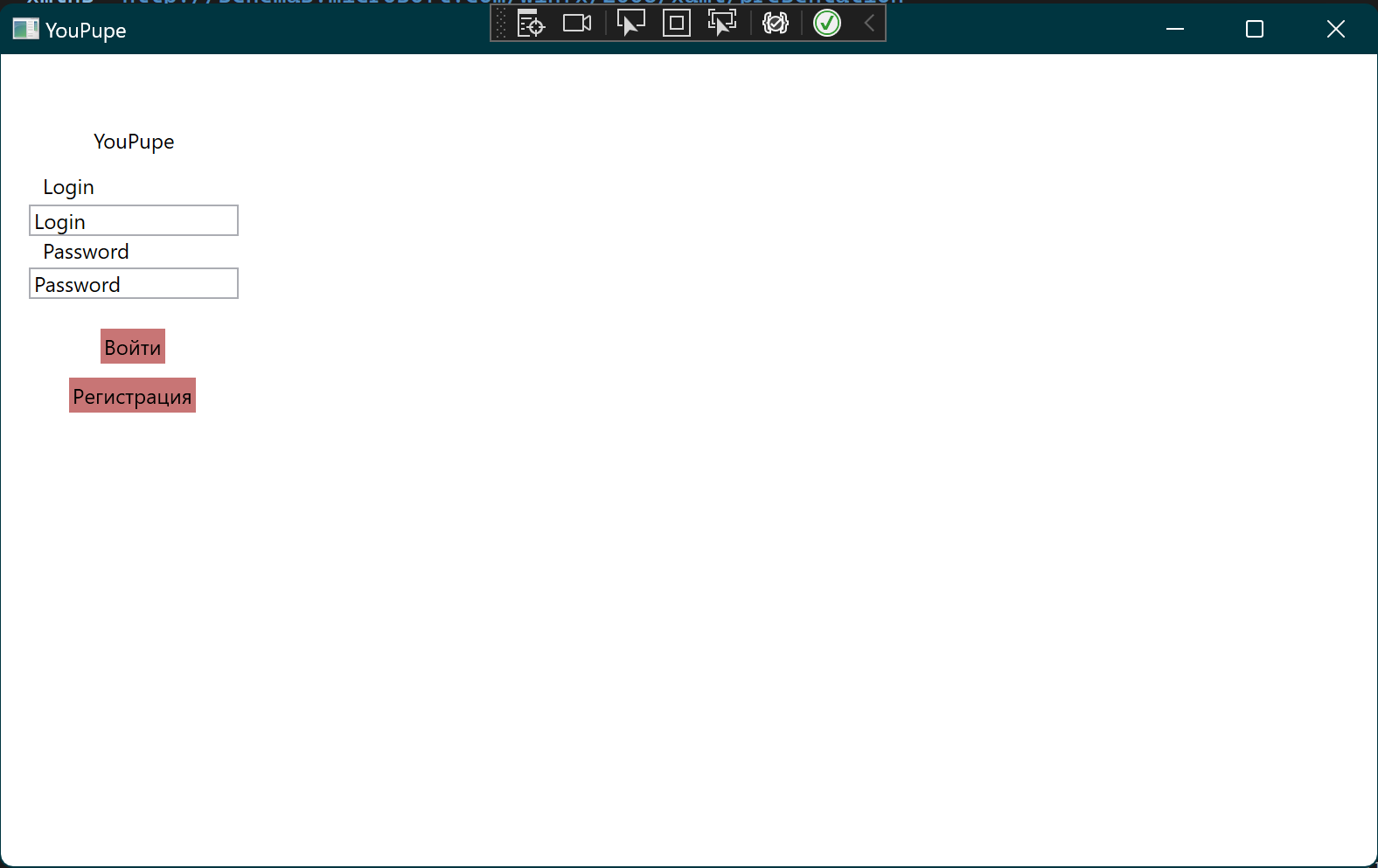


Рисунок 2. Окно регистрации.

Окно регистрации содержит поля для ввода логина и пароля, а также кнопки "Войти" и "Зарегистрироваться". При корректном вводе пароля и логина:

* Кнопка "Войти" перенаправляет на окно главного экрана.
* Если введенные данные неверны, выводится уведомление "Неверный логин или пароль."

Кнопка "Зарегистрироваться" добавит нового пользователя и перенаправит на окно главного экрана.

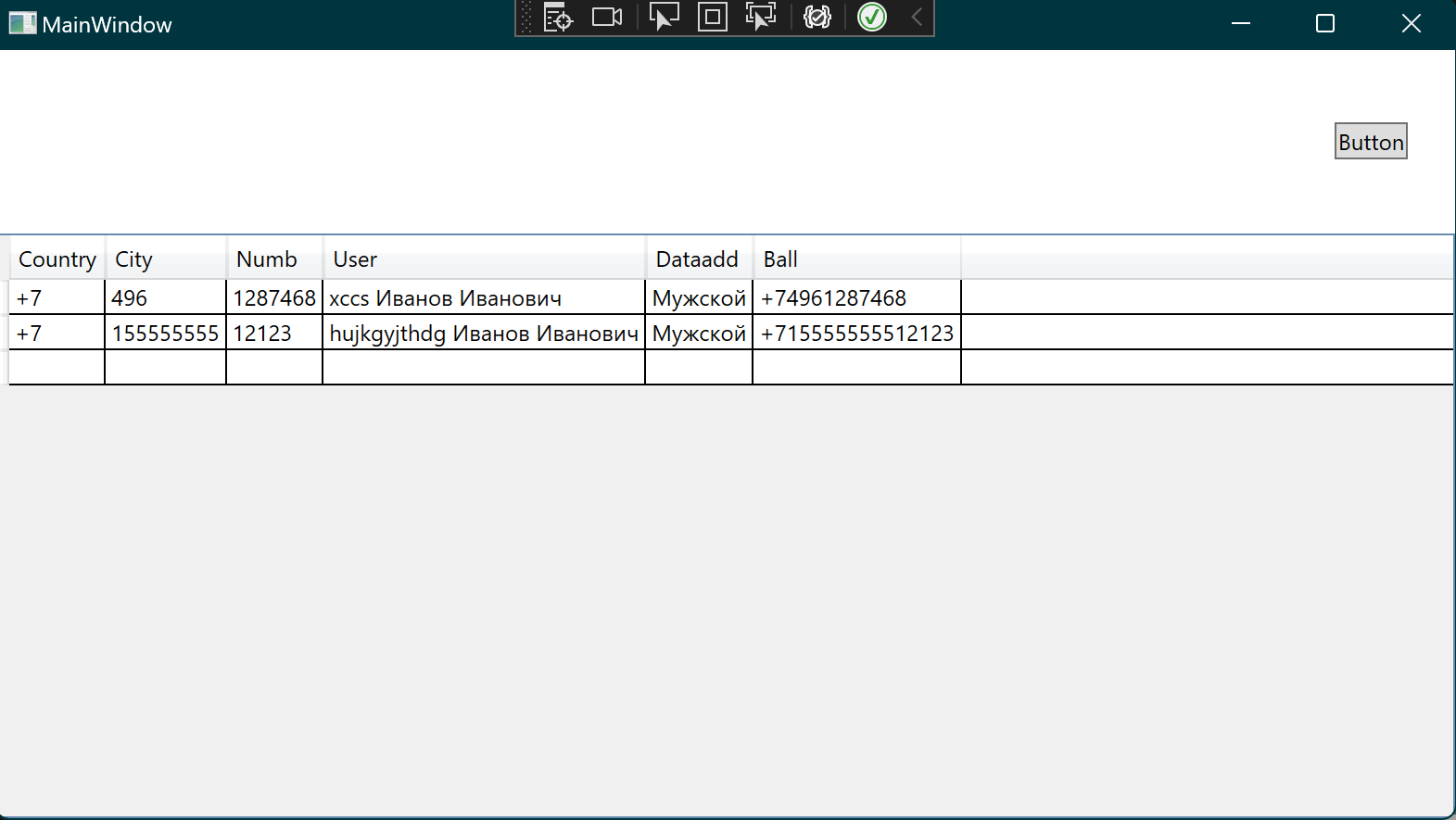


Рисунок 3. Главный экран.

Главный экран содержит следующие элементы:

* Таблица, содержащая информацию
* Кнопка "*button*" перенаправляет на окно добавления информации.
* При двойном нажатии на эллемент в столбце “ball” перенаправляется на окно редактирования или удаления информации строки.

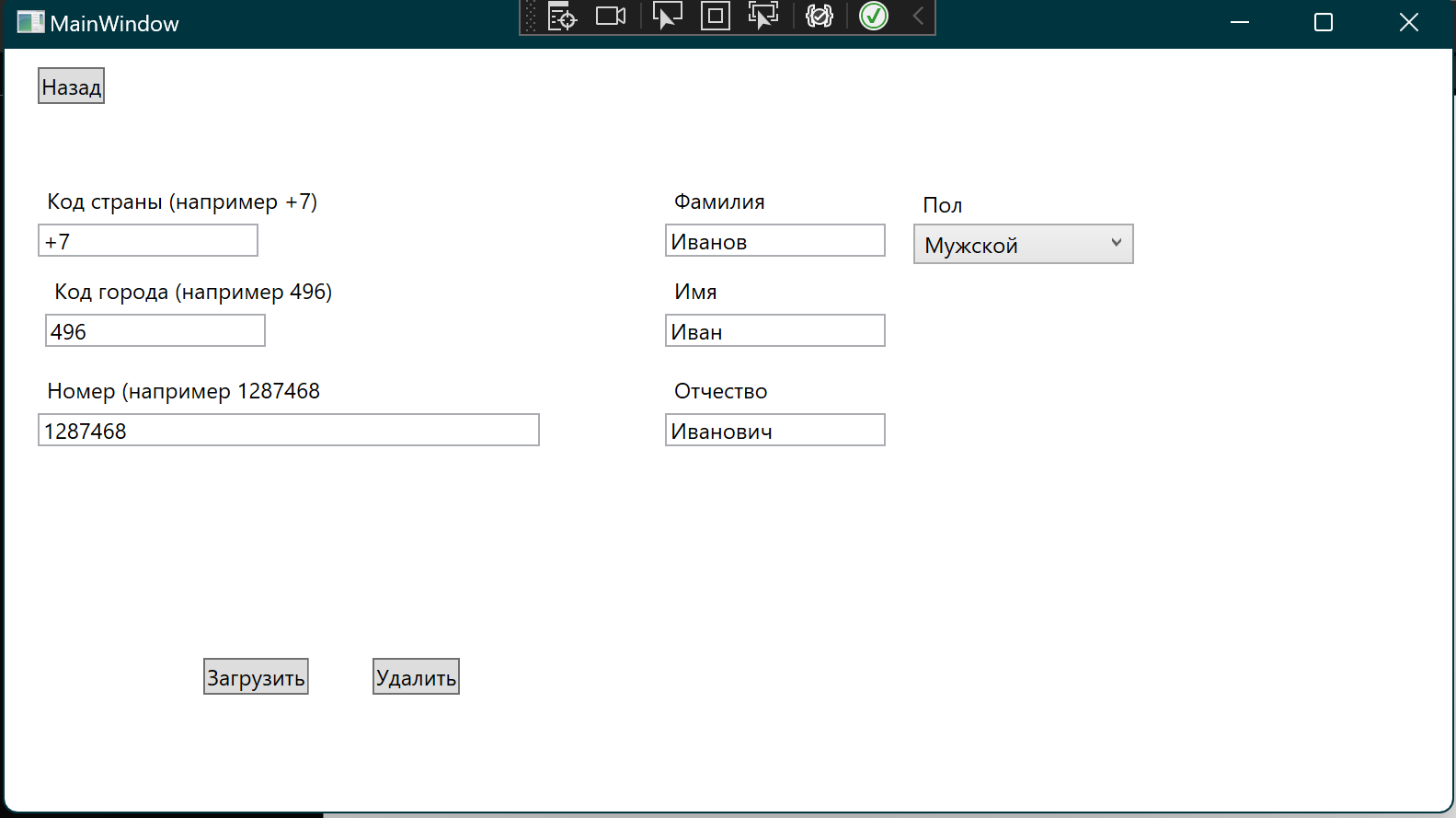


Рисунок 4. Окно редактирования.

Окно видео содержит следующие элементы:

* Поля:
  + Код страны ­ — изменение первой цифры кода страны
  + Код города — изменение кода города
  + Номер — изменение информации о номере телефона
  + Фамилия— изменение информации о фамилии
  + Имя — изменение имени
  + Отчество ­— изменение отчества
  + Пол— смена пола
* Кнопка “ удалить” —удаляет запись
* Кнопка “ загрузить” — обновляет информации о измененной записи

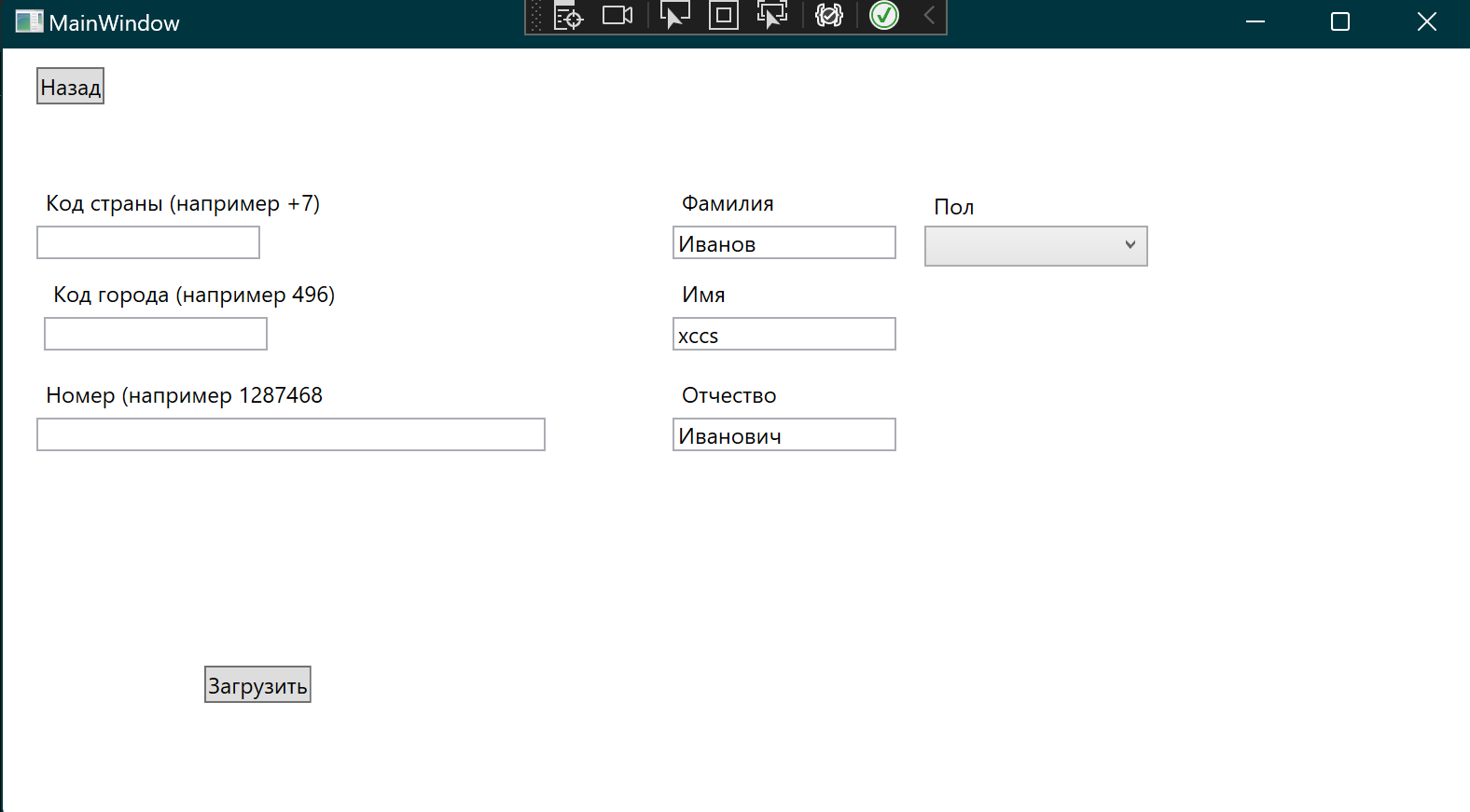


Рисунок 5. Окно добавления.

Окно видео содержит следующие элементы:

* Поля:
  + Код страны ­ — изменение первой цифры кода страны
  + Код города — изменение кода города
  + Номер — изменение информации о номере телефона
  + Фамилия— изменение информации о фамилии
  + Имя — изменение имени
  + Отчество ­— изменение отчества
  + Пол— смена пола
* Кнопка “ удалить” —удаляет запись
* Кнопка “ загрузить” — обновляет информации о измененной записи