Донской политехнический колледж

**Практическая работа**

По дисциплине «Проектирование пользовательских интерфейсов»

На тему:

«Страховая компания – “МИАЛ’»

Выполнил:

Студент группы П-20-1

Миляев А.А.

Преподаватель:

Гвоздев С.М.

Г.Донской

2022

**1 Создание проекта и разметка MainWindow**

Создаю новый проект – выбираю «Приложение WPF (.Net Framework)» и нажимаю «Далее».

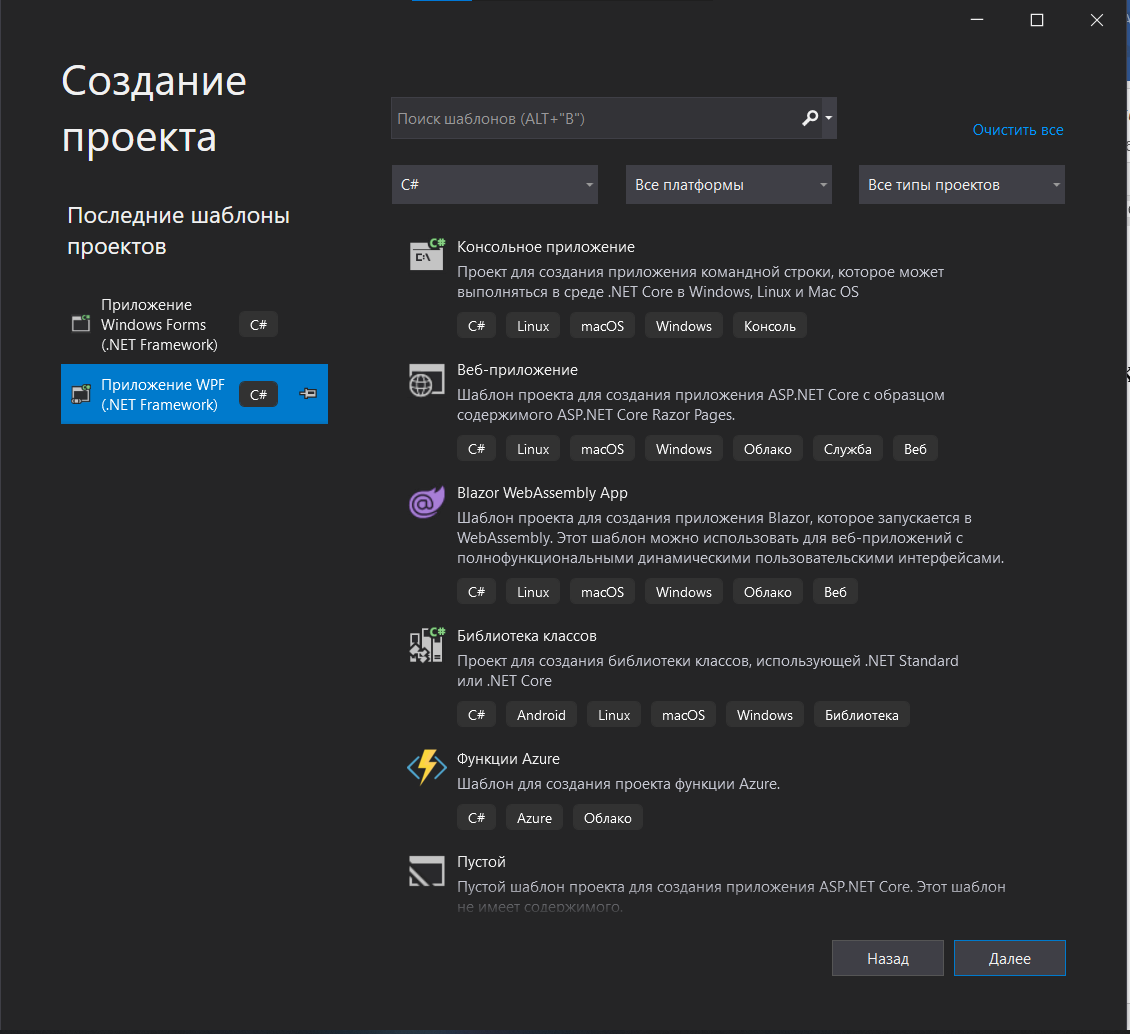


Рисунок 1.1 – Создание проекта

В окне «Настроить новый проект» заполняю поля:

* Имя проекта;
* Расположение;
* Выбираю платформу.

После заполнения всех полей нажимаем на кнопку «Создать».

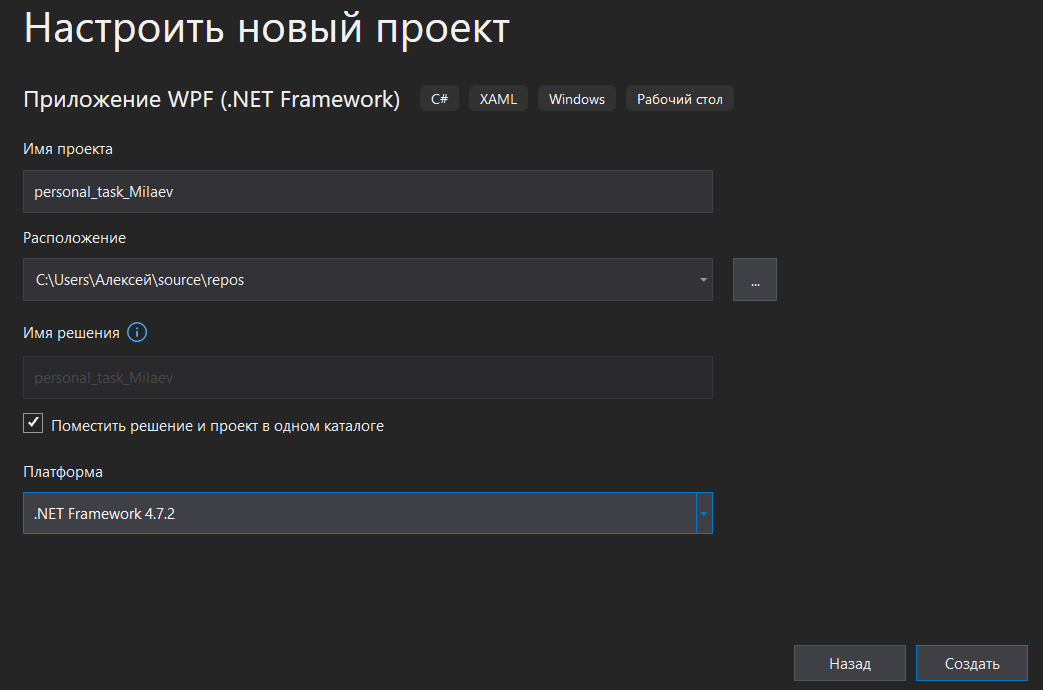


Рисунок 1.2 – Настройка нового проекта

Создадаю общую структуру проекта. Она будет состоять из следующих директорий:

* Core – директория которая содержит вспомогательные классы;
* Images – директория которая содержит графические ресурсы приложения;
* Model – директория будет содержать классы данных.
* View – директория которая содержит: окна, страницы и пользовательские элементы используемые в приложении.

Устанавливаю графическую библиотеку «Material Design». Устанавливаю два компонента «MaterialDesignThemes» и «MaterialDesignColors».

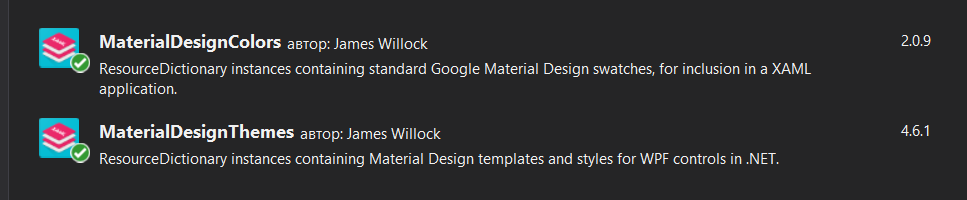


Рисунок 1.3 – Установка «MaterialDesignThemes» и «MaterialDesignColors».

Теперь переношу «MainWindow» в директорию «View» и изменяю файл «App.xaml». Изменяю путь к «MainWindow» в «StartupUri» и добавляю справочник ресурсов, а также добавляю новое пространство имен «materialDesign».

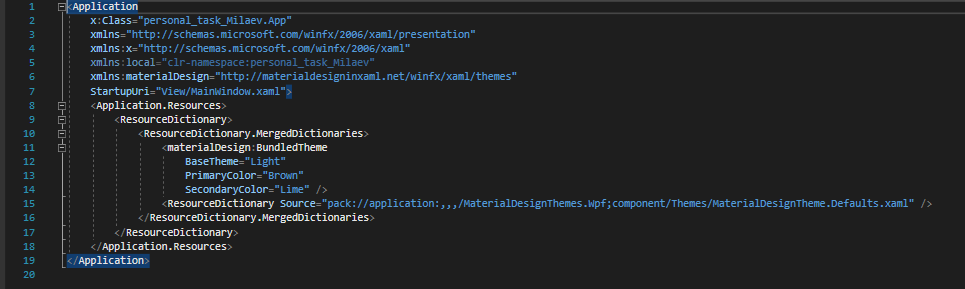


Рисунок 1.4 – Редактирование файла «App.xaml»

Далее перехожу в директорию «Model». В данную директорию импортирую, используя «EntityFramework» объектно-ориентированную модель базы данных. После импорта перехожу к разметке «MainWindow».

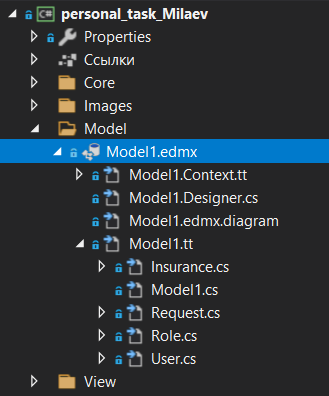


Рисунок 1.5 – Импорт данных

В «MainWindow» добавляю ссылку на «materialDesign», изменяю параметры высоты и ширины, отключаю стандартные стили окна, добавляю событие «MouseDown» и последним параметром я запрещаю пользователю менять размер окна (Разметка «MainWindow» предоставлена на рисунках 1.6 – 1.9).

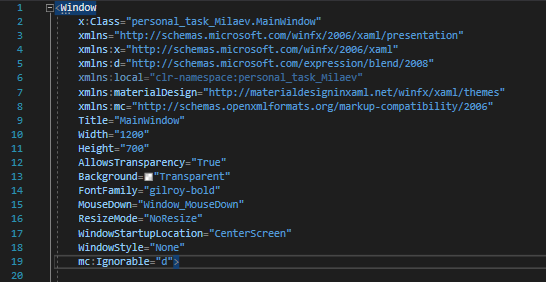


Рисунок 1.6 – Разметка «MainWindow»

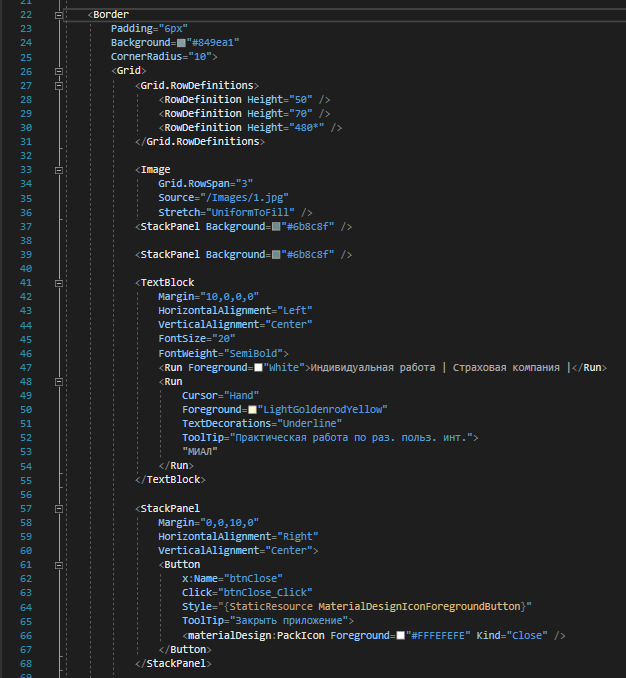


Рисунок 1.7 – Разметка «MainWindow»



Рисунок 1.8 – Разметка «MainWindow»



Рисунок 1.9 – Разметка «MainWindow»

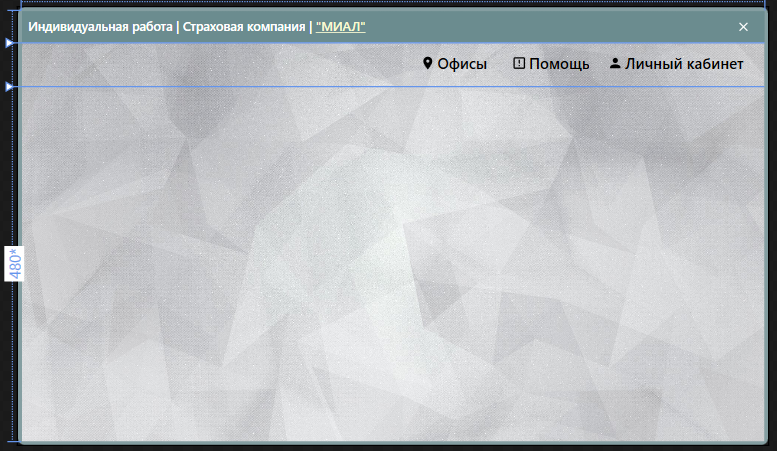


Рисунок 1.10 – Итог разметки «MainWindow»

Перейдем в директорию «Core». В этой директории создадим статический класс «FrameNavigate» с двумя статическими свойствами «FrameObject» (будет иметь тип данных «Frame») и «DB» (будет иметь тип данных «dbEntities3». Поле «FrameObject» будет использоваться для упрощенной навигации в приложении. Поле «DB» предоставить доступ объектно-ориентированной модели данных.

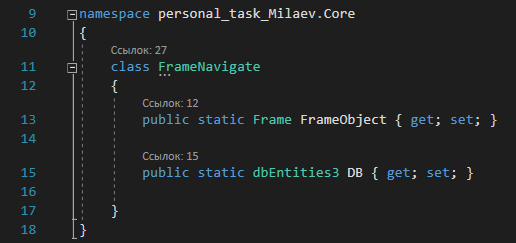


Рисунок 1.11 – Реализация класса «FrameNavigate»

Далее переходим в файл «MainWindow.xaml.cs». В этом файле свяжем поле (из класса «FrameNavigate») «FrameObject» с «MainWindowFrame», а в поле «DB» создадим экземпляр «dbEntities3». Также реализуем два события: «Window\_MouseDown» – данное событие разрешить пользователю перемещать «окно» приложения; «BtnClose\_Click» – данное событие будет завершать работу приложения.



Рисунок 1.12 – Реализация логики в файле «MainWindow.xaml.cs»

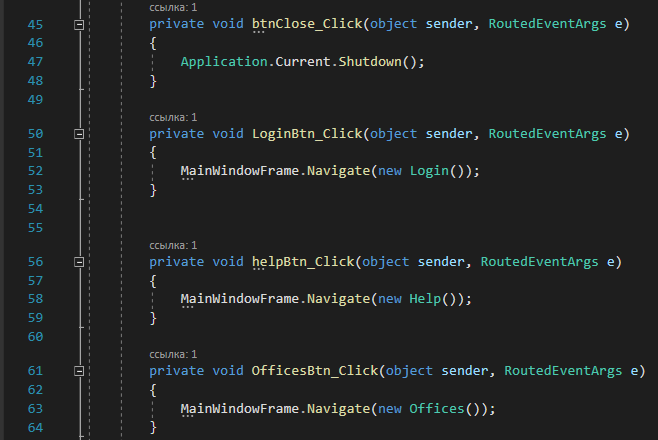


Рисунок 1.13 – Реализация логики в файле «MainWindow.xaml.cs»

Переходим на страницу «Help.xaml» (Разметка «Help.xaml» приведена на рисунках 1.17 – 1.22).

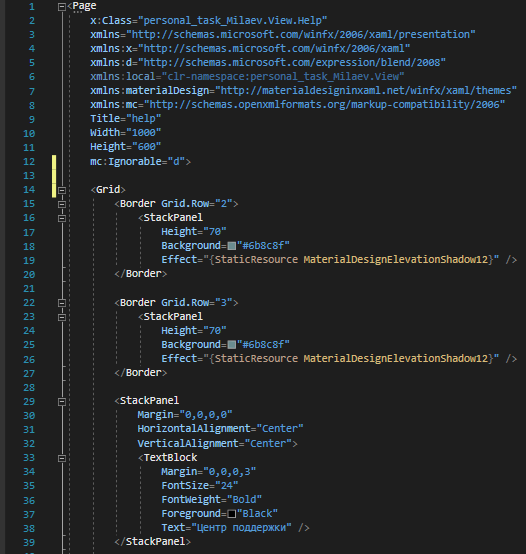


Рисунок 1.14 – Разметка «Help.xaml»

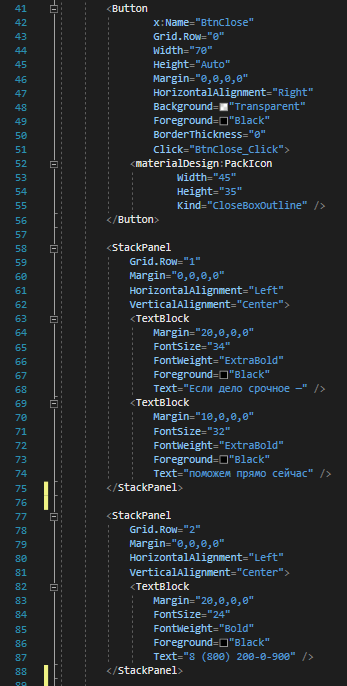


Рисунок 1.15 – Разметка «Help.xaml»

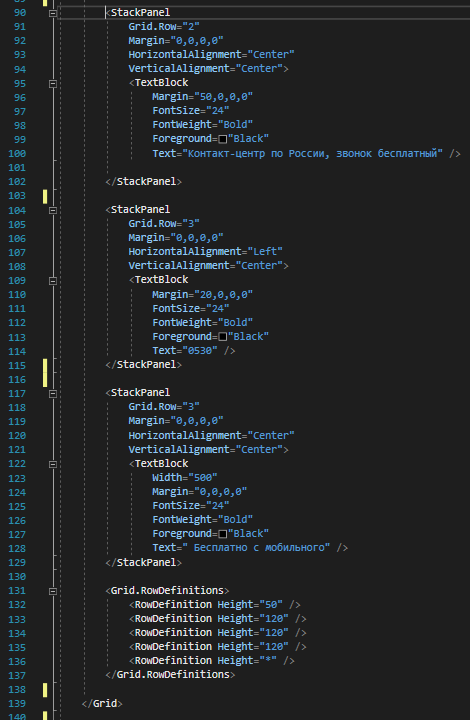


Рисунок 1.16 – Разметка «Help.xaml»

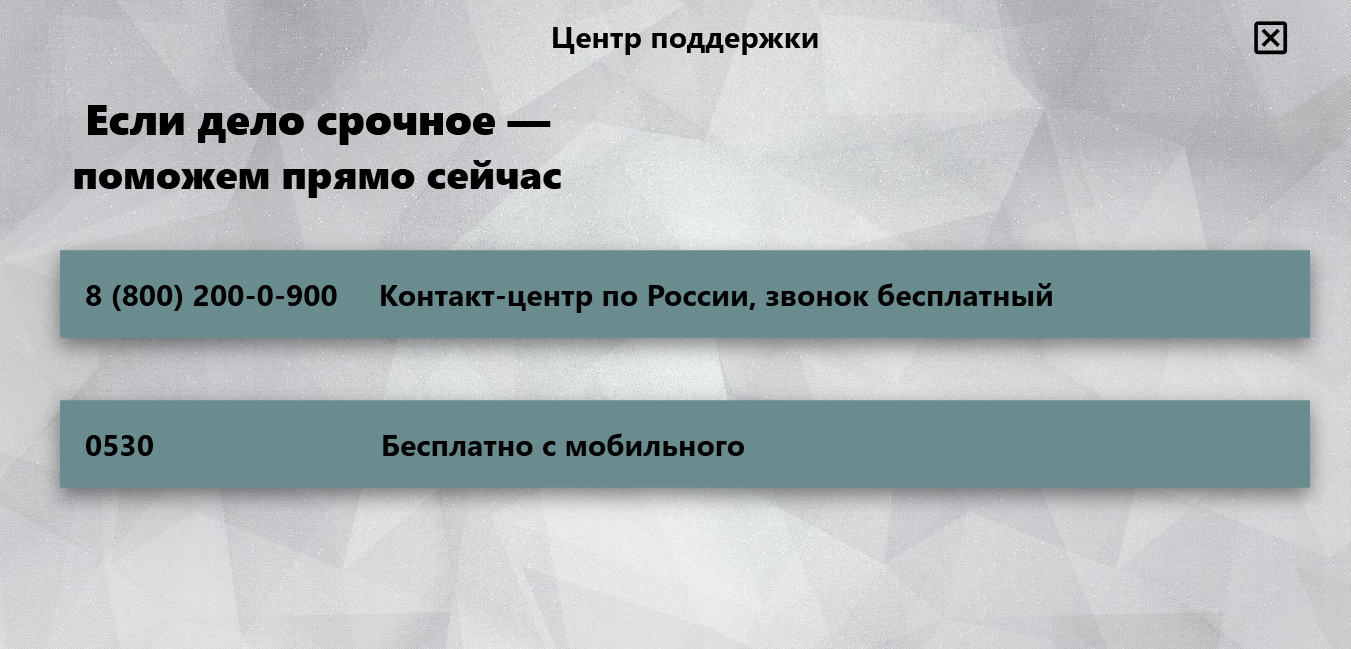


Рисунок 1.17 – Результат разметки «Login.xaml»

Переходим на страницу «Offices.xaml» (Разметка «Offices.xaml» приведена на рисунках 1.17 – 1.22).



Рисунок 1.18 – Разметка «Offices.xaml»

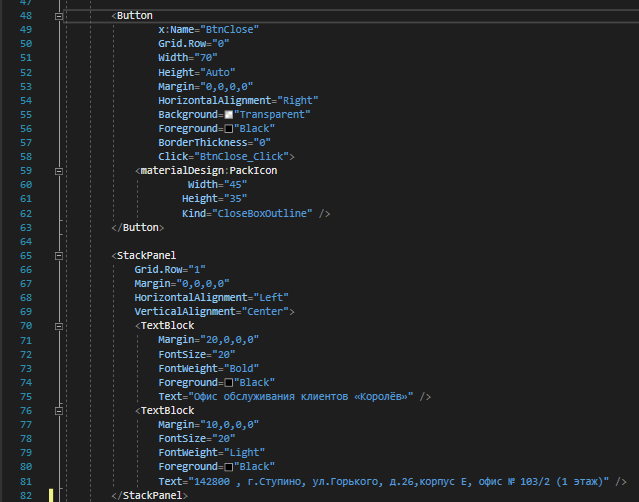


Рисунок 1.19 – Разметка «Offices.xaml»

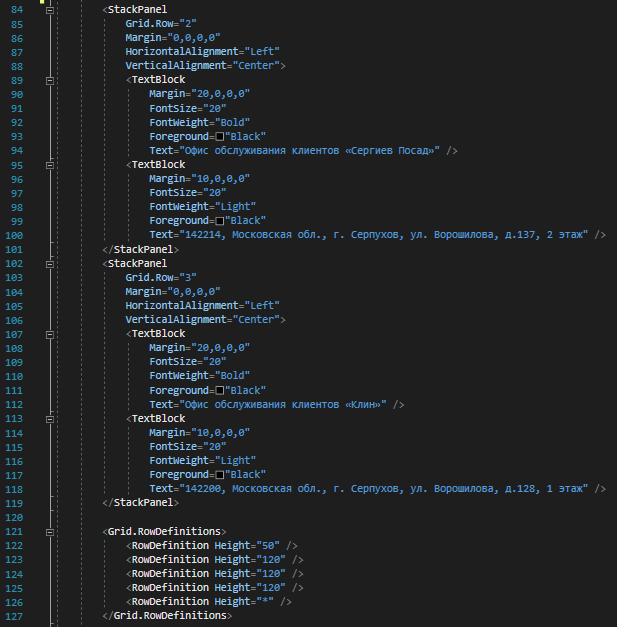


Рисунок 1.20 – Разметка «Offices.xaml»

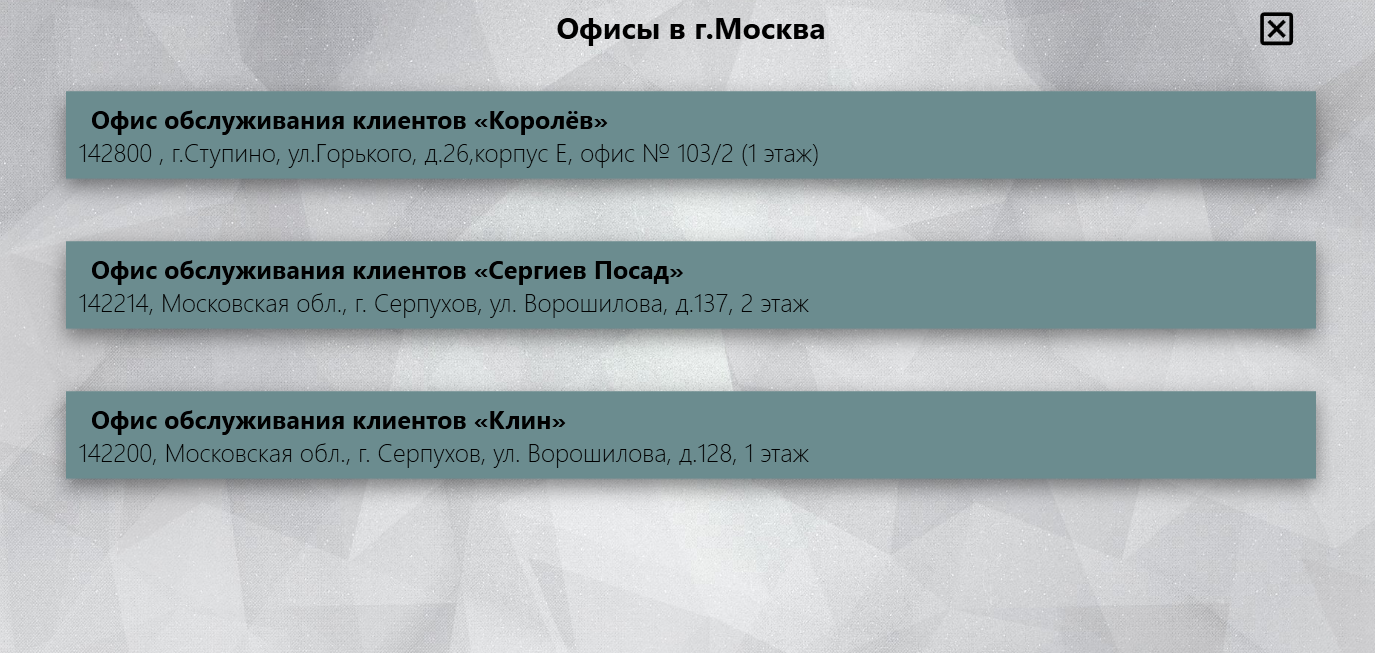


Рисунок 1.21 – Результат разметки «Offices.xaml»

Переходим на страницу «Login.xaml». Импортируем пространство имен «materialDesign» и переходим к разметке (Разметка «Login.xaml» приведена на рисунках 1.17 – 1.22).



Рисунок 1.22 – Разметка «Login.xaml»

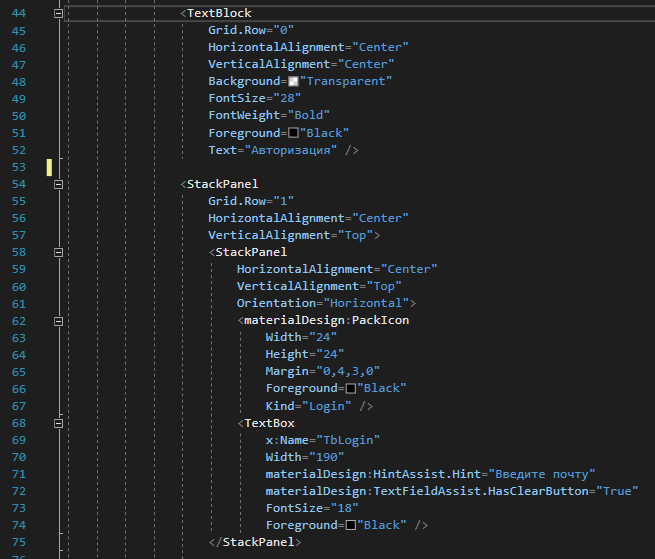


Рисунок 1.23 – Разметка «Login.xaml»

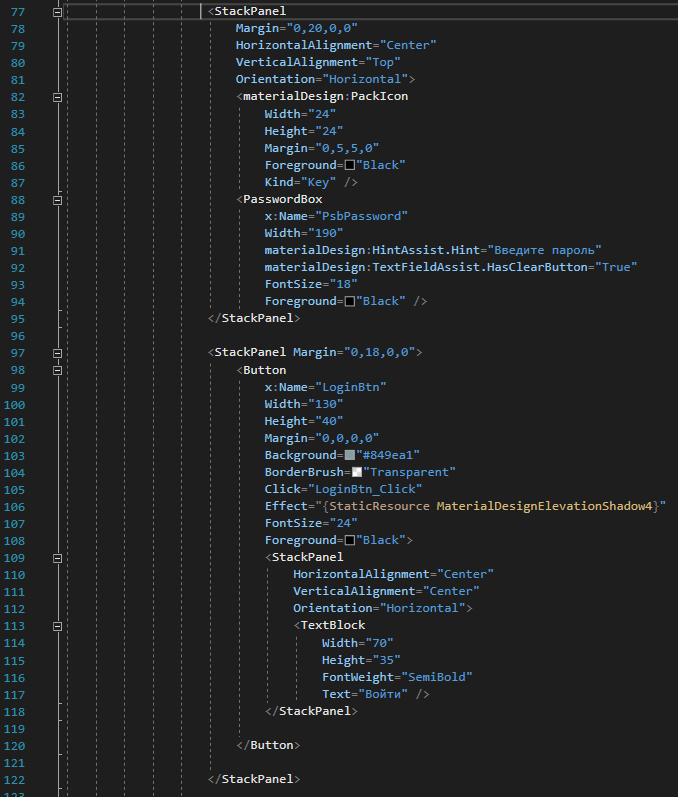


Рисунок 1.24 – Разметка «Login.xaml»

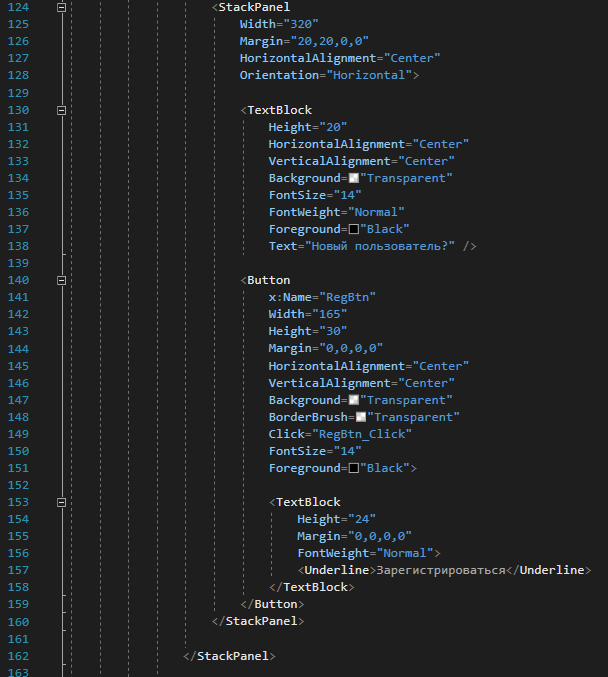


Рисунок 1.25 – Разметка «Login.xaml»

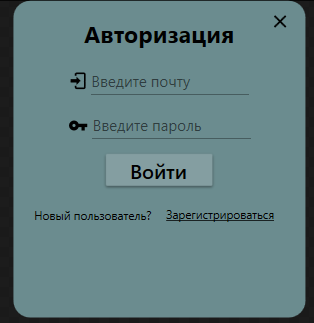


Рисунок 1.26 – Результат разметки «Login.xaml»

Далее переходим в файле «Login.xaml.cs». Этот файл будет содержать логику авторизации при клике на кнопку «Войти» (Реализация логики приведена на рисунках 1.23 – 1.24).

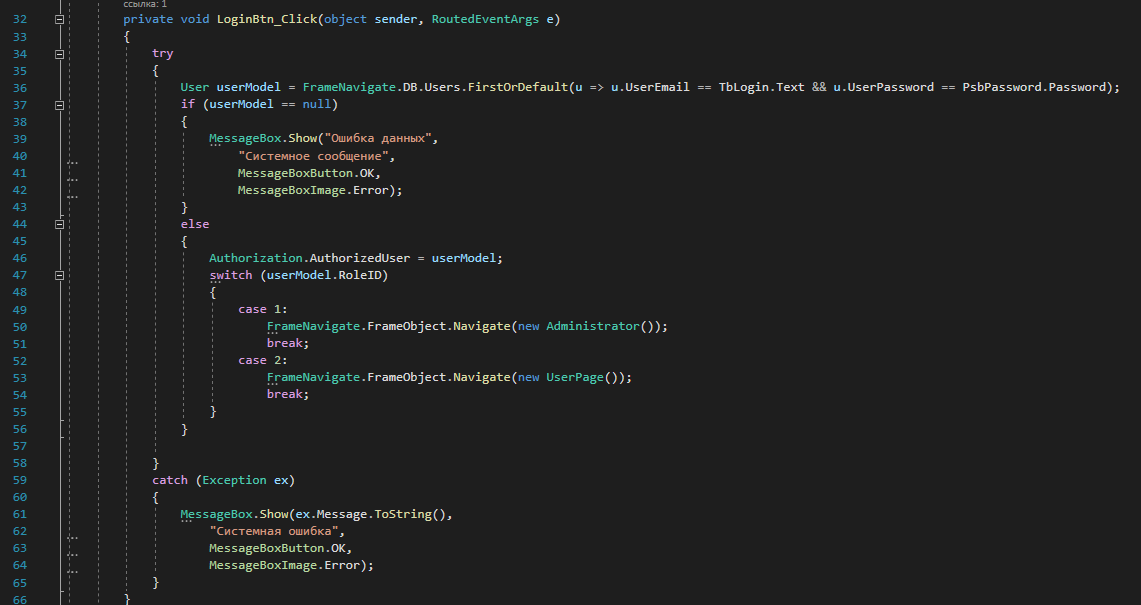


Рисунок 1.27 – Реализация логики файла «Login.xaml.cs»

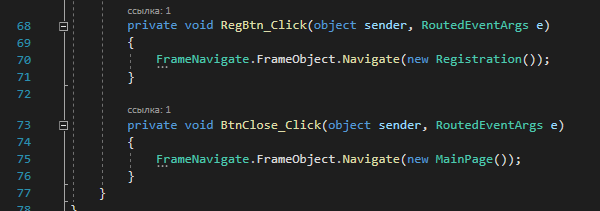


Рисунок 1.28 – Реализация логики файла «Login.xaml.cs»

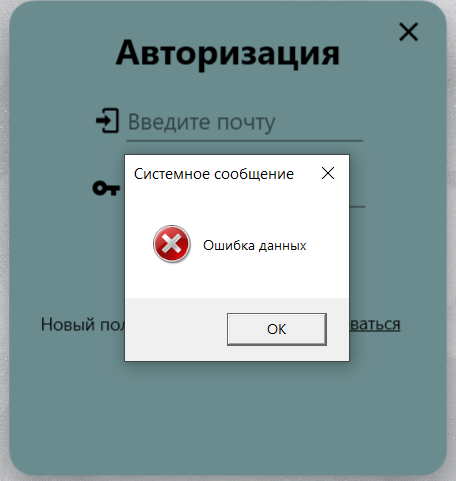


Рисунок 1.29 – Введены некорректные данные

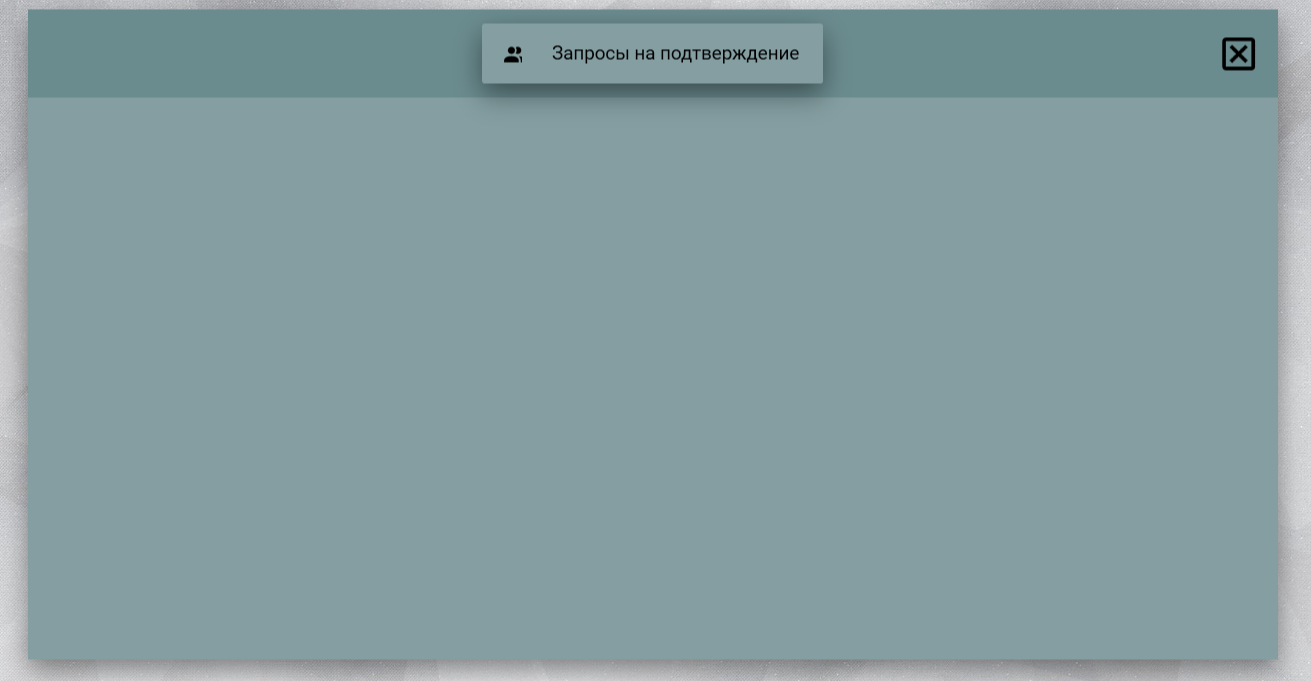


Рисунок 1.30 – Введены данные учетной записи администратора

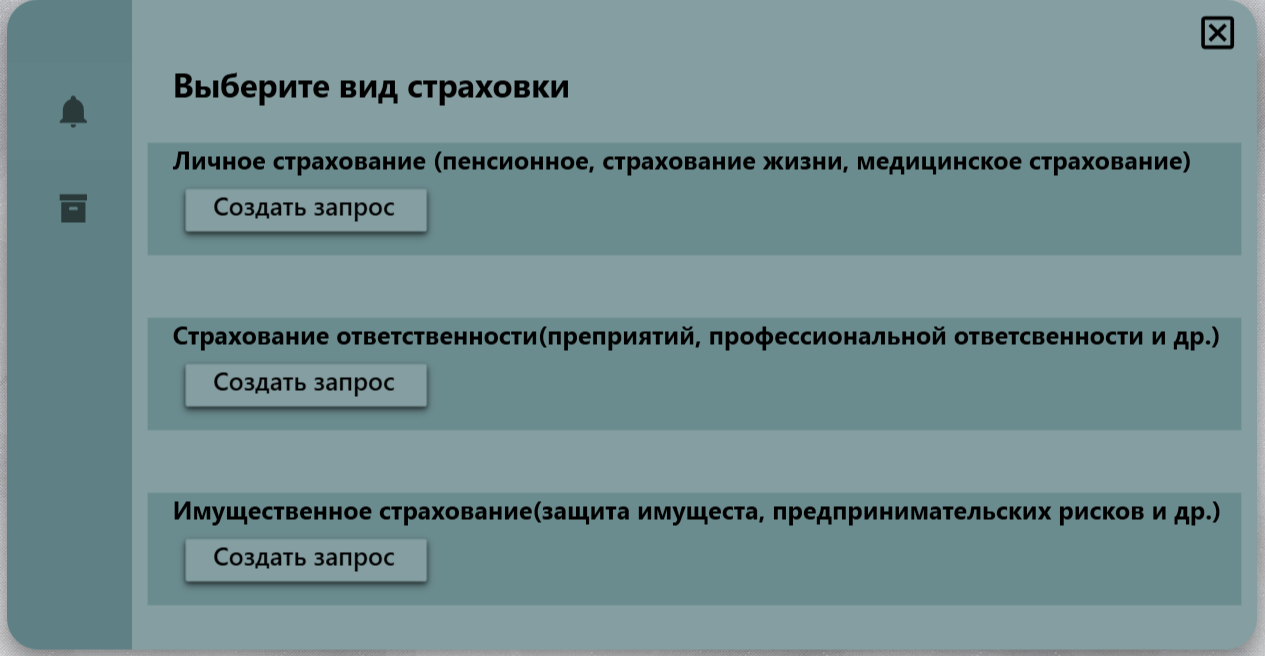


Рисунок 1.31 – Введены данные учетной записи пользователя

**2 Реализация Registration**

Перейдем к разметке «Registration» (разметка представлена на рисунках 2.1 – 2.5).

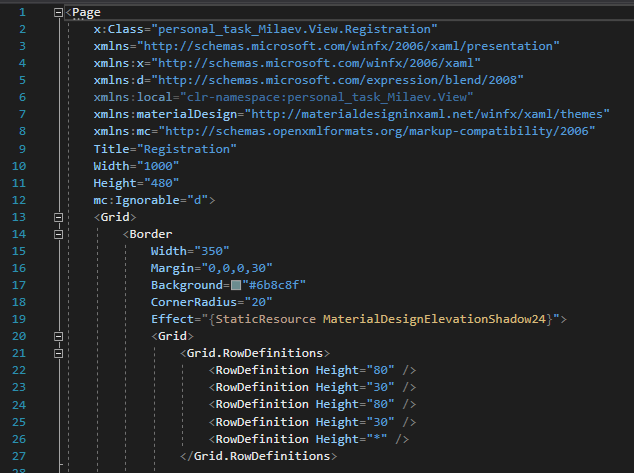


Рисунок 2.1 – Разметка «Registration»

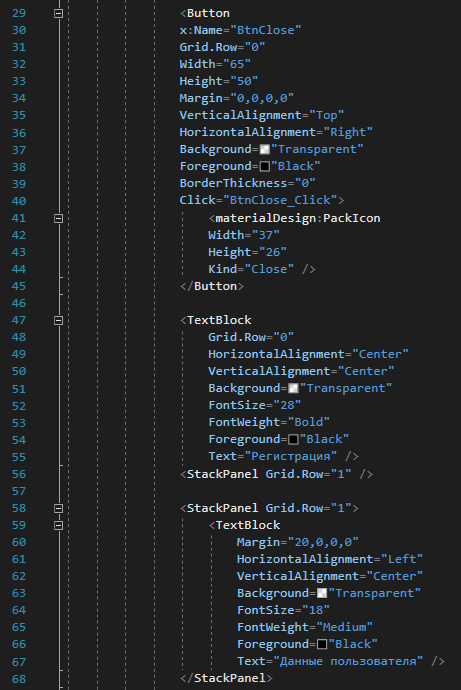
****

Рисунок 2.2 – Разметка «Registration»

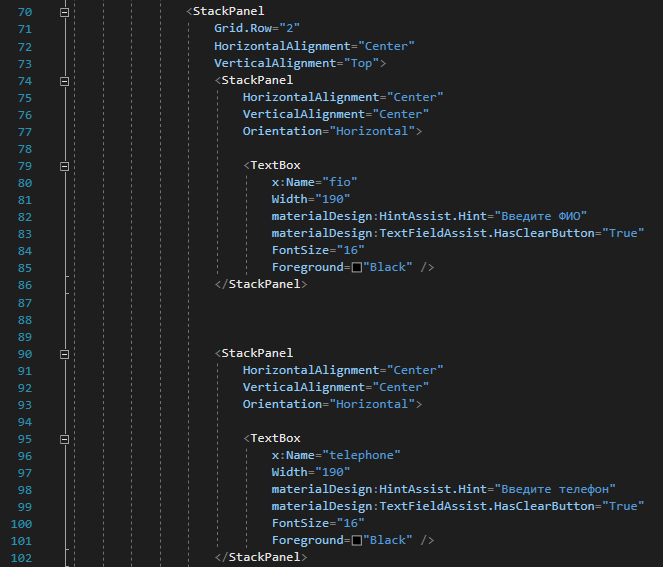


Рисунок 2.3 – Разметка «Registration»



Рисунок 2.4 – Разметка «Registration»

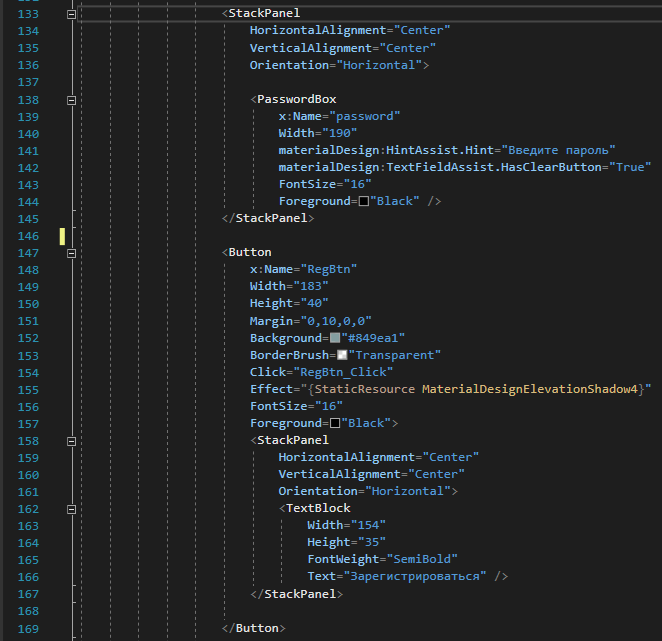


Рисунок 2.5 – Разметка «Registration»

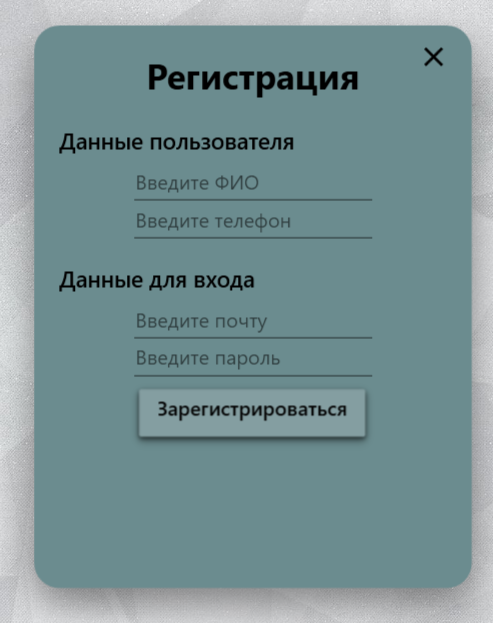


Рисунок 2.6 – Результат разметки «Registration»

Теперь перейдем в файл «Registration.xaml.cs» для реализации логики регистрации. Сделаем проверку заполнения полей, проверку на ошибку соединения с базой данных, также при успешной регистрации будет выводит диалоговое окно и перенаправлять на страницу «Login» (реализация логики представлена на рисунках 2.7 – 2.8).

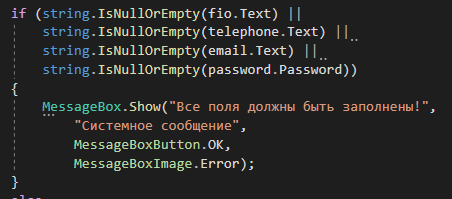


Рисунок 2.7 – Реализация файла «Registration.xaml.cs»

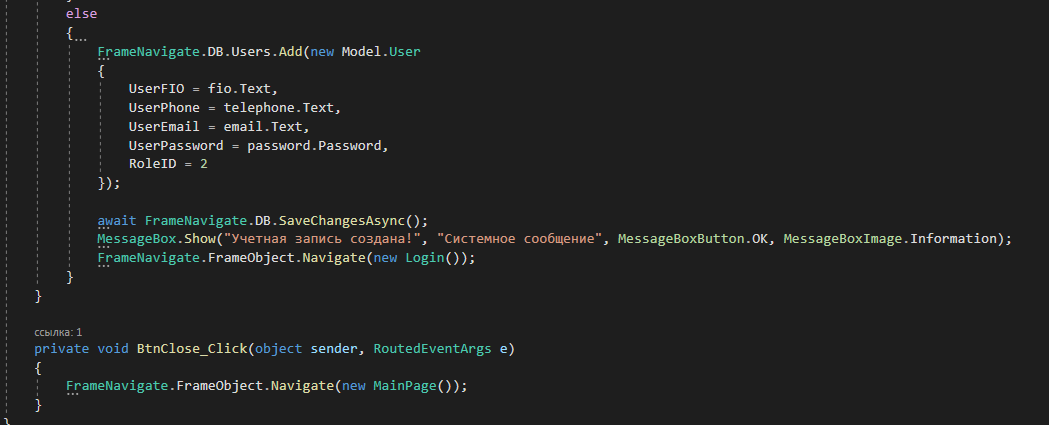


Рисунок 2.8 – Реализация файла «Registration.xaml.cs»

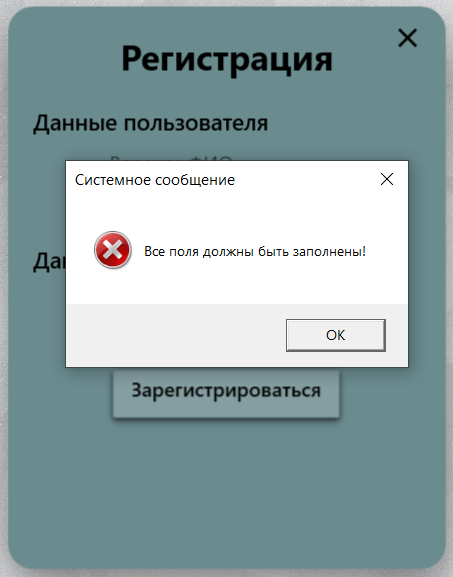


Рисунок 2.9 – Ошибка заполнения полей

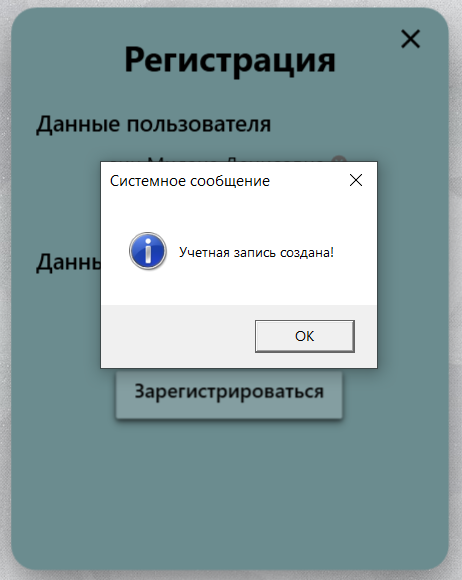


Рисунок 2.10 – Успешное создание учетной записи

**3 Реализация UserPage**

Создаём страницу «UserPage» куда будет попадать пользователь после авторизации (разметка и логика «UserPage» представлена на рисунках 3.1 – 3.5).

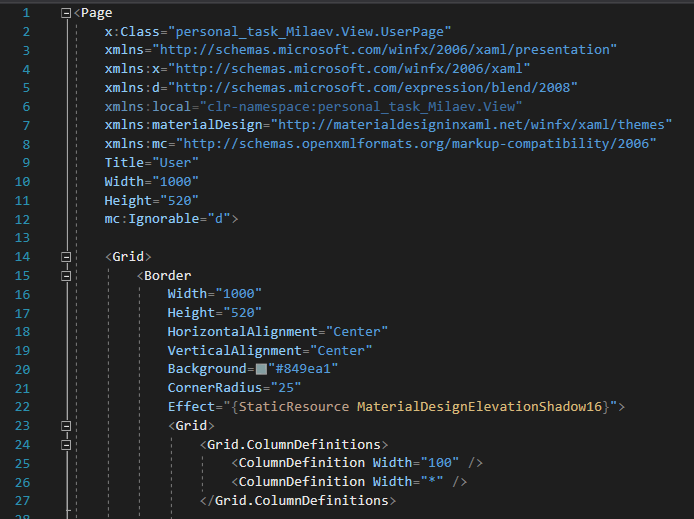


Рисунок 3.1 – Разметка «UserPage»

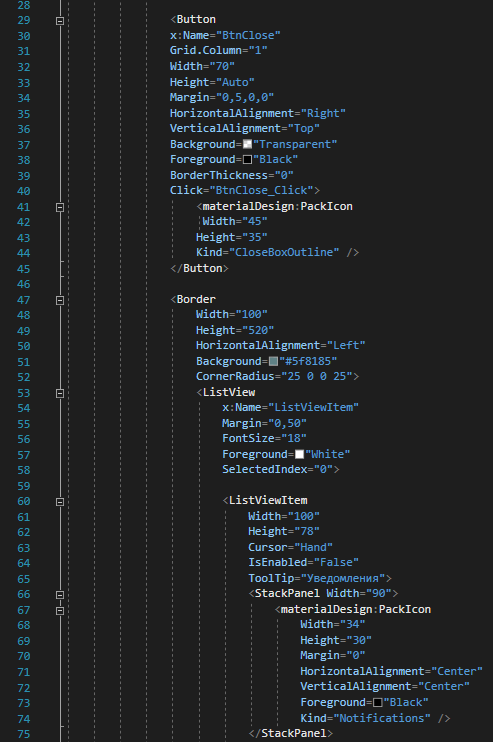


Рисунок 3.2 – Разметка «UserPage»

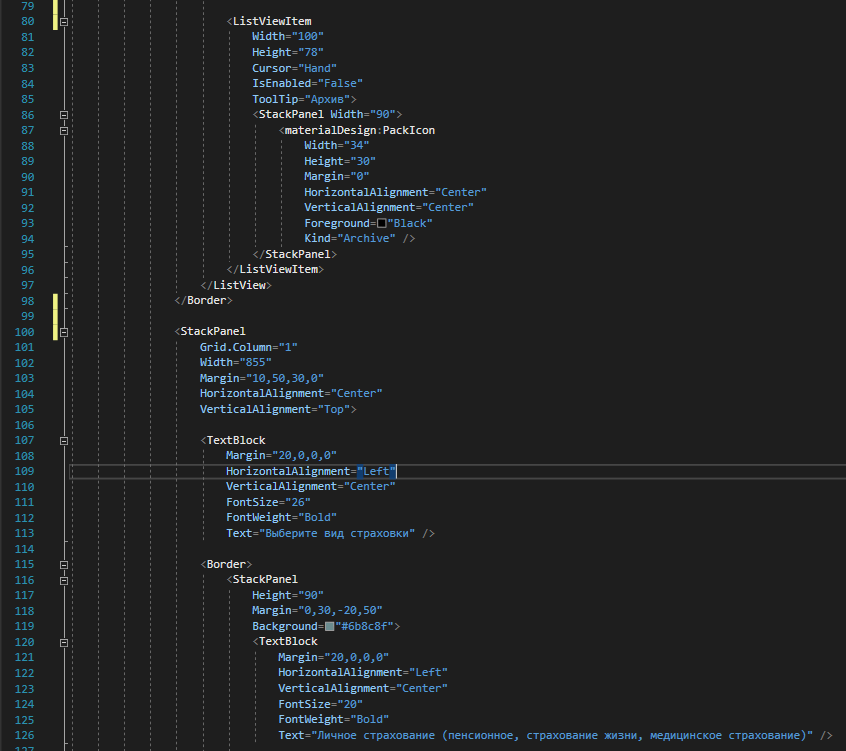


Рисунок 3.3 – Разметка «UserPage»

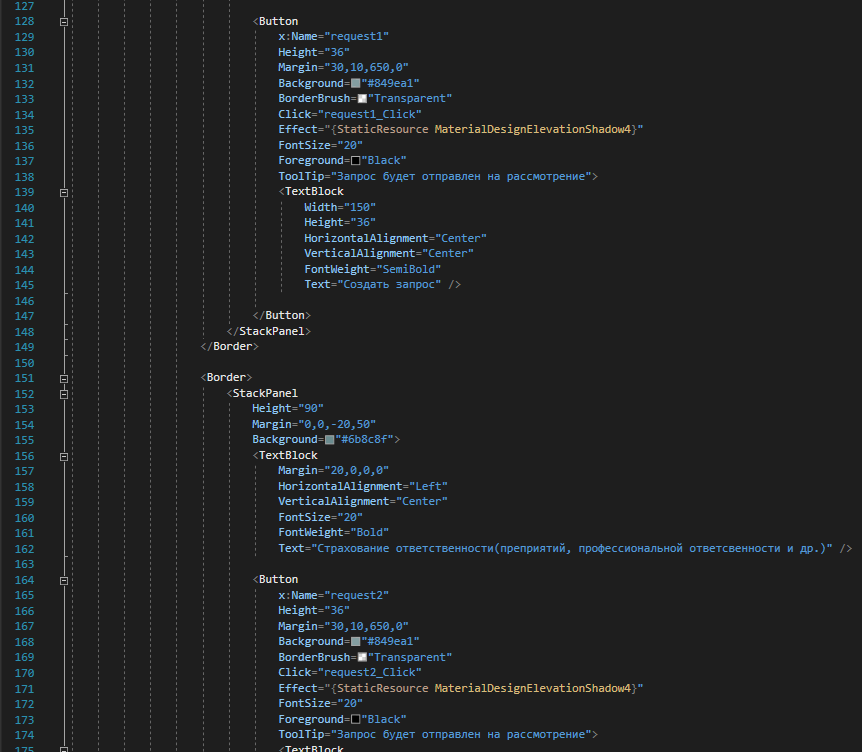


Рисунок 3.4 – Разметка «UserPage»

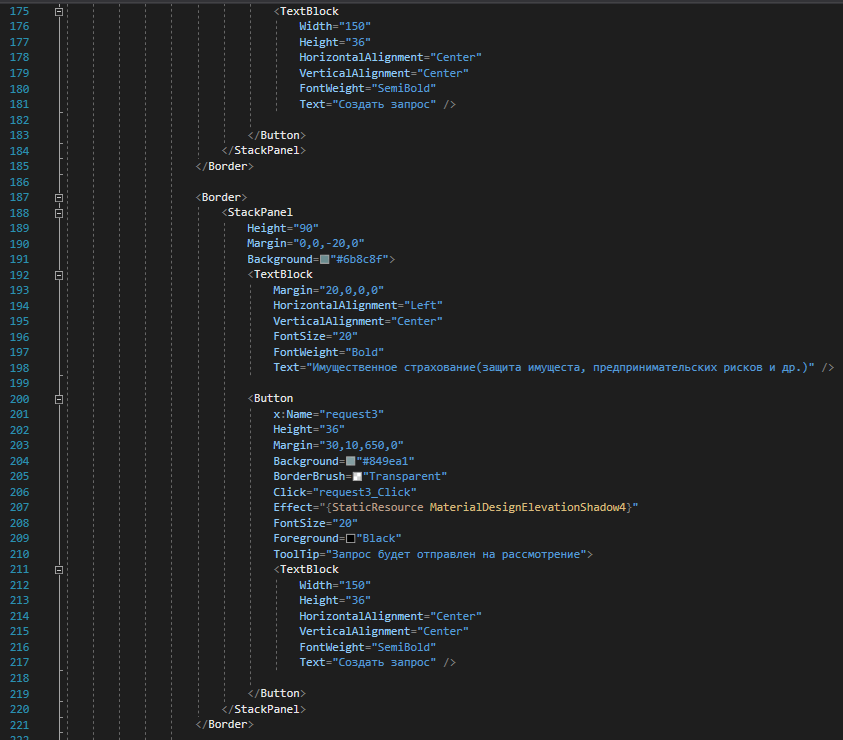


Рисунок 3.5 – Разметка «UserPage»

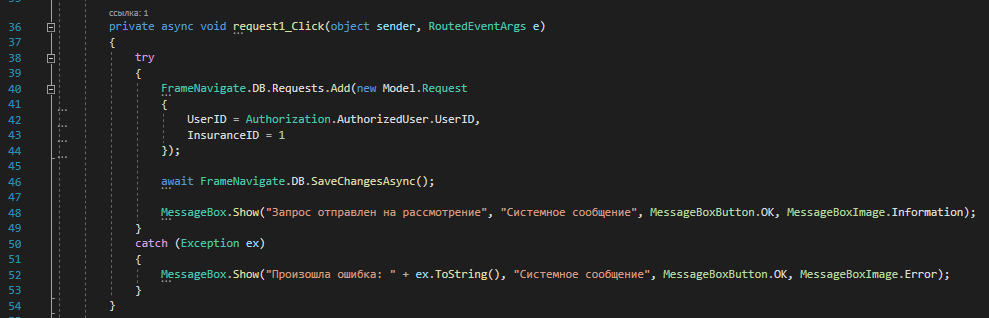


Рисунок 3.6 – Реализация логики файла «UserPage.xaml.cs»

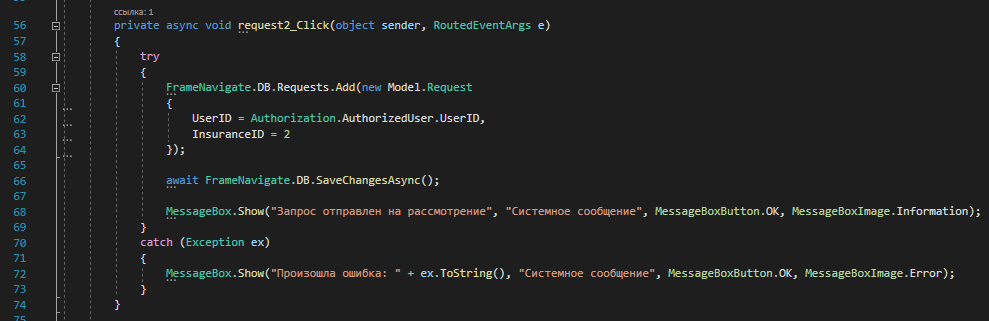


Рисунок 3.7 – Реализация логики файла «UserPage.xaml.cs»



Рисунок 3.8 – Реализация логики файла «UserPage.xaml.cs»

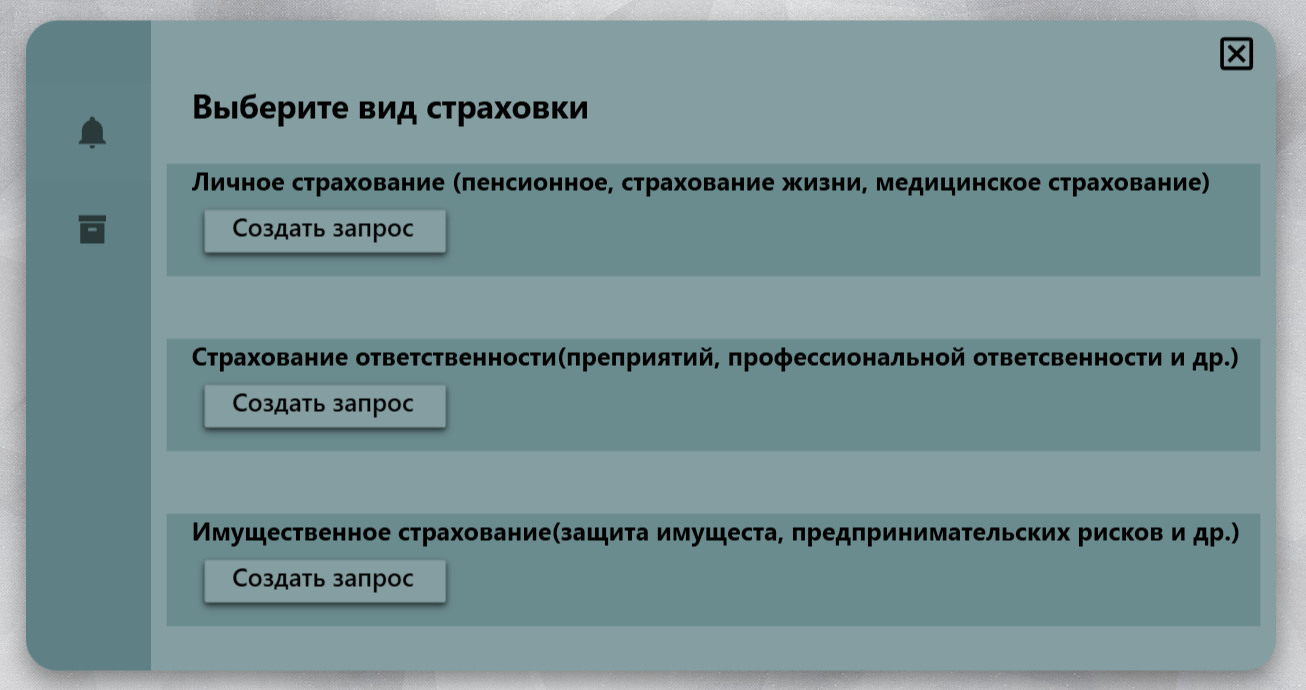


Рисунок 3.9 – Результат разметки «UserPage»

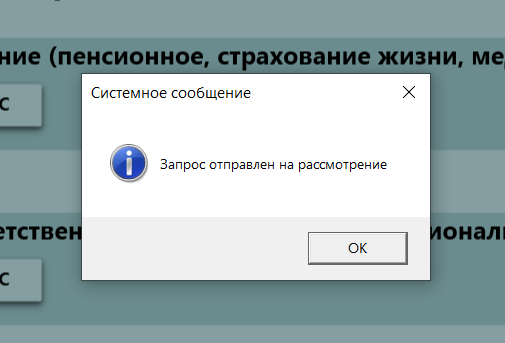


Рисунок 3.10 – Логика работы «UserPage»

**4 Реализация Administrator**

Для реализации логики создадим директорию «AdminUserControls» и пользовательский элемент AdminUserControl». «Administrator» будет содержать меню в зависимости от выбора будет подгружаться пользовательский элемент со своей логикой работы. Администратор в данном приложении будет иметь возможность отменять и принимать запросы (реализация разметки и разметки представлены на рисунках 5.1 – 5.).

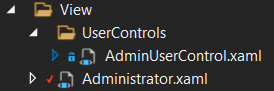


Рисунок 4.1 – Создание страницы и пользовательского элемента

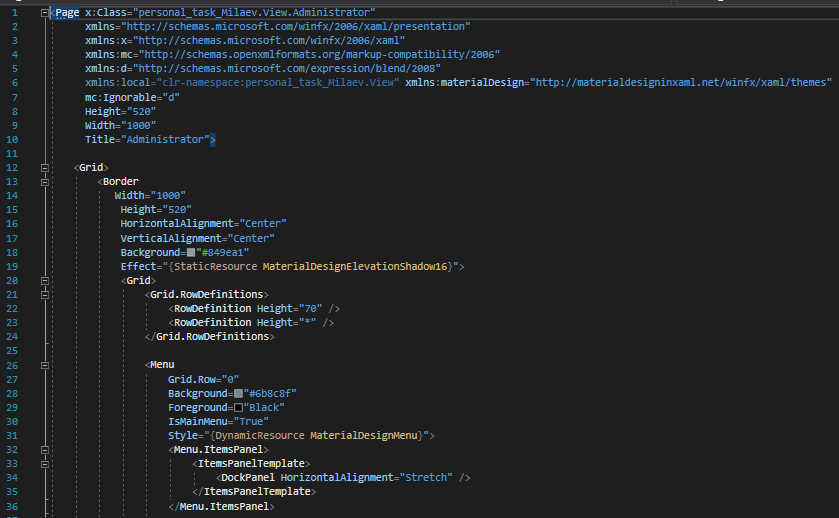


Рисунок 4.2 – Разметка «Administrator»



Рисунок 4.3 – Разметка «Administrator»

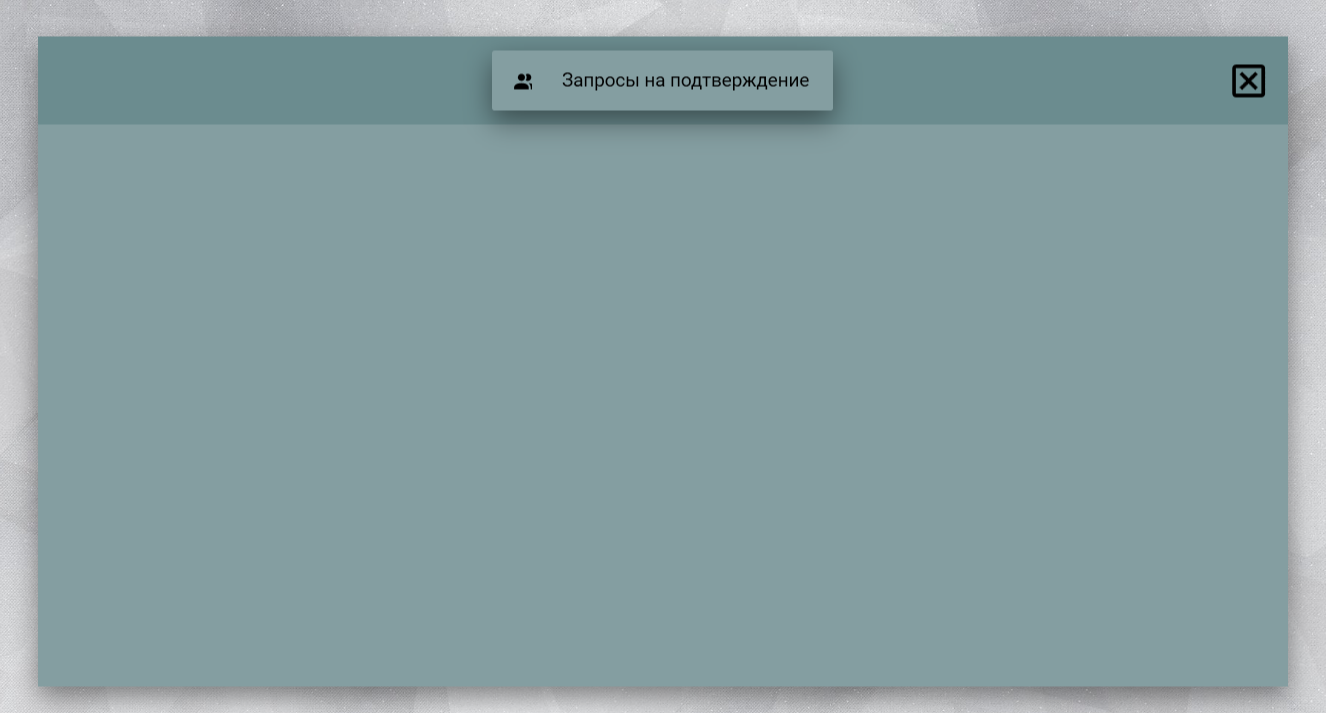


Рисунок 4.4 – Результат разметки «Administrator»

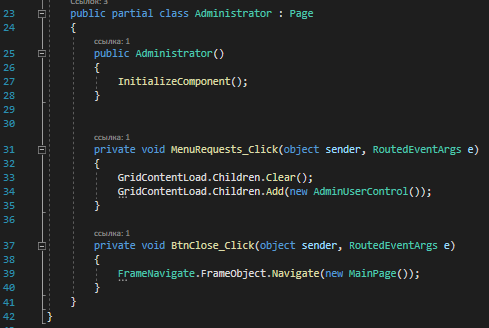


Рисунок 4.5 – Реализация логики файла «Administrator.xaml.cs»

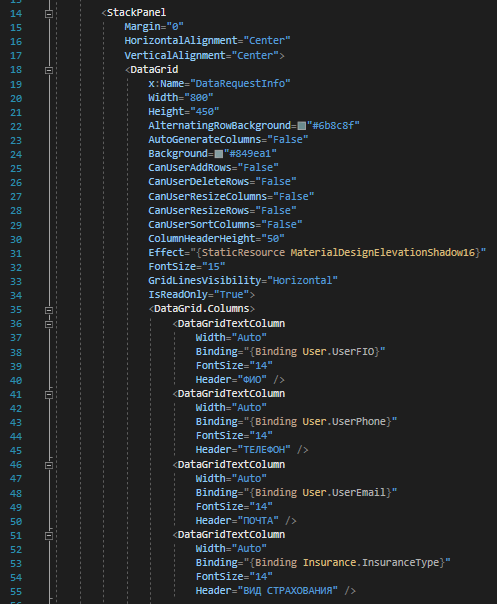


Рисунок 4.6 – Разметка «AdminUserControl»

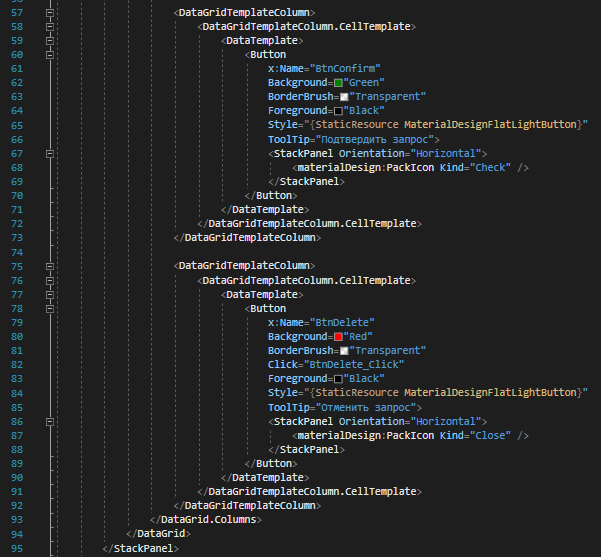


Рисунок 4.7 – Разметка «AdminUserControl»

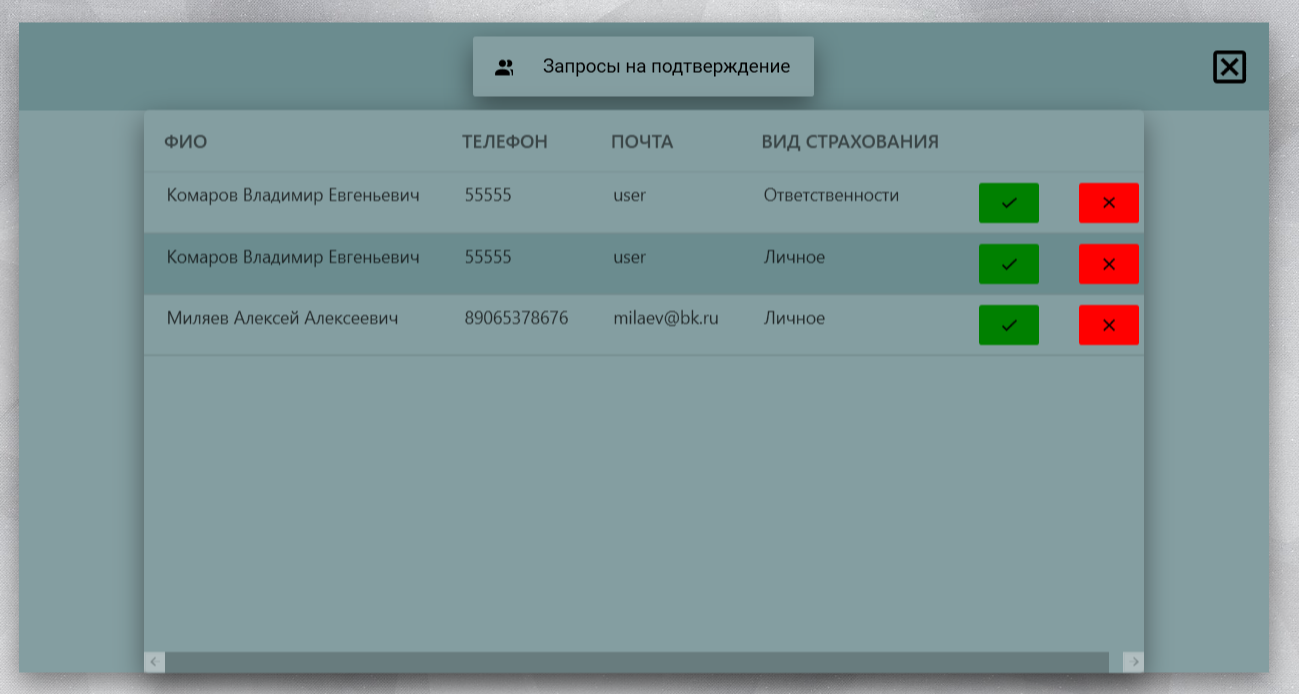


Рисунок 4.8 – Результат разметки «AdminUserControl»

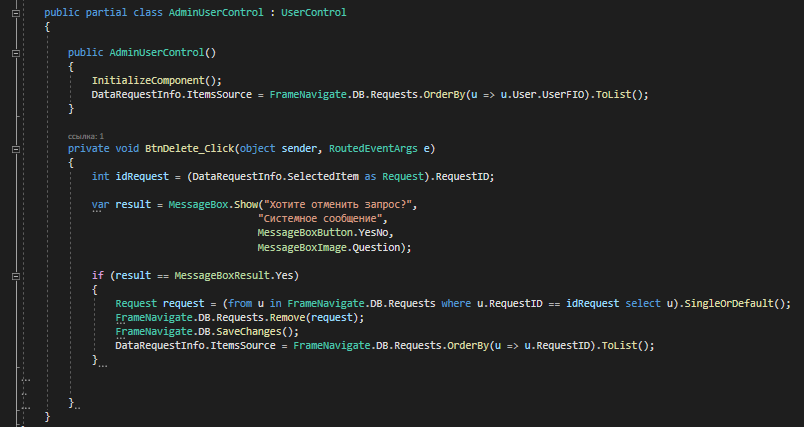


Рисунок 4.9 – Реализация логики файла «AdminUserControl.xaml.cs»

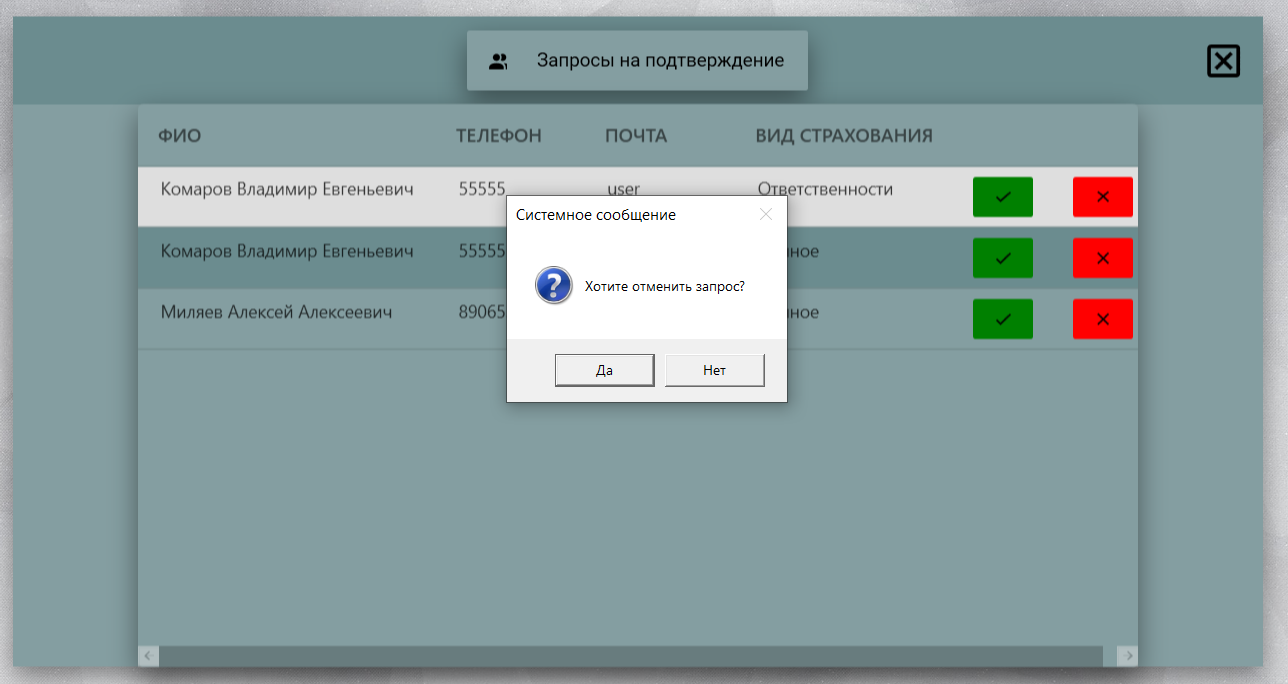


Рисунок 4.10 – Результат работы «AdminUserControl»