**ОТЧЕТ**

**о выполнении лабораторной работы №6**

**Дисциплина «МДК 03.01 Моделирование и анализ программного обеспечения»**

**Тема: «Проверки на стороне клиента»**

**Специальность: 09.02.07 Информационные системы и программирование**

**Квалификация: Специалист по информационным системам**

Выполнил: Хотин М.Ю.

Студент группы: ИС50-11-22

Проверил:

Преподаватель: Кретова Е.М.

Дата: «\_\_\_» \_\_\_\_\_\_\_\_\_ 2023 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Теоретические сведения**

**Валидация** – это процесс проверки данных на соответствие различным критериям.

При разработке любого приложения в большинстве случаев разработчику приходится иметь дело с обработкой данных, которые ввел пользователь в соответствующие поля на форме. По разным причинам пользователь может вводить некорректные данные. Например, в поле "Возраст" он по ошибке может указать отрицательное или вовсе нечисловое значение.

На практике встречается множество ситуаций, где определенные поля в пользовательском интерфейсе могут содержать только те данные, которые строго соответствуют определенным шаблонам. Если не учитывать подобные ситуации, то в процессе работы приложения могут возникать сбои, связанные с некорректным вводом пользователя. Ошибки могут быть допущены случайно или намеренно. В последнем случае пользователь может вводить некорректные данные с целью вывести приложение из строя и таким образом произвести атаку на приложение. Поэтому каждое приложение, которое работает с данными, которые вводит пользователь (а таких приложений большинство) должно производить предварительную проверку корректности введенных данных.

Существует два вида валидации данных, введенных пользователем:

* клиентская валидация;
* серверная валидация.

**Клиентская валидация** производится в браузере на стороне клиента. Как правило, логика валидации на стороне клиента реализуется посредством сценариев JavaScript, которые запускаются внутри браузера. Клиентская валидация обычно содержит несложные алгоритмы проверки. Это происходит в силу того, что клиентский код физически не может обратиться к серверным ресурсам (например, к базе данных). Поэтому на стороне клиента проверяются самые тривиальные сценарии (такие как проверка длины строки, проверка на вхождение в диапазон и т.д.).

*Клиентская валидация* может отсутствовать в приложении. Однако, если сценарии валидации присутствуют на стороне клиента, это может избавить от необходимости лишних обращений к серверу в случае невыполнения простых условий проверки.

**Серверная валидация** работает в рамках программного кода, размещенного на стороне сервера. Здесь проверяются всевозможные случаи, в том числе те, которые уже были проверены на стороне клиента. Кроме тривиальных проверок, на стороне сервера могут работать более сложные алгоритмы. Необходимость дублирования проверки сценариев, которые уже были проверены на стороне клиента, обусловлена тем, что клиентские проверки могут не сработать, если в браузере у клиента отключено исполнение сценариев JavaScript. Другими словами, наличие клиентской валидации не может гарантировать успешную проверку определенных там ограничений.

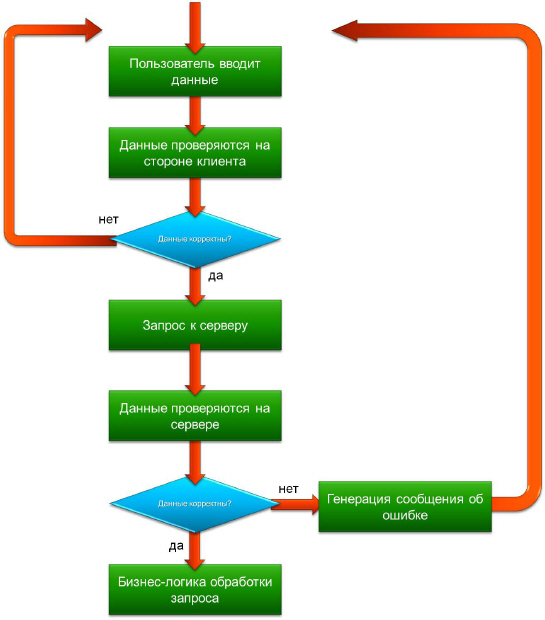
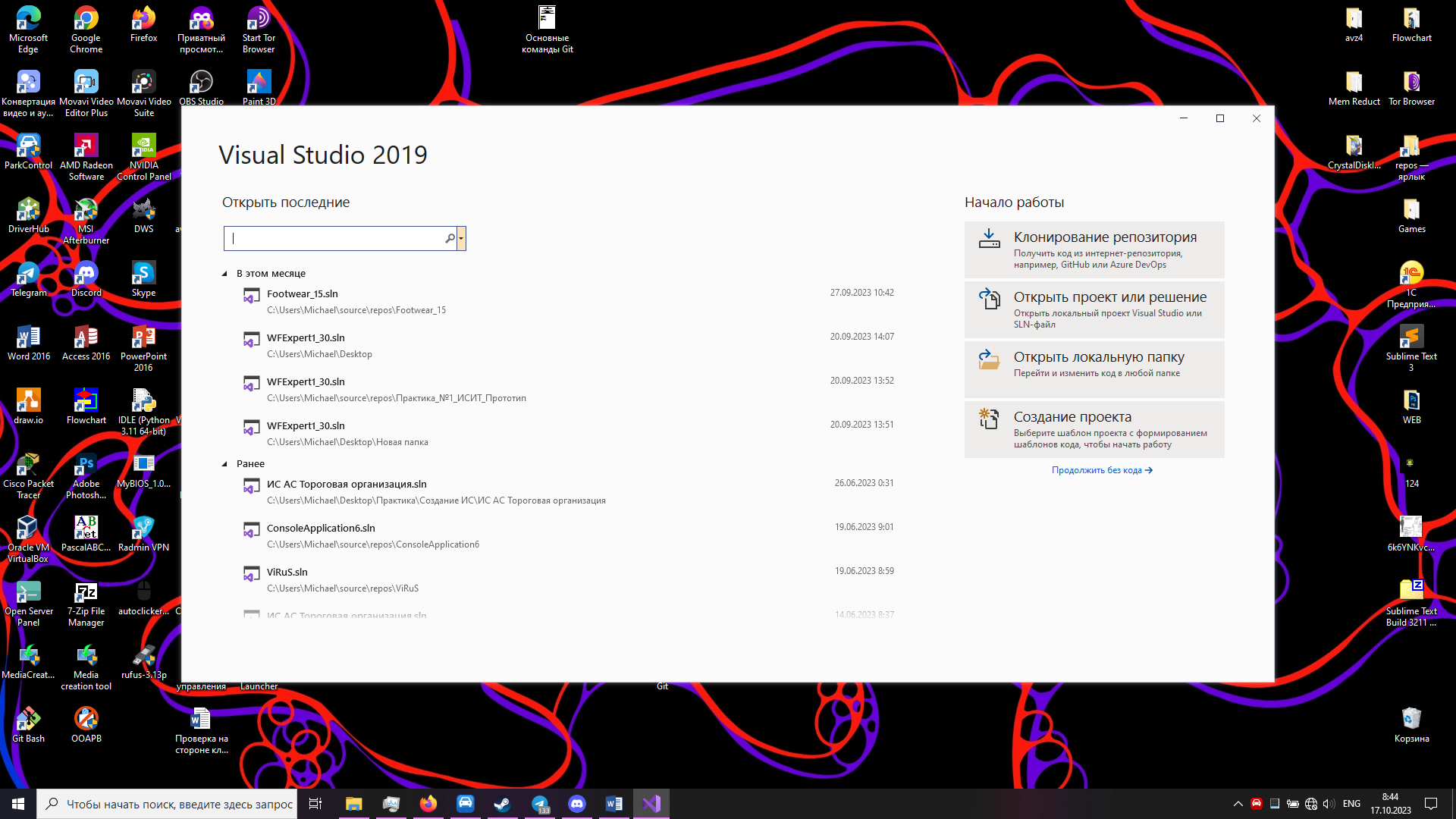


Рисунок – Общий алгоритм валидации данных, введенных пользователем можно представить следующим образом

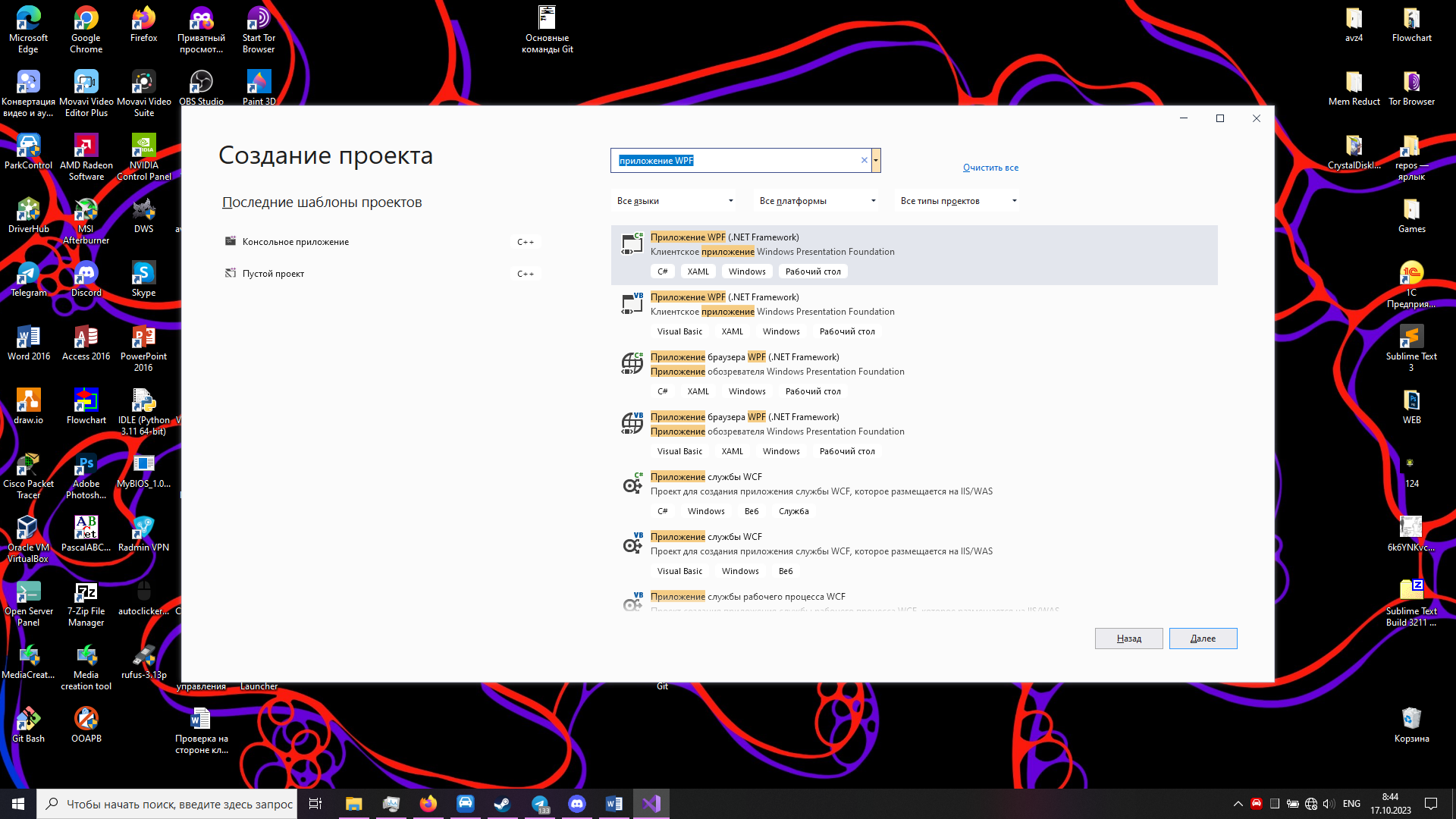
Таким образом, **серверная валидация** – это необходимый процесс, который должен выполнятся всякий раз, когда обрабатывается пользовательский ввод, а **клиентская валидация** – это опциональный компонент, который позволяет избавить пользователя от необходимости лишних обращений к серверу и повысить удобство работы с приложением.

**Порядок выполнения работы**

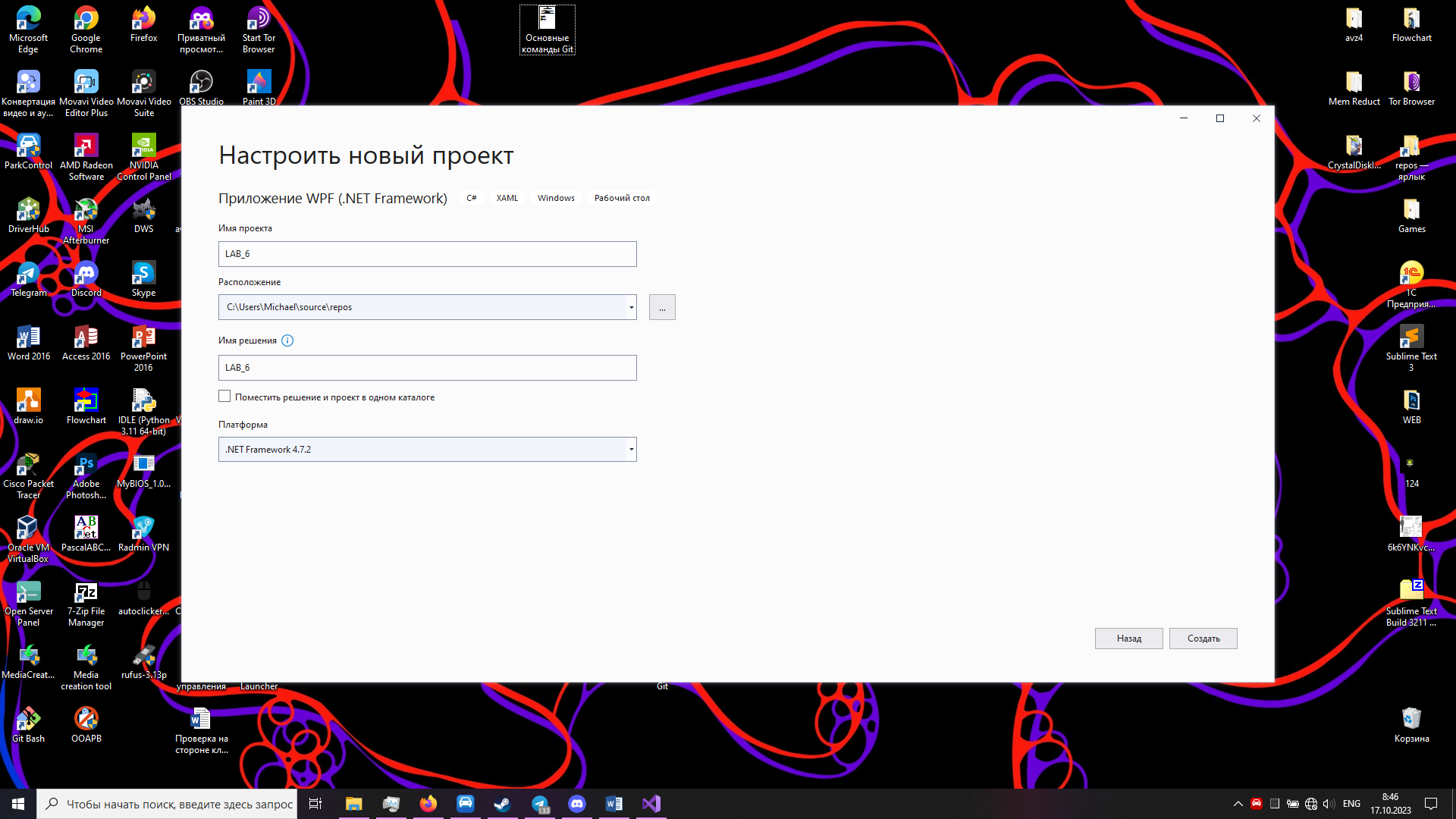
1. Запускаем Visual Studio и нажимаем на «Создание проекта»



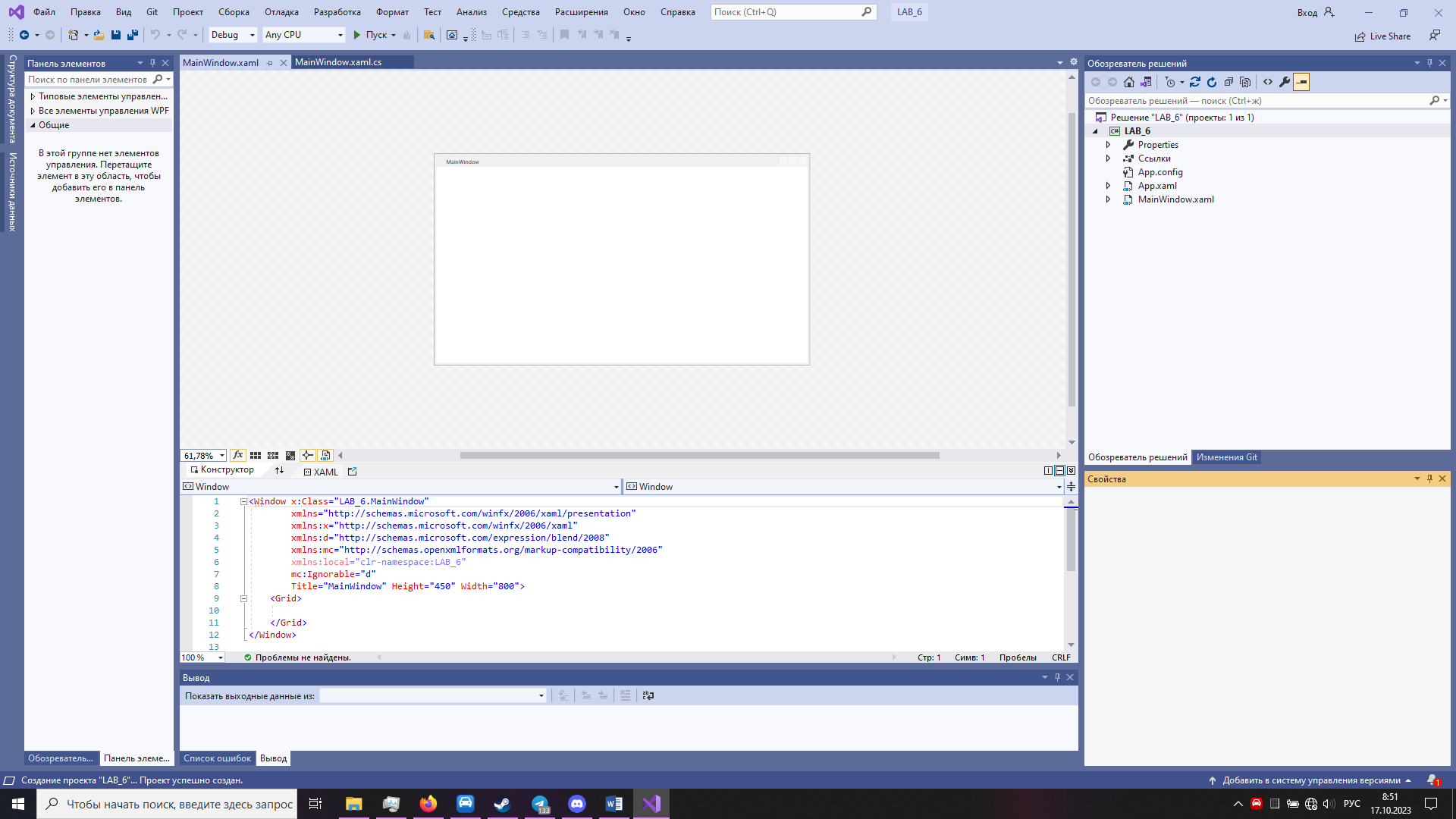
1. В поиске пишем «Приложение WPF» и выбираем приложение WPF с языком С#



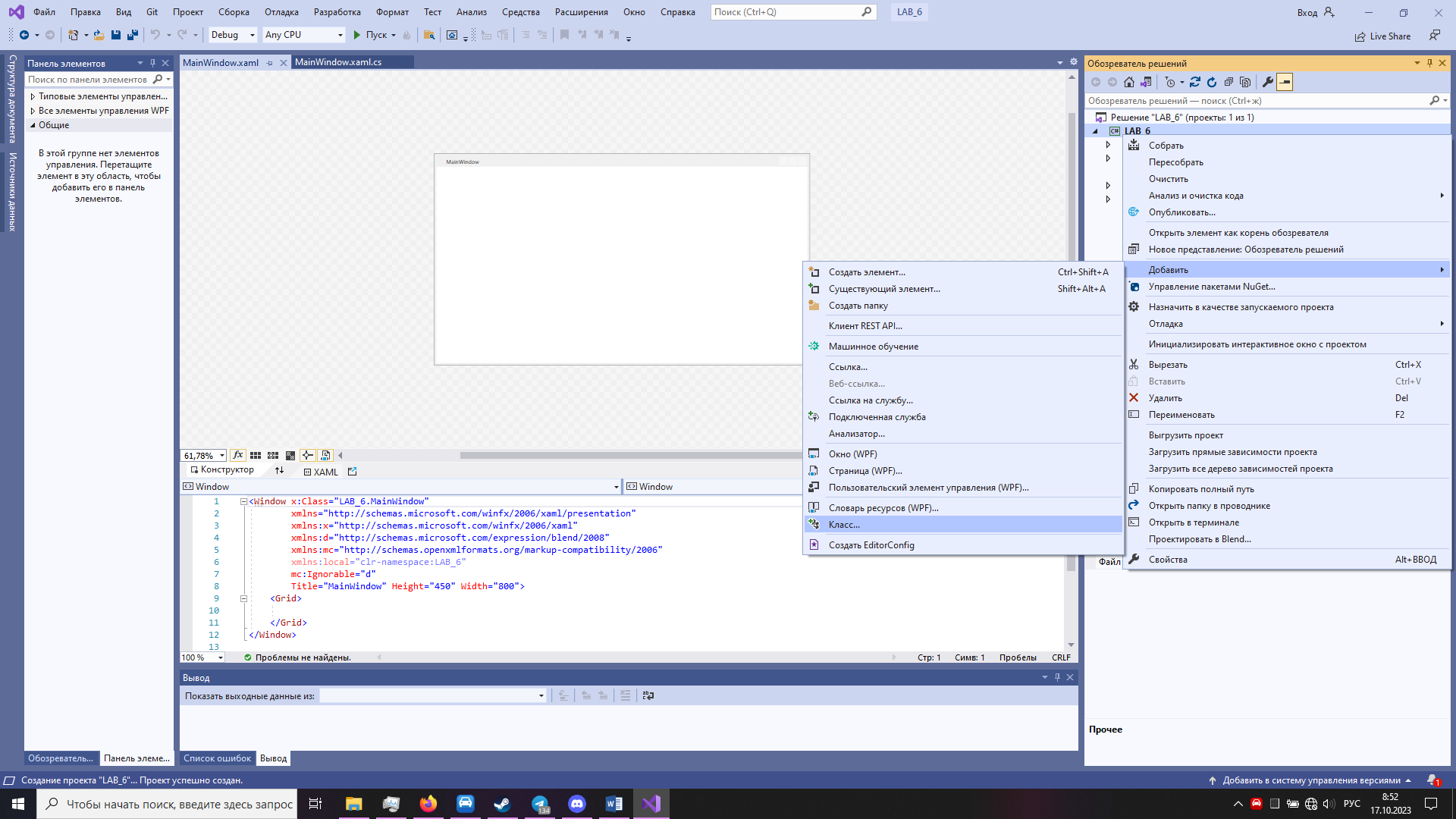
1. Задаем имя проекту и выбираем папку для сохранения



1. Откроется окно с проектом, в файле с расширением .xaml будет прописываться разметка формы

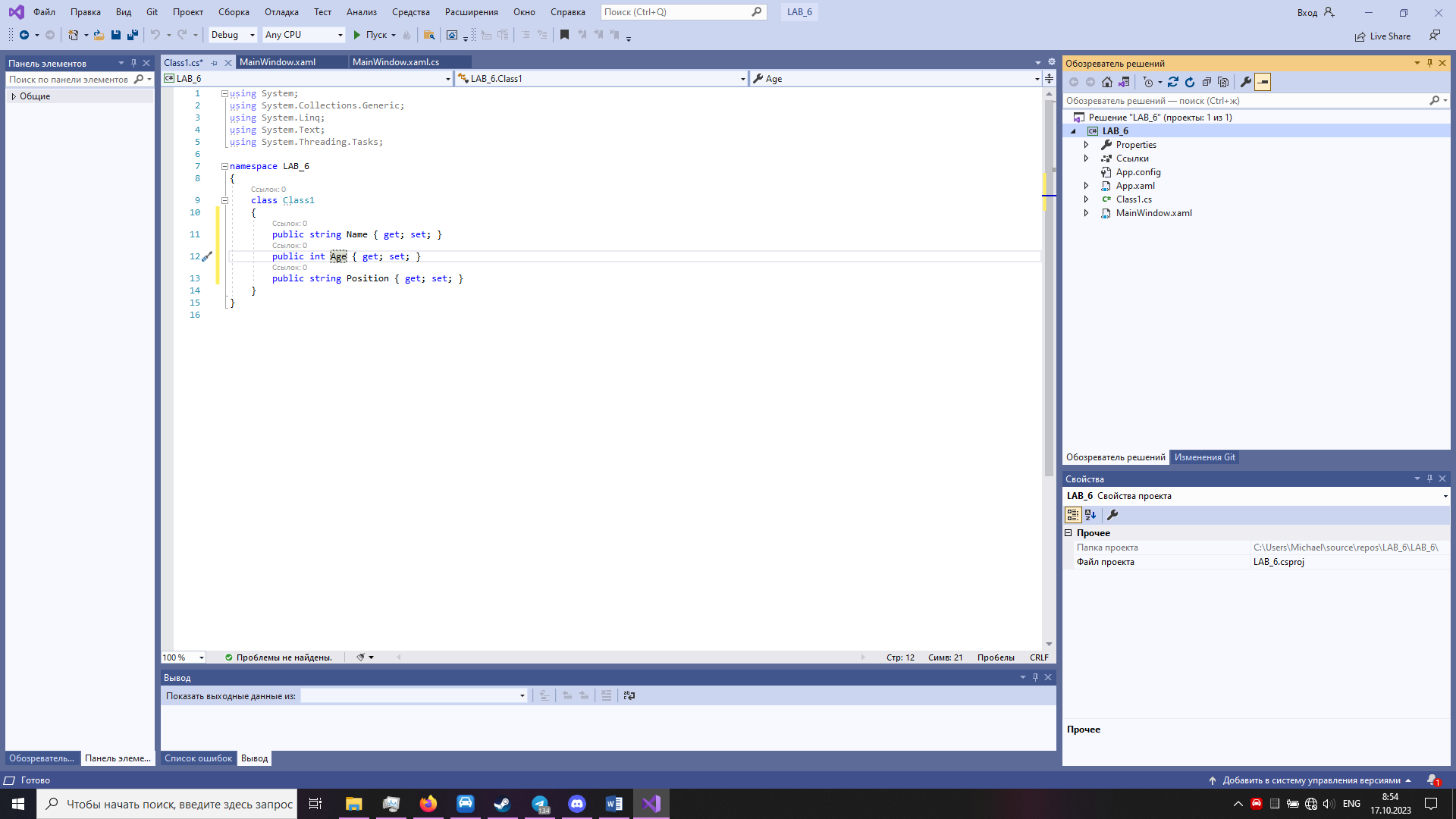


1. Создадим класс с переменными. Во вкладке «Обозреватель решений» правой кнопкой мыши нажимаем на имя проекта (оно выделено полужирным шрифтом) и в контекстном меню выбираем пункт «Добавить» - «Класс». Этот класс представляет человека и предполагает три свойства: имя, возраст и должность. Понятно, что возраст должен представлять числовое значение. Однако пользователи могут ввести что угодно. Мы можем обрабатывать ввод с клавиатуры, а можем воспользоваться классом **ExceptionValidationRule**, который в случае неудачи преобразования строки в число установит красную границу вокруг текстового поля.

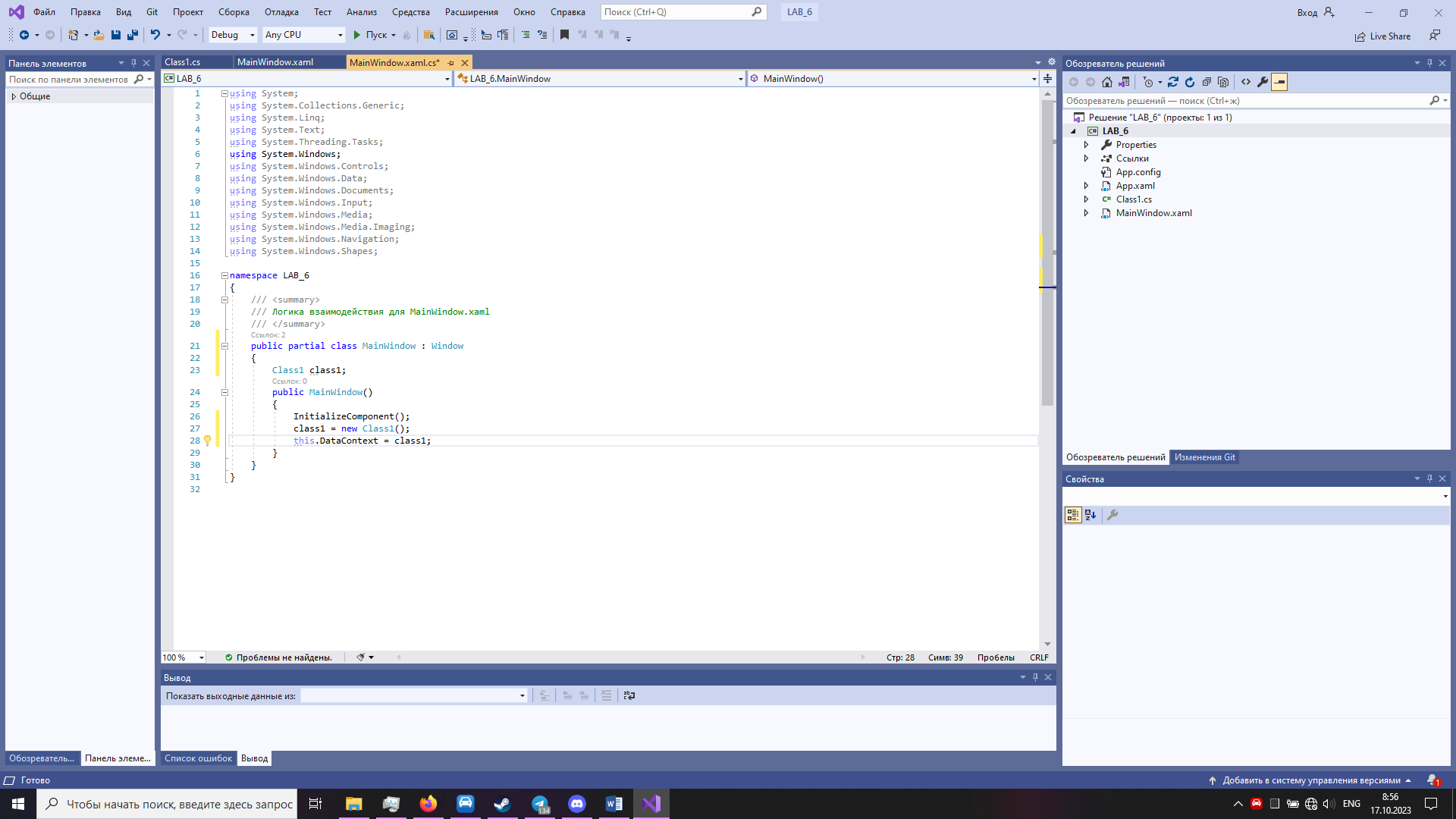




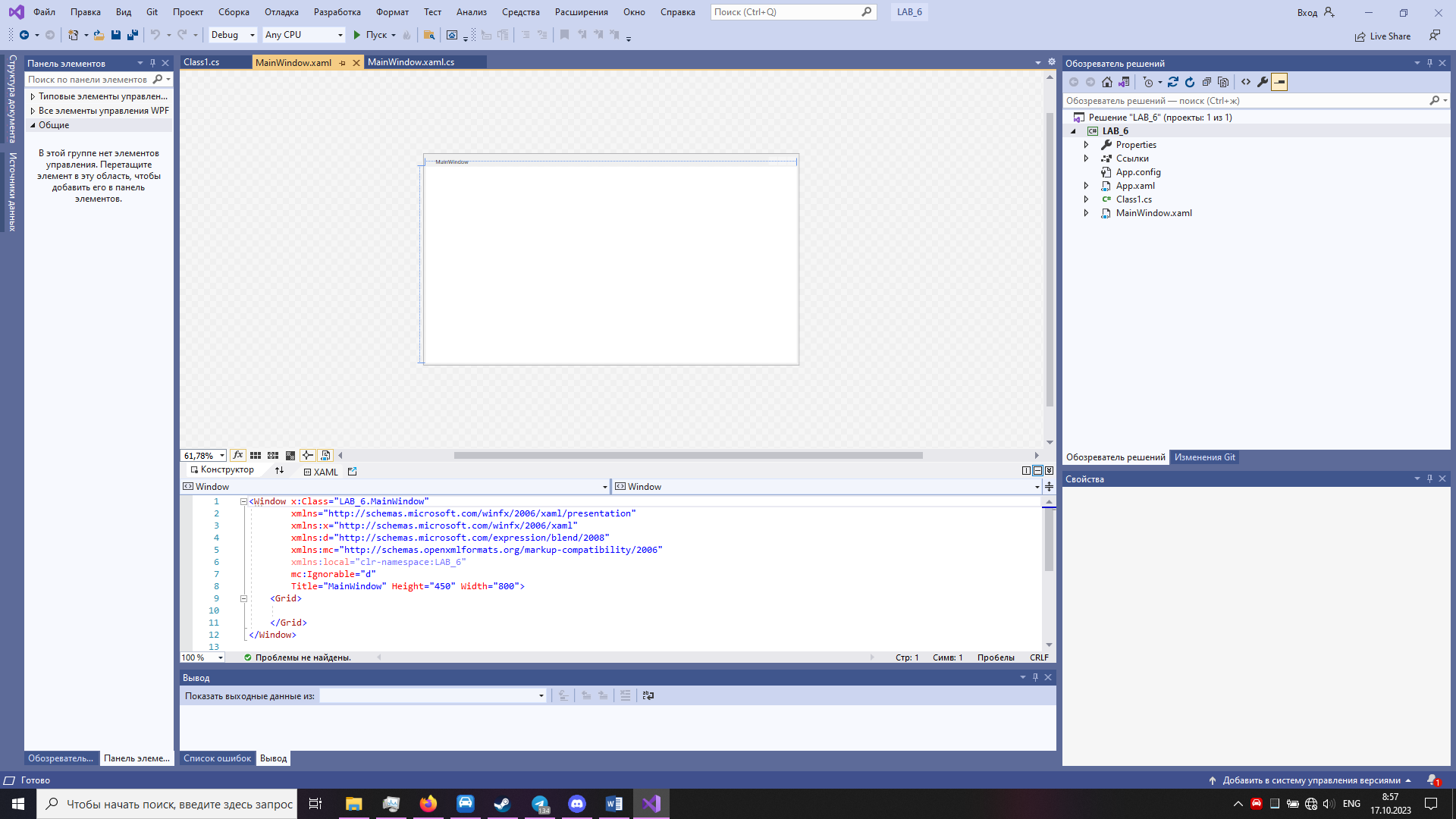
И объявляем следующие переменные



1. Переходим в класс MainWindow.xaml.cs и создадим в файле кода объект нашего класса Class1 и установим контекст данных окна

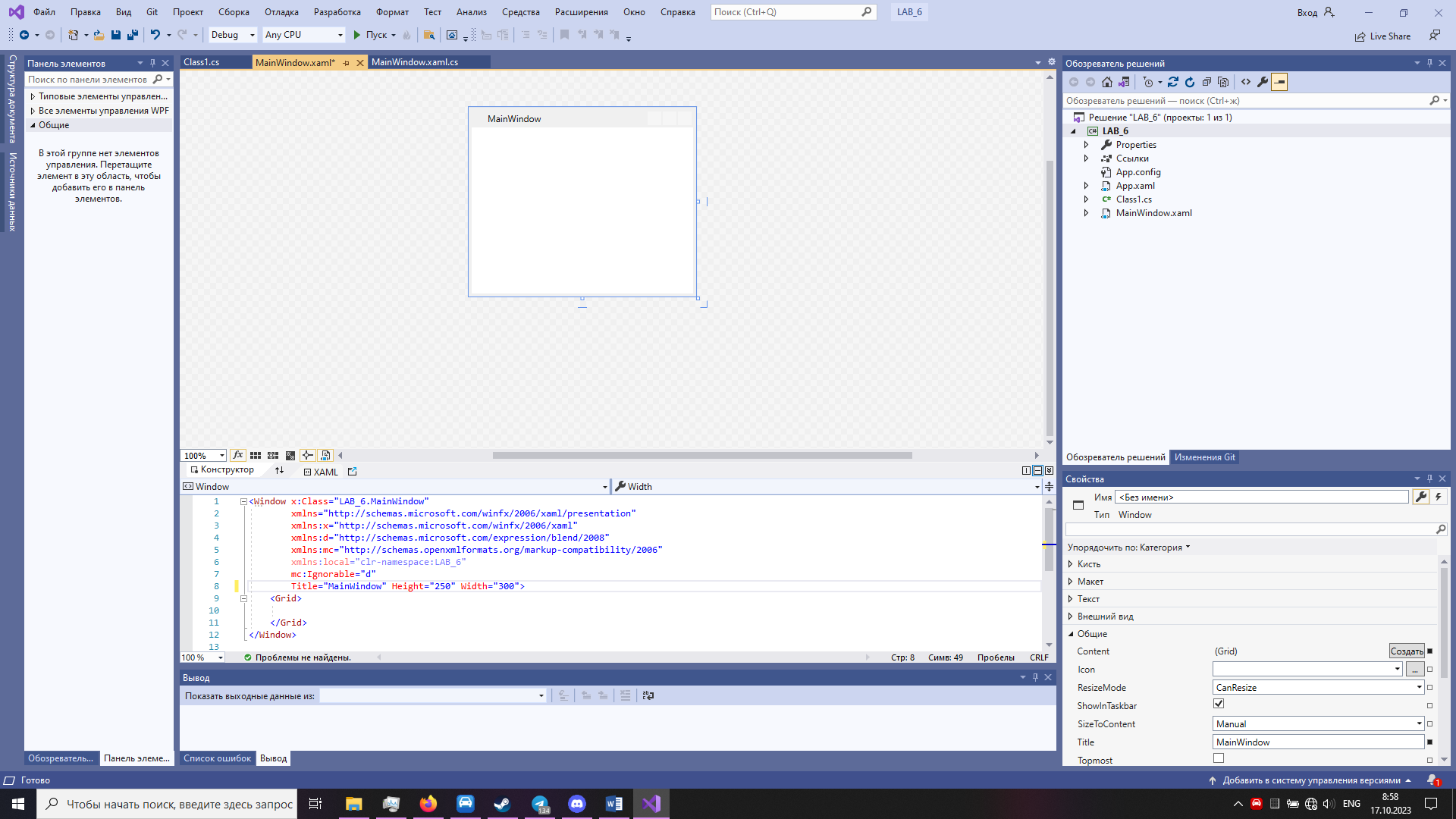


1. Теперь установим привязку в xaml-коде. Переходим в MainWindow.xaml, в которой находится разметка окна.

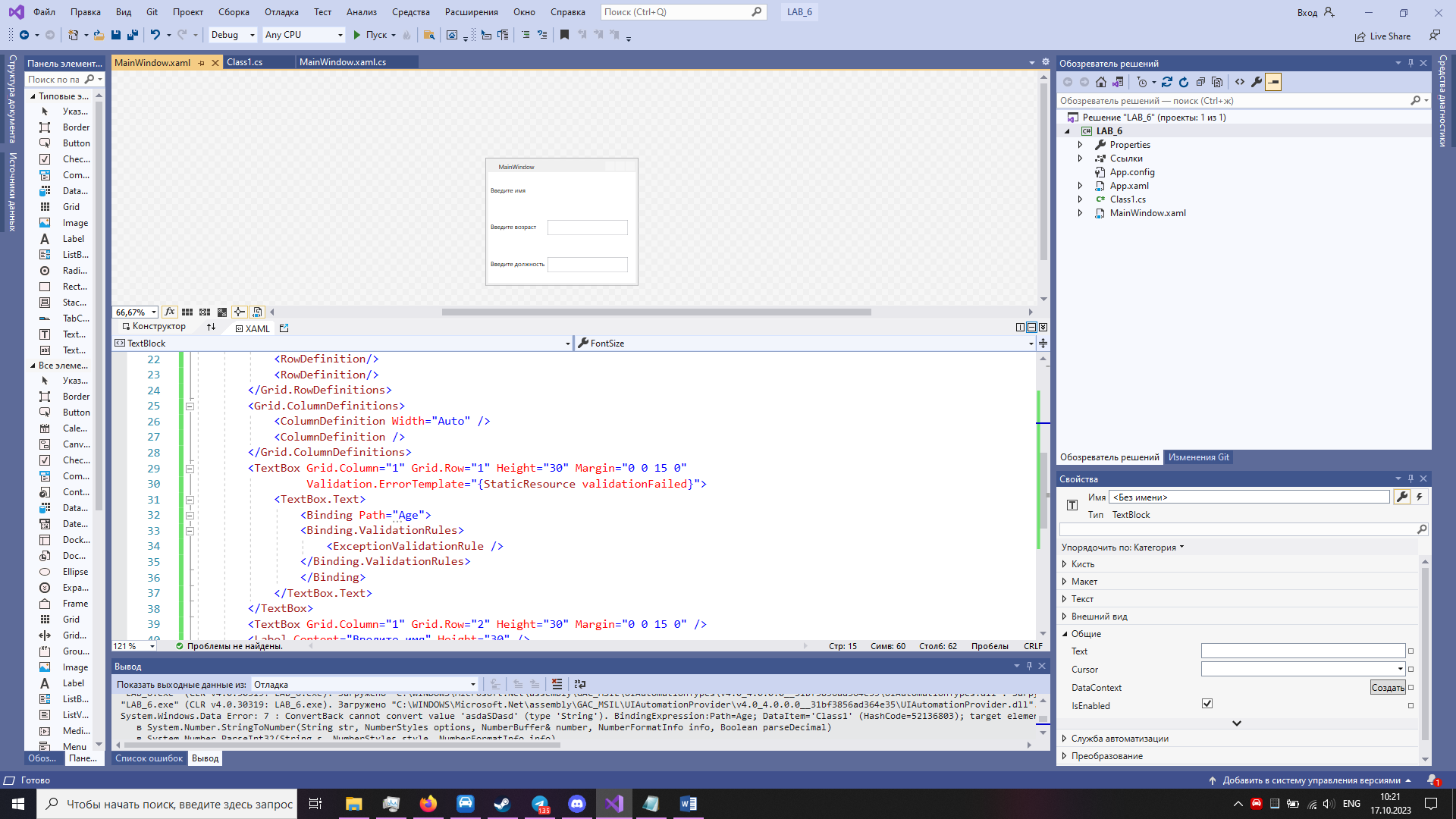


1. Теперь прописываем следующий код:

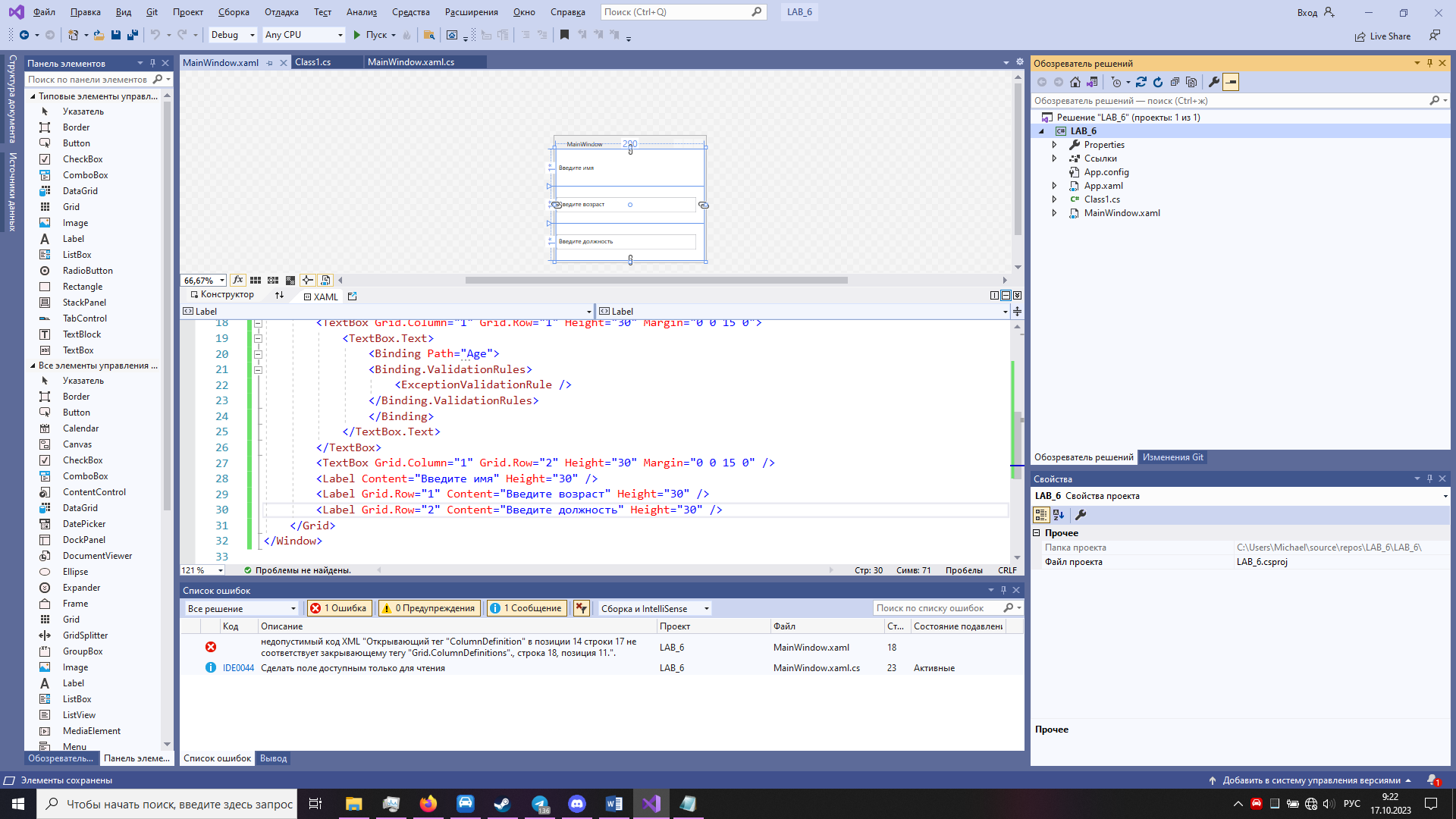
* Для начала изменяем высоту и ширину окна



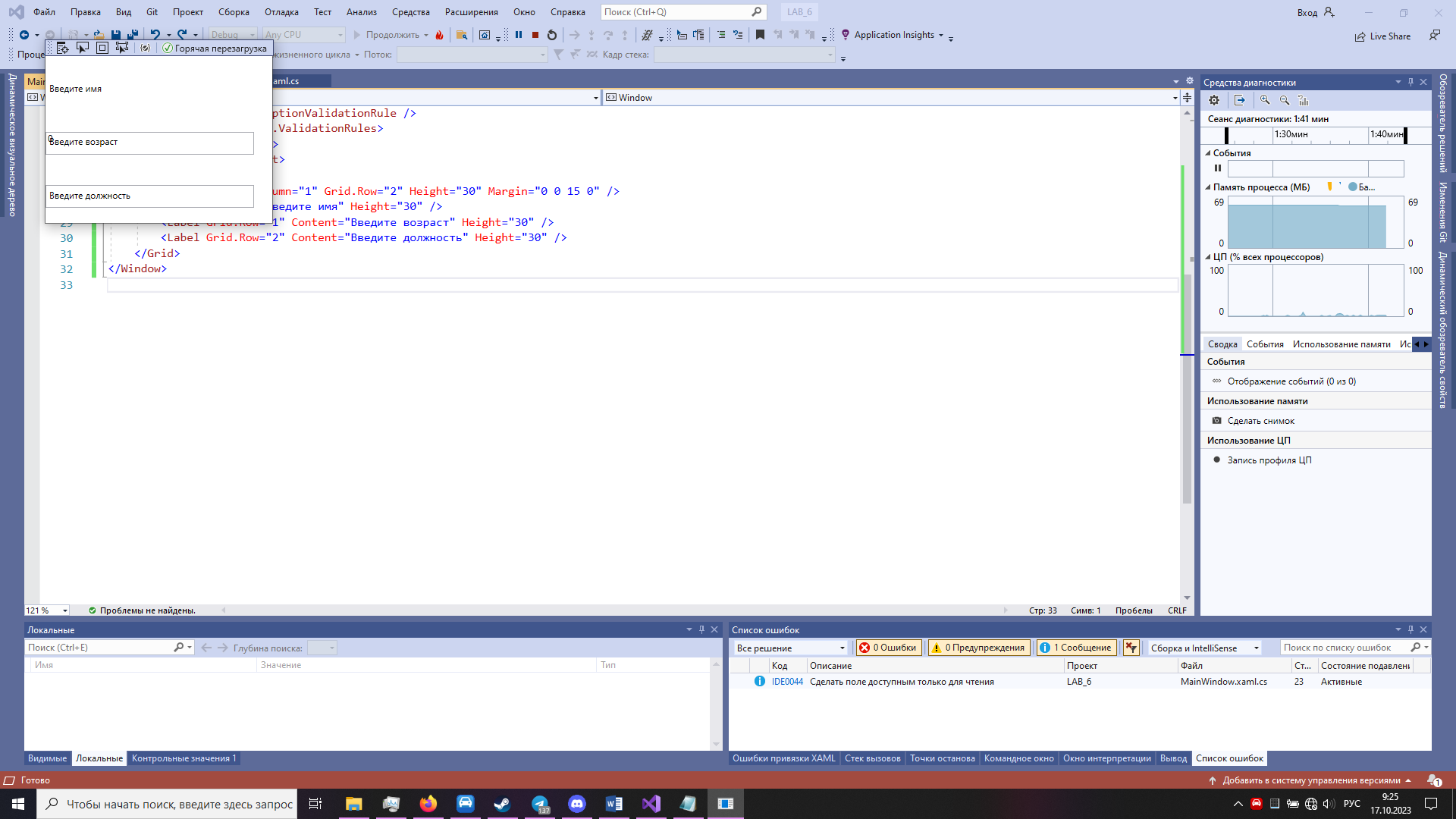
* Далее прописываем разметку для объектов

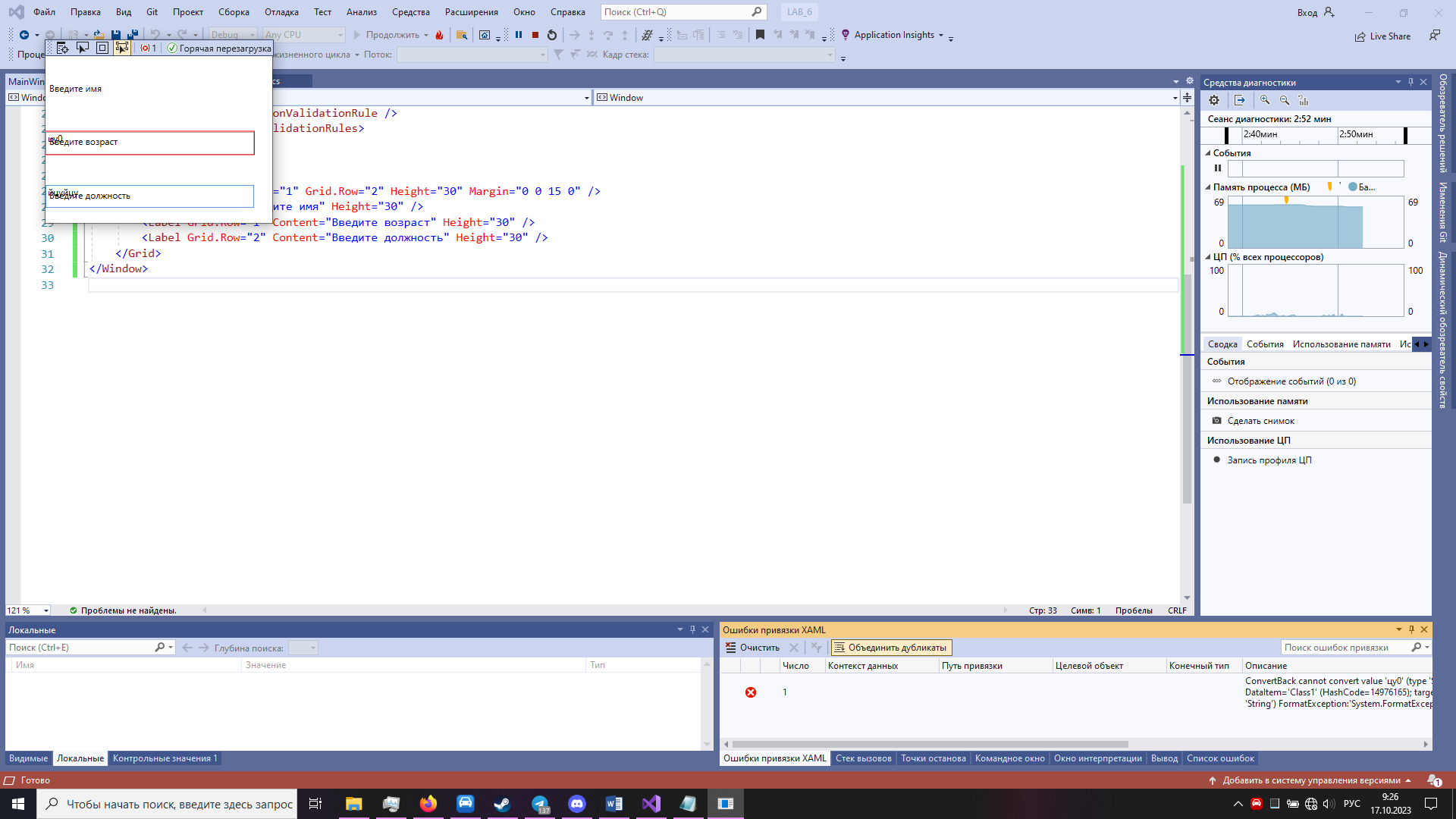


* Далее прописываем размещение объектов в окне

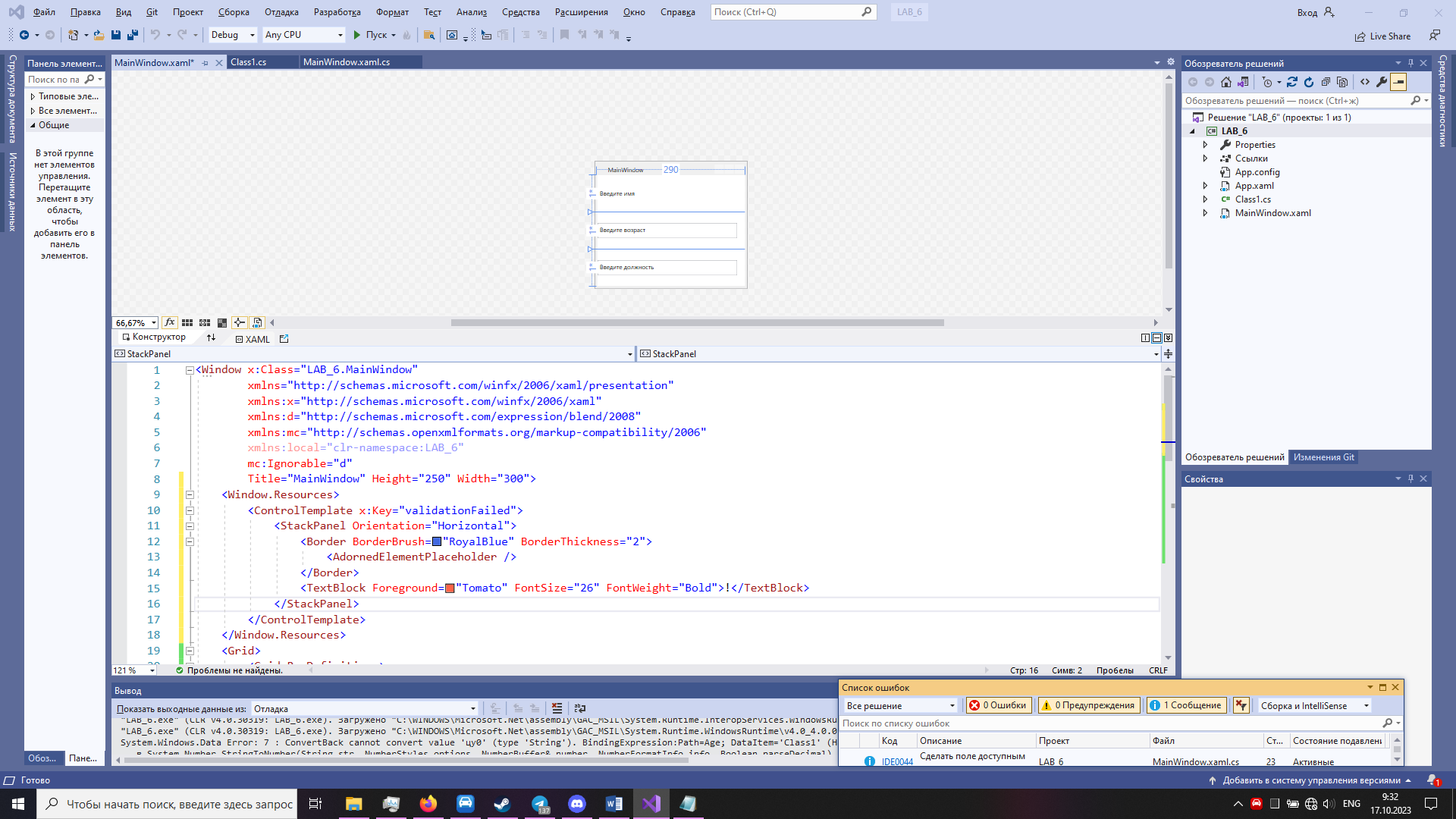


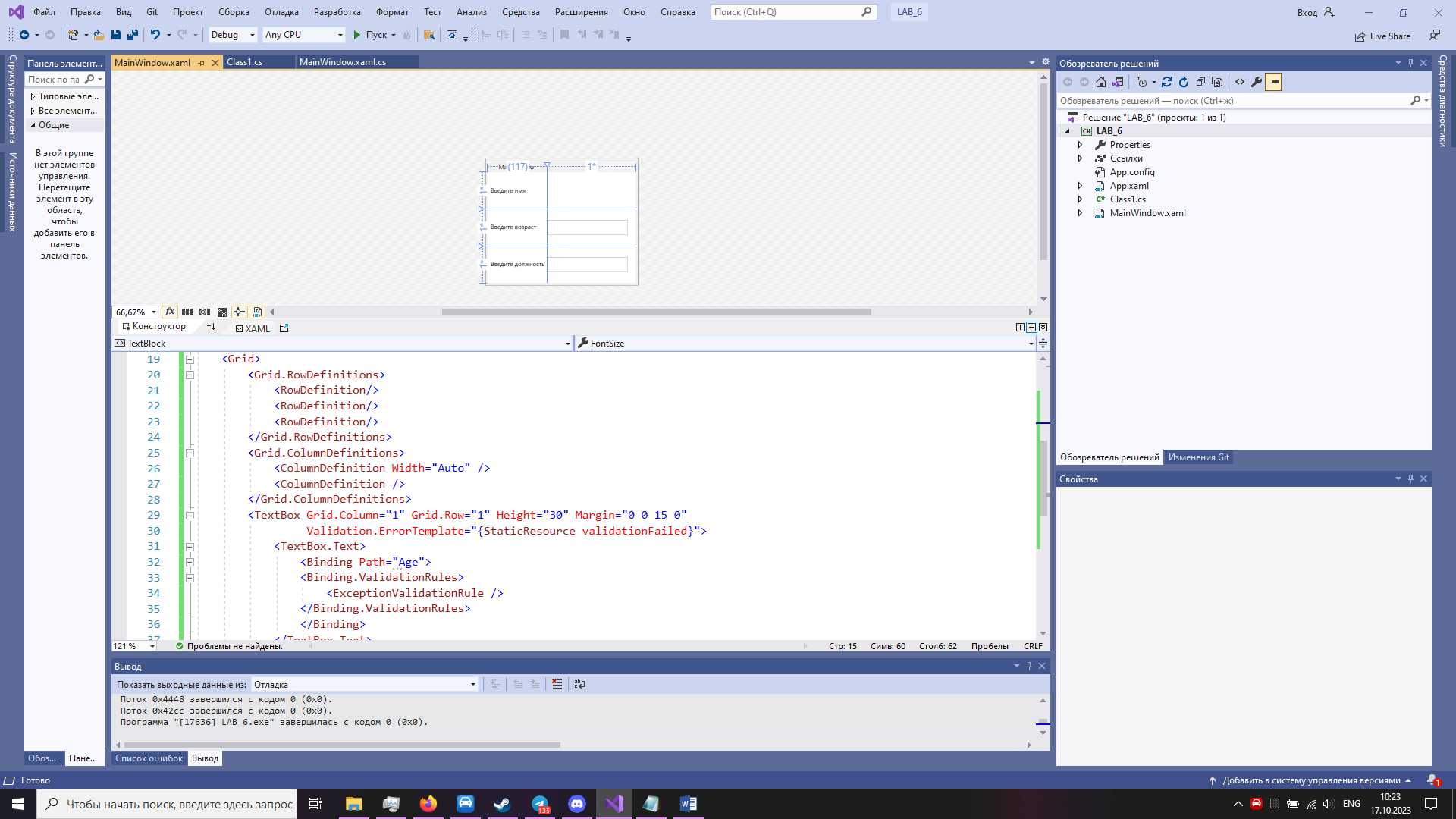
1. В данном случае мы задаем объект **Binding** для свойства **Text**. Данный объект имеет коллекцию правил валидации вводимых данных - **ValidationRules**. Эта коллекция принимает только одно правило валидации, представленное классом **ExceptionValidationRule**. Запустим приложение на выполнение и попробуем ввести в текстовое поле какое-нибудь нечисловое значение. В этом случае текстовое поле будет обведено красным цветом, указывая на то, что в вводимых данных имеются ошибки.





1. Настройка внешнего вида при ошибке валидации. Мы можем сами управлять через шаблоны отображением ошибки ввода. В предыдущем случае граница текстового поля при ошибке окрашивалась в красный цвет. Для настройки данного действия нужно использовать элемент AdornedElementPlaceholder





1. С помощью свойства **Validation.ErrorTemplate** мы получаем шаблон, который будет отрабатывать при ошибке валидации. Этот шаблон, определенный выше в ресурсах окна, определяет границу элемента ввода, а также отображает рядом с ним восклицательный знак. Запустим приложение и попробуем ввести в текстовое поле какое-нибудь некорректное значение. В результате сработает наш шаблон

