**Министерство образования и науки Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**

**МЕХАНИКИ И ОПТИКИ»**

**Факультет «ИКТ»**

**Дисциплина**

«Создание программного обеспечения инфокоммуникационных систем»

**Лабораторная работа 3**

**«**Реализация иерархии классов на языке С#**»**

**Выполнил:**

студент группы K3415

Соловьев Н. С.

**Проверил**:

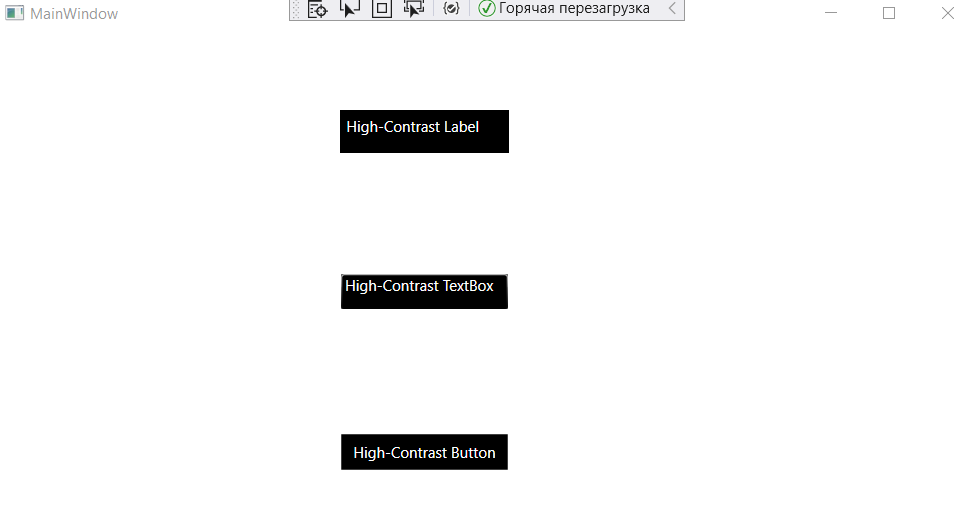
Осипов Н. А.

Санкт-Петербург

2021

**Практическое задание 8. Использование стилей и анимации Упражнение 1. Использование стилей для создания высококонтрастных элементов**

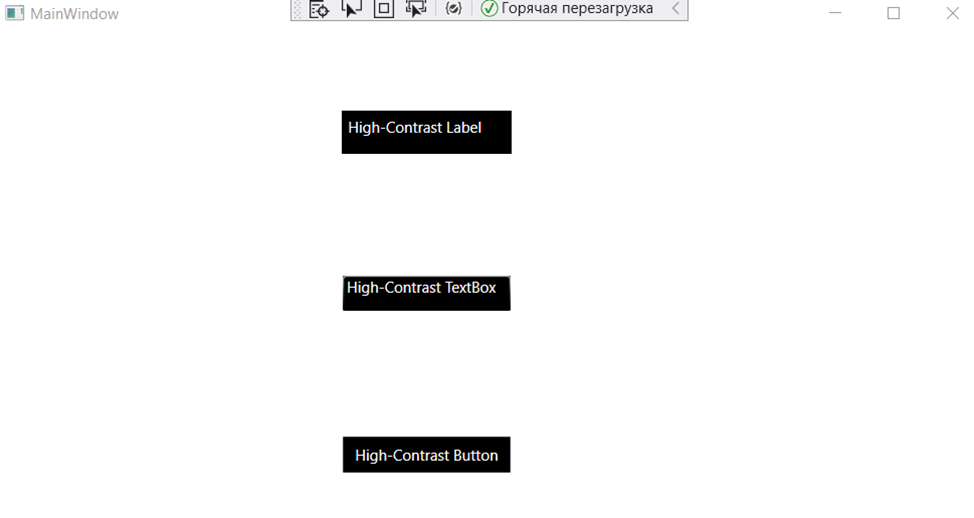
1. Создадим новый проект WPF Application с именем «Style01».
2. Определим стили для элементов Button, Label и TextBox в секции Window.Resources.
3. Изменим XAML разметку формы добавив элементы Button, Label и TextBox.
4. Построим и запустим приложение, чтобы протестировать его работу и проверить, что созданные стили применились к добавленным элементам.



*Рисунок 1 - тестирование приложения "Style01"*

**Практическое задание 8. Использование стилей и анимации Упражнение 2. Использование триггеров для улучшения внешнего вида интерфейса**

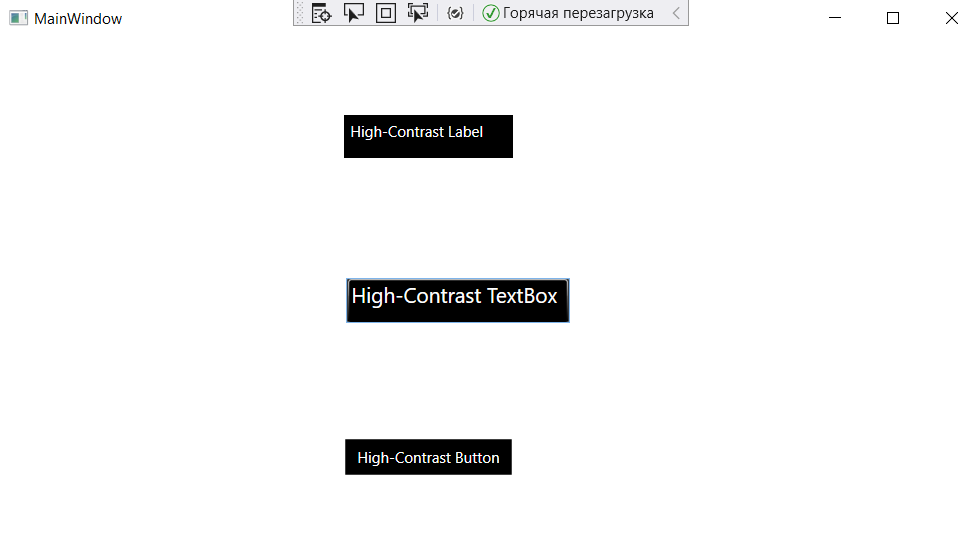
1. Откроем проект предыдущего упражнения.
2. Добавим для стилей элементов **TextBox, Button** и **Label** секции Style.Triggers и триггеры для свойства **IsMouseOver**.
3. Построим и запустим приложение, чтобы протестировать его работу.



*Рисунок 2 - тестирование приложения, проверка интерактивного отклика при* *наведении.*

**Практическое задание 8. Использование стилей и анимации Упражнение 3. Анимация высококонтрастных стилей**

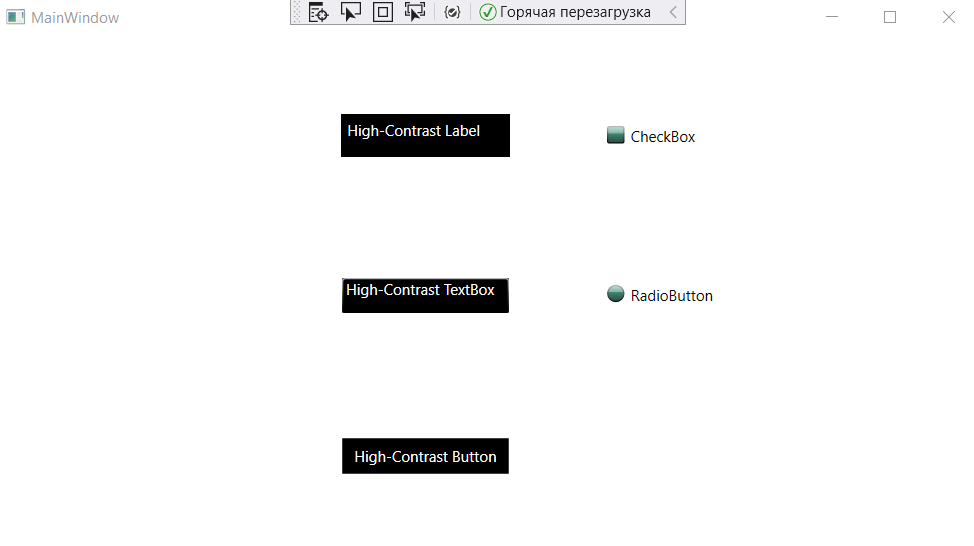
1. Откроем проект предыдущего упражнения.
2. Заменим секции Setter секциями **Trigger.EnterActions** и **Trigger.ExitActions**. И определим параметры анимации в соответствии с заданием.
3. Построим и запустим приложение, чтобы протестировать его работу



*Рисунок 3 - тестирование приложения, проверка интерактивного отклика при* *наведении. Происходит плавное увеличение размеров элемента.*

**Практическое задание 8. Использование стилей и анимации Упражнение 4. Добавление готовых тем в приложение**

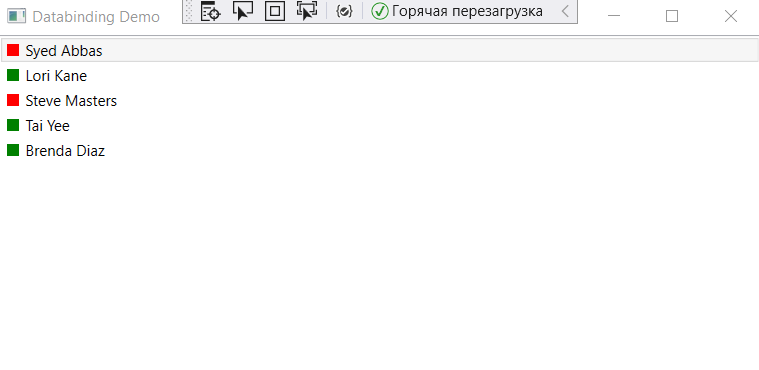
1. Откроем проект предыдущего упражнения.
2. Скачаем и добавим в проект XAML файл темы, меняющий внешний вид базовых элементов.
3. В файле App.xaml, добавим ссылку на желаемую тему в секции Application.Resources.
4. Добавим несколько новых элементов (CheckBox, RadioButton), чтобы продемонстрировать применение добавленных стилей.
5. Построим и запустим приложение, чтобы протестировать его работу



*Рисунок 4 - тестирование приложения*

**Практическое задание 9. Использование стилей и анимации Упражнение 1. Создание привязки данных с помощью конструктора WPF**

1. Создадим новый проект WPF Application с именем «DataBindingDemo».
2. Добавим в проект новый класс с именем **Student**.
3. В классе Student определим два автоматических свойства **StudentName** и **IsEnrolled**.
4. В пространстве имен **DataBindingDemo** определим новый класс **StudentList** производный от класса ObservableCollection<Student>. Объект данного класса будет служить источником данных для элемента интерфейса **ListBox**.
5. В пространстве имен **DataBindingDemo** определим новый класс **BoolToBrushConverter** и отредактируем в соответствии с заданием лабораторной работы, добавив методы Convert, ConvertBack и атрибут ValueConversion.
6. Отредактируем файл MainWindow.xaml, определив набор объектов класса **Student** в коллекции **StudentList**, и шаблон для отображения элементов списка **ListBox.**
7. С помощью конструктора WPF настроим привязку данных для элемента **ListBox**.
8. Выберем шаблон DataTemplate для преобразования данных и их более наглядного отображения.
9. Построим и запустим приложение, чтобы протестировать его работу

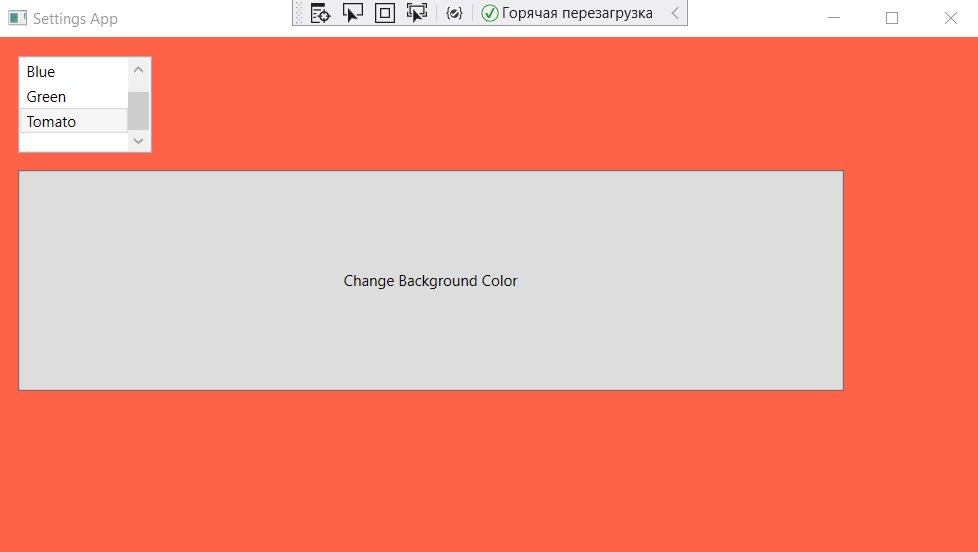


*Рисунок 5 - тестирование приложения*

**Практическое задание 10. Использование параметров приложения**

**Упражнение 1. Работа с параметрами**

1. Создадим новый проект WPF Application с именем «Param».
2. Добавим два параметра в приложение **ApplicationName** и **BackgroundColor.**
3. Отредактируем XAML представление формы, добавив список **ListBox** с четырьмя элементами, описывающими доступные цвета, и кнопку смены цвета фона.
4. Добавим обработчик события Click для созданной кнопки, производящий смену цвета фона окна приложения.
5. Добавим присвоение свойствам **Title** и **Background** параметров, определенных ранее, в конструкторе класса MainWindow.
6. Построим и запустим приложение, чтобы протестировать его работу



*Рисунок 6 - тестирование приложения.*

**Лабораторная работа 1 (Задание). Разработка приложения WPF**

**Упражнение 1. Работа с параметрами**

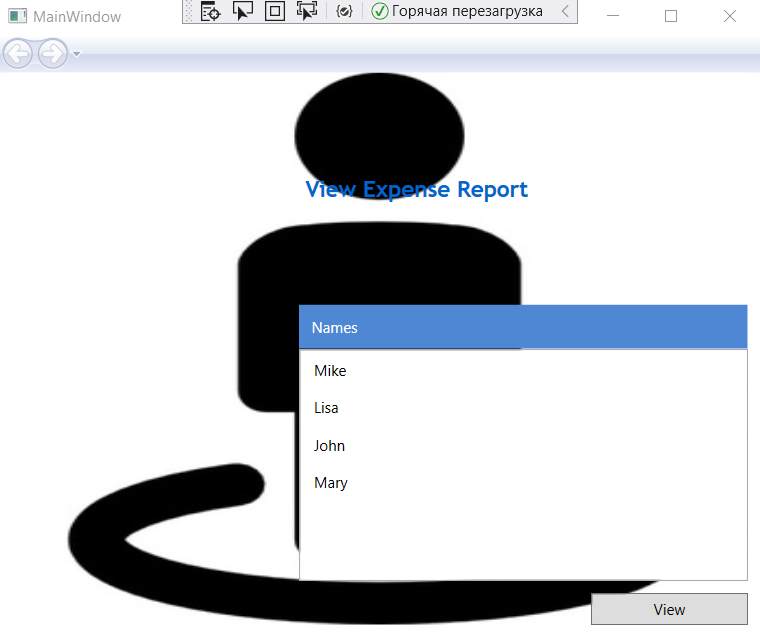
1. Создадим новый проект WPF Application с именем «ExpenseIt».
2. Заменим элемент Window на NavigationWindow.
3. Отредактируем и определим свойства элемента NavigationWindow в соответствии с заданием.
4. Добавим в проект новые страницы с именами **ExpenseItHome** и **ExpenseReportPage**, настроим их параметры.
5. Добавим на страницу тестовое изображение и построим проект.



*Рисунок 7 - промежуточное тестирование приложения.*

**Упражнение 2. Реализация пользовательского интерфейса приложения**

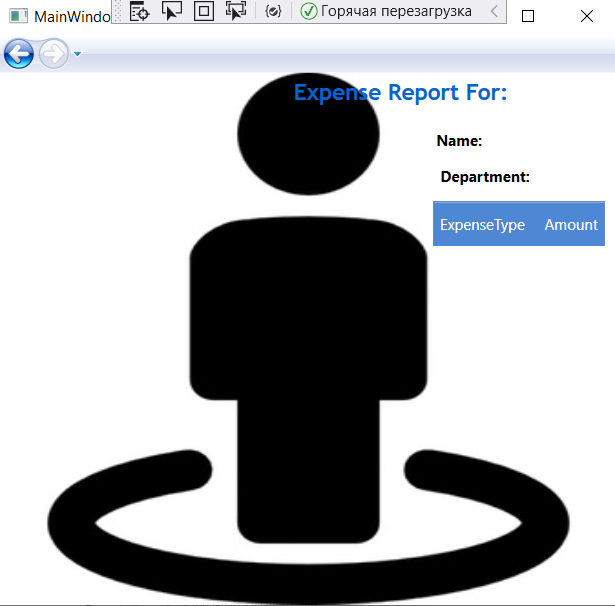
1. Отредактируем файл ExpenseItHome.xaml в соответствии с заданием, определив сетку компонентов и элементы пользовательского интерфейса.
2. Зададим добавленное ранее изображение в качестве фона для класса Grid.
3. Добавим обработчик события Click для кнопки View, открывающий новое окно просмотра информации о выбранном человеке.
4. Построим и проверим наше приложение.



*Рисунок 8 - промежуточное тестирование приложения*

**Упражнение 3. Создание пользовательского интерфейса отдельной страницы**

1. Отредактируем файл ExpenseReportPage.xaml в соответствии с заданием, определив сетку компонентов и элементы пользовательского интерфейса. Данная страница отвечает за отображения подробной информации о человеке, выбранном на главной странице.
2. Построим и проверим наше приложение.

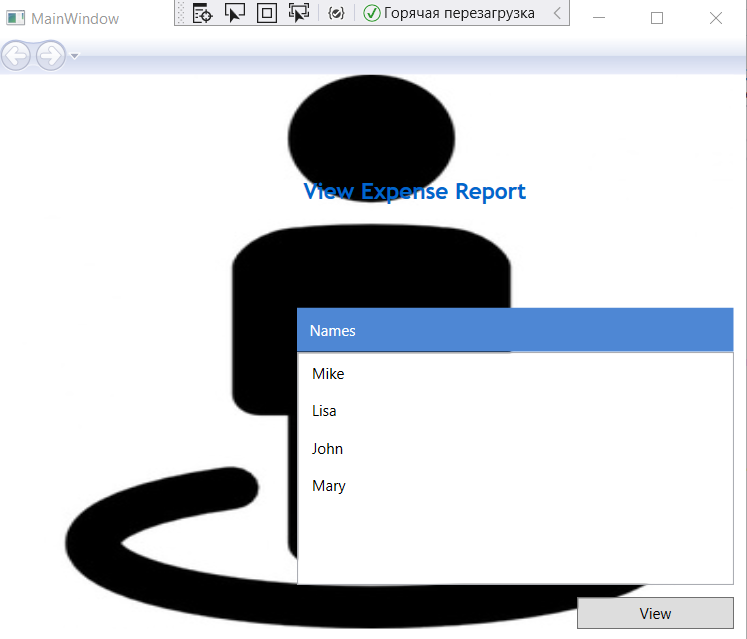


*Рисунок 9 - промежуточное тестирование приложения*

Как и ожидалось, после нажатия кнопки View происходит переход на пока не наполненную страницу информации о выбранном человеке.

**Упражнение 4. Использование стилей**

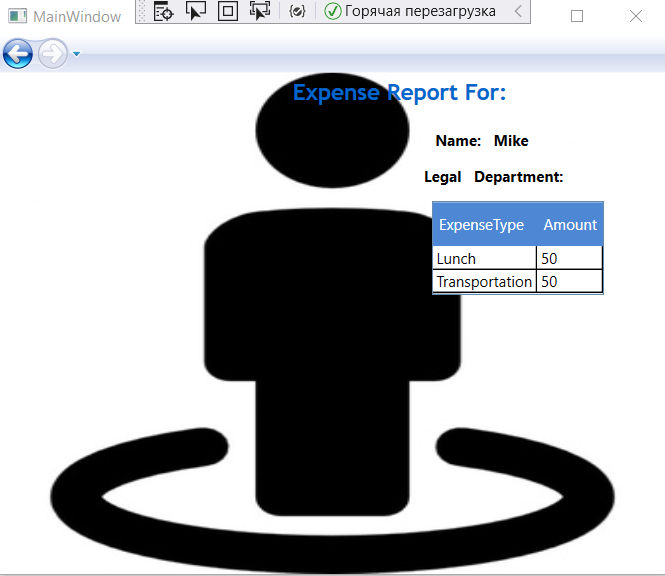
1. Определим стили для нашего приложения в секции Application.Resources файла App.xaml.
2. Применим описанные выше стили к элементам страниц ExpenseReportPage и ExpenseItHome.
3. Построим и проверим наше приложение.



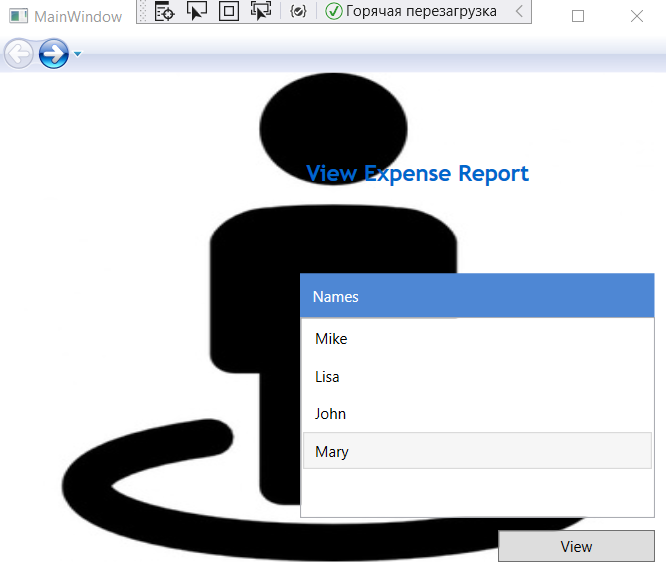
*Рисунок 10 - промежуточное тестирование приложения*

**Упражнение 5. Привязка данных к элементу управления**

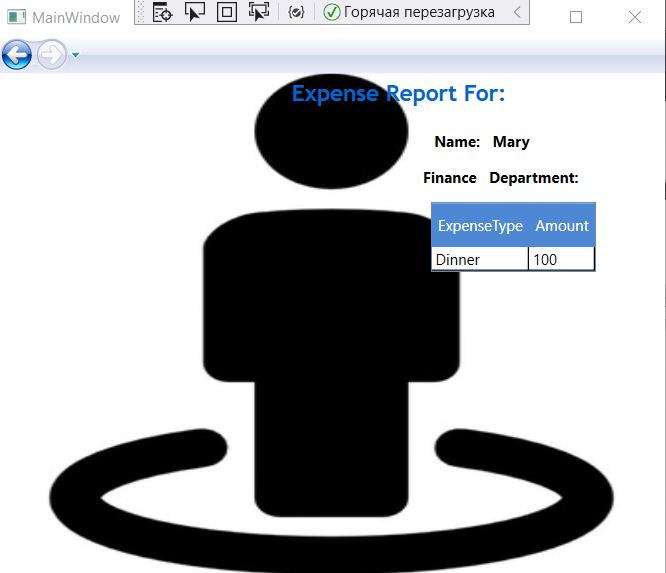
1. Добавим в проект папку с именем Data и файлом Data.xml, который будет выступать в качестве источника данных для нашего приложения.
2. Отредактируем файл ExpenseItHome.xaml, определив раннее созданный файл в качестве ресурса и задав шаблон для отображения элементов компонента ListBox.
3. Отредактируем файл ExpenseReportPage.xaml.cs, добавив в конструктор параметр data – информации о выбранном человеке.
4. Переопределим создание новой страницы в файле ExpenseItHome.xaml.cs.
5. Отредактируем код ExpenseReportPage.xaml для отображения
6. Построим и проверим наше приложение.



*Рисунок 11 - тестирование приложения. Страница 2.*

**

*Рисунок 12 - тестирование приложения. Страница 1.*



*Рисунок 13 - тестирование приложения. Страница 2.*

**Вывод**

**В данной лабораторной работе были рассмотрены:**

• Использование стилей для изменения внешнего вида элементов управления.

• Использование триггеров для реакции на поведение пользователя (наведение курсора мыши на элементы управления)

• Анимация стилей

• Использование сторонних тем для изменения внешнего вида элементов управления.

• Использование привязки данных к компонентам для их отображения.

• Использование преобразователей значений для редактирования отображения данных.

• Рассмотрена работа с параметрами приложения для сохранения состояния между сессиями.

• Рассмотрены механизмы создания и переходов между страницами в приложении.

**На практике были освоены следующие принципы окон WPF:**

• Окно WPF может содержать только один элемент.

• Чтобы разместить более одного элемента и создать практичный пользовательский интерфейс, нужно поместить в окно контейнер и добавлять элементы в этот контейнер.

• В WPF компоновка формируется с использованием разнообразных контейнеров.

• Каждый контейнер обладает собственной логикой компоновки

• WPF использует потоковую модель расположения (flow-based layout) для позиционирования элементов управления

• Потоковая модель расположения позволяет контролам WPF быть адаптированным к размеру окна, когда они начинают визуализацию

В лабораторной были изучены принципы добавления стилей и анимаций к элементам. По итогу выполнения, можно отметить, что WPF имеет богатые возможности, а использование сторонних библиотек и стилевых компонентов только расширяют их.

Многие компоненты WPF поддерживают привязку данных, что значительно упрощает наполнение и отображение информации.

Было реализовано сложное приложение с древовидным интерфейсом. Благодаря WPF реализация была значительно ускорена с помощью переиспользуемых компонентов с привязкой данных, что позволило не создавать отдельную страничку под каждый случай, а динамически обрабатывать имеющиеся данные