МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

СОВРЕМЕННЫЕ СИСТЕМЫ ПРОГРАММИРОВАНИЯ

Методические рекомендации к лабораторным работам для студентов направлений подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» дневной формы обучения

Могилев 2023

УДК 004.4
ББК 32.973
С56

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «17» октября 2023 г., протокол № 3

Составитель ст. преподаватель Н. В. Выговская

Рецензент канд. техн. наук, доц. С. К. Крутолевич

Приведены методические рекомендации к лабораторным работам по дисциплине «Современные системы программирования».

Учебное издание

СОВРЕМЕННЫЕ СИСТЕМЫ ПРОГРАММИРОВАНИЯ

Ответственный за выпуск	А. И. Якимов
Корректор	И. В. Голубцова
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Цель методических рекомендаций к выполнению лабораторных работ по дисциплине «Современные системы программирования» заключается в овладении студентами практических навыков разработки приложений в среде Microsoft Visual Studio на разных платформах, используемых в современных технологиях программирования, а также приобретение навыков разработки мобильных кросплатформенных приложений на фреймворке React Native.

Цель изучения дисциплины – подготовка студентов к использованию современных технологий и программных средств как профессионального инструмента для решения научных и практических задач.

Дисциплина «Современные системы программирования» является неотъемлемой частью знаний инженера-программиста и связана с разработкой современных программных систем и комплексов.

Полученные при изучении дисциплины знания и навыки могут быть востребованы в курсовом и дипломном проектировании и станут инструментом для реализации собственных проектов.

Общие требования ко всем лабораторным работам.

Отчёт включает в себя:

- титульный лист;
- цель работы;
- коды программ;
- скриншот с выполнением программы с подписями рисунков;
- вывод по лабораторной работе (чему научились).

1 Лабораторная работа № 1. Функциональное программирование

Цель работы: приобрести навыки работы с интерпретатором языка Haskell; получить представление об основных типах языка Haskell; научиться определять простейшие функции.

Порядок выполнения работы

1 Изучить основные базовые принципы и методы функционального программирования.

2 Выполнить практические задания.

Основные теоретические положения

Основы работы с интерпретатором Hugs [1].

Для выполнения лабораторных работ будет использоваться интерпретатор языка Haskell. Существует несколько реализаций интерпретатора; в настоящем курсе будет использоваться интерпретатор Hugs (это название является аббревиатурой слов Haskell users' Gofer system. Gofer – название языка программирования, который был одним из предшественников Haskell).

Prelude>.

Вообще, перед символом > выводится имя последнего загруженного модуля. Если приглашение интерпретатора изменено на **Main>**, значит, загружен какой-то модуль по умолчанию. Дело в том, что если не указано имя модуля, считается, что оно равно Main. Перед началом работы следует выгрузить лишние модули. Для этого зайдите в меню **File -> Module Manager** и удалите файл с именем Hugs.hs. Если приглашение интерпретатора изменилось на **Prelude>** , значит, все сделали верно.

После вывода приглашения можно вводить выражения языка Haskell либо команды интерпретатора. Команды интерпретатора отличаются от выражений языка Haskell тем, что начинаются с символа двоеточия (**:**). Примером команды интерпретатора является команда **:quit**, по которой происходит завершение работы интерпретатора. Команды интерпретатора можно сокращать до одной буквы; таким образом, команды **:quit** и **:q** эквивалентны.

1 Типы.

Программы на языке Haskell представляют собой выражения, вычисление которых приводит к значениям. Каждое значение имеет тип. Интуитивно тип можно понимать просто как множество допустимых значений выражения. Для того чтобы узнать тип некоторого выражения, можно использовать команду интерпретатора :type (или :t). Кроме того, можно выполнить команду :set +t для того, чтобы интерпретатор автоматически печатал тип каждого вычисленного результата.

2 Арифметика.

Интерпретатор Hugs можно использовать для вычисления арифметических

выражений. При этом можно использовать операторы +, -,*, / (сложение, вычитание, умножение и деление) с обычными правилами приоритета. Кроме того, можно использовать оператор ^ (возведение в степень). Таким образом, сеанс работы может выглядеть следующим образом.

3 Кортежи.

В языке Haskell пару можно задать, перечислив компоненты через запятую и взяв их в скобки: (5,3). Компоненты пары не обязательно должны принадлежать одному типу: можно составить пару, первым элементом которой будет строка, а вторым – число и т. д. Например, пара (5,3) имеет тип (Integer, Integer); пара (1, 'a') принадлежит типу (Integer, Char). Можно привести и более сложный пример: пара ((1,'a'),1.2) принадлежит типу ((Integer,Char),Double). Проверьте это с помощью интерпретатора.

4 Списки.

В отличие от кортежей, список может хранить произвольное количество элементов. Чтобы задать список в Haskell, необходимо в квадратных скобках перечислить его элементы через запятую. Все эти элементы должны принадлежать одному и тому же типу. В списке может не быть ни одного элемента. Пустой список обозначается как [].

Элементами списка могут быть любые значения – числа, символы, кортежи, другие списки и т. д.

Оператор : (двоеточие) используется для добавления элемента в начало списка. Его левым аргументом должен быть элемент, а правым – список.

5 Строки.

Строковые значения в языке Haskell, как и в Си, задаются в двойных кавычках. Они принадлежат типу String.

В действительности строки являются списками символов; таким образом, выражения"hello", ['h','e','l','l','o'] и 'h':'e':'l':'l':'o':[] означают одно и то же, а тип String является синонимом для [Char]. Все функции для работы со списками можно использовать при работе со строками

6 Функции.

До сих пор использовали встроенные функции языка Haskell. Теперь пришла пора научиться определять собственные функции. Для этого необходимо изучить еще несколько команд интерпретатора (напомним, что эти команды могут быть сокращены до одной буквы):

– команда :load позволяет загрузить в интерпретатор программу на языке Haskell, содержащуюся в указанном файле;

– команда :edit запускает процесс редактирования последнего загруженного файла;

– команда :reload перечитывает последний загруженный файл.

7 Функции многих переменных и порядок определения функций.

В языке Haskell можно определять функции, принимающие произвольное количество аргументов.

Функции работы со строками.

GetN(*L*,*n*) – функция вычленения N-го элемента из заданного списка.

ListSumm(L_1,L_2) – функция сложения элементов двух списков. Возвращает список, составленный из сумм элементов списков-параметров. Учесть, что переданные списки могут быть разной длины.

OddEven(*L*) – функция перестановки местами соседних чётных и нечётных элементов в заданном списке.

Практические задания

Задание 1

По приведенному примеру проверьте, каких типов будут следующие данные:

50000
5.555
‘5’
“5555”
True

Задание 2

Выполните арифметические действия:

$5^2 + 2 * 4 – 3$

$\sqrt{9-1}$

$\sqrt{9-1}$

5^{5000}

$2*\text{Cos}\ \pi$

Задание 3

Задайте кортеж вида (5, ((“hello’’, “world”),’a’)).

Извлеките из него:

число 5;

слово «hello»;

символ ‘a’.

Задание 4

Приведите пример нетривиальных выражений, принадлежащих следующему типу:

((Char,Integer), String, [Double])

[(Double,Bool,(String,Integer))]

([Integer],[Double],[(Bool,Char)])

[[[(Integer,Bool)]]]

(([Double],[Bool]),[Integer])

Требование нетривиальности в данном случае означает, что встречающиеся в выражениях списки должны содержать больше одного элемента.

Задание 5

Напишите функции, решающие следующие задачи.

1 Дано 2 числа. Проверьте заданные числа на четность/нечетность.

2 По введенному номеру недели выведите наименование этого дня. Предусмотрите сообщение в случае некорректного ввода данных.

3 Функция возвращает произведение всех положительных чисел в списке.

4 Дано 3 числа. Упорядочьте их по возрастанию.

5 Дано 3 числа. Упорядочьте их по убыванию.

6 Даны два списка целых чисел. Объедините их попарно в один список.

7 Дан список целых чисел. Добавьте элемент в конец списка.

8 Реализуйте факториал числа.

9 Реализуйте число Фибоначчи.

10 Дан список чисел. Найдите количество чисел, которые больше 10.

11 Даны два списка целых чисел. Объедините их попарно и посчитайте сумму второй пары.

12 Дан список чисел. Найдите в нем сумму положительных чисел.

13 Дан список чисел. Найдите его длину.

14 Дан список чисел. Найти их произведение.

15 Дан список чисел. Найти в нем количество нулей.

Задание 6

Согласно своему варианту решите следующие задачи: конструирование конечных списков (N – количество элементов в списке).

1 Список натуральных чисел. N = 20.

2 Список нечётных натуральных чисел. N = 20.

3 Список чётных натуральных чисел. N = 20.

4 Список квадратов натуральных чисел. N = 30.

5 Список кубов натуральных чисел. N = 30.

6 Список факториалов. N = 50.

7 Список степеней двойки. N = 25.

8 Список степеней тройки. N = 25.

9 Список степеней пятерки. N =15.

10 Список пятых степеней натуральных чисел. N = 20.

11 Список чисел Фибоначи. N= 20.

12 Список степеней семерки. N= 20.

Конструирование бесконечных списков.

1 Список факториалов.

2 Список чисел Фибоначи.

3 Список квадратов натуральных чисел.

4 Список кубов натуральных чисел.

5 Список пятых степеней натуральных чисел.

6 Список степеней пятёрки.

7 Список степеней семёрки.

8 Список степеней двойки.

9 Список четных чисел.

10 Список нечетных чисел.

11 Список степеней тройки.

12 Список натуральных чисел.

Задание 7

Напишите функции, решающие следующие задачи.

1 Три точки A, B, C лежат на одной прямой. Заданы длины AB, BC, AC. Может ли точка A лежать между точками B и C?

2 Чему равен угол, если два смежных с ним угла составляют в сумме заданное число градусов?

3 Периметр равнобедренного треугольника равен P метрам, а боковая сторона L. Найдите основание треугольника.

4 Найдите углы треугольника, если они пропорциональны заданным числам A, B, C.

5 В равнобедренном треугольнике боковая сторона A, а основание B. Найдите высоту, опущенную на основание.

6 Даны координаты центров и радиусы двух окружностей на плоскости. Может ли вторая окружность целиком содержаться внутри первой? Если нет, то сколько точек пересечения имеют окружности?

7 Можно ли из отрезков с заданными длинами A, B, C построить прямоугольный треугольник?

8 Можно ли из круглого листа железа диаметром D метров вырезать квадрат со стороной A метров?

9 Стороны треугольника A, B, C. Найдите высоту, опущенную на строну C.

10 Высота равнобедренного треугольника H метров, основание L. Найдите углы треугольника и длину боковой стороны.

11 Найдите точку, равноудаленную от осей координат и от точки с заданными координатами (x, y).

12 Даны четыре точки A, B, C, D на плоскости. Является ли четырехугольник ABCD параллелограммом?

13 Составьте уравнение прямой, проходящей через две заданные точки.

14 Даны четыре точки A, B, C, D на плоскости. Является ли четырехугольник ABCD квадратом?

Контрольные вопросы

1 В чем отличие команд интерпретатора от выражений языка Haskell?

2 Основные типы языка Haskell.

3 Функции для работы с кортежами.

4 Функции для работы со списками.

5 Условные выражения в языке Haskell.

6 Определение функций в языке Haskell.

2 Лабораторная работа № 2. Экстремальное программирование. Scrum

Цель работы: изучить особенности разработки программ и роли на проекте по методологии Scrum; получить практический опыт разработки ПО в команде по методологии Scrum.

Порядок выполнения работы

1 Ознакомится с Scrum и подготовится к проекту.
2 Выполнить практическое задание.

Основные теоретические положения

Рекомендуемая литература: официальное руководство Scrum [2].
Дополнительные материалы: статья на Хабре о методологии Scrum: https://habr.com/ru/articles/651479.

Практическое задание

Разработайте программное обеспечение в соответствии с вариантом, используя методологию SCRUM.

Студенты делятся на группы по 3 человека: 1 ScrumMaster и 2 разработчика.

Первый час пары студенты должны провести стартовый митинг и выбрать задачи для первого спринта (оставшая часть пары + доработка дома). Во время митинга требуется выполнить следующие задачи:

– разбить задачу по разработке ПО на более мелки подзадачи;

– выбрать из Product Backlog Items вашего варианта наиболее приоритетные задачи (выбирает ScrumMaster);

– сделать оценку временных затрат на выполнение задач первого спринта;

– создать документ по результатам проведения спринта (Пул задач, примерное время выполнения, кто выполняет). Ответственный – ScrumMaster.

После того, как разработка завершилась, требуется добавить в документ с результатами первого митинга реальное время, которое потрачено на выполнение задач.

По окончании спринта будете предоставлять преподавателю результаты проведения спринта – документ, который составляли, и презентацию возможностей ПО. Преподаватель будет проверять соответствие возможностей ПО и требований в Product Backlog Items. Также ScrumMaster должен отчитаться преподавателю о трудностях, которые возникли во время разработки, и о результатах спринта (совпало ли примерное время разработки с реальным и почему получился такой результат).

Контрольные вопросы

1 Чем отличается подход к разработке в методологии экстремального программирования от классических методов разработки ПО?

2 Какие принципы лежат в основе экстремального программирования и как они способствуют улучшению качества разработки?

3 Какие ключевые роли существуют в методологии Scrum и какие функции выполняют эти роли в рамках проекта?

4 В чем заключается концепция итеративности и инкрементальности в Scrum, и какие преимущества они приносят в процессе разработки ПО?

5 Каким образом происходит оценка и приоритизация задач в Scrum и почему эти процессы важны для успешной реализации проекта?

3 Лабораторная работа № 3. Анализ предметной области и требований к ПО

Цель работы: разработать интерфейс информационной системы.

Порядок выполнения работы

1 Выполнить сбор сведений по проектируемой системе и анализ предметной области.

2 Разработать функциональные и нефункциональные требования к системе.

3 Оформить документ описания системы.

4 Изучить основные теоретические положения по разработке функциональной спецификации.

5 Спроектировать интерфейс информационной системы в соответствии с функциональными требованиями, разработанными в п. 2.

6 Сделать электронный отчет.

Основные теоретические положения

Изучите материал по анализу требований к программному обеспечению:

- анализ требований к программному обеспечению на AppMaster [3];
- дополнительный материал: https://studfile.net/preview/993777/page:11/.

Практическое задание

Разработайте интерфейс информационной системы на основе предварительного анализа и созданной функциональной спецификации. Подготовьте электронный отчет о выполненной работе, включая описание системы и процесса проектирования интерфейса.

Контрольные вопросы

1 Что описывает диаграмма вариантов использования?
2 Какие элементы присутствуют на диаграмме вариантов использования?
3 Может ли быть несколько диаграмм вариантов использования?
4 Что такое функциональная спецификация?

4 Лабораторная работа № 4. Разработка программ методом TDD

Цель работы: изучить unit-тестирование и методологию TDD на практике.

Порядок выполнения работы

1 Изучить теорию и основы unit-тестирования.
2 Изучить методологию TDD.
3 Выполнить практическое задание.

Основные теоретические положения

Test Driven Development (TDD), или Test Driven Design – методология разработки программного обеспечения, при которой разработчик сначала пишет модульный тест, а затем добавляет программный код, функциональных возможностей которого достаточно для прохождения данного теста. Модульный тест можно рассматривать как небольшую спецификацию поведения системы, написание которого позволяет разработчику сосредоточиться на добавление кода, достаточного только для прохождения данного теста, тем самым обеспечивается плотность, легкость системы и добавляемые функциональные возможности полностью соответствуют требованиям документации.

Практическое задание

Разработайте программный код на языке C# с методом для решения задачи по своему варианту. Разработайте UNIT-test для проверки работы метода.

Вариант 1

Дан текст. Сделайте заглавной каждую букву каждого слова, начинающегося с заглавной буквы.

Вариант 2

Дан текст. В каждом слове текста замените заданную литеру заданной литерой (сочетанием литер). Пример:

Заменяемая литера : “б”, заменяющее сочетание литер : “ку”, слово : “абракадабра”, результат : “акуракадакура”.

Вариант 3

В каждом слове удалите литеру, стоящую между двумя заданными.

Вариант 4

Сформируйте список, информирующий о вхождении заданной литеры в текст в текст в виде ((<0 1 5 2 0>) (<3 0 1 5 2 0 1 0>)...). Цифры указывают количество вхождений литеры в каждое слово предложения.

Вариант 5

Дан текст. Замените в каждом предложении все вхождения заданного слова на заданное новое слово.

Вариант 6

Дан текст. Удалить из каждого слова в каждом предложении все повторяющиеся литеры.

Вариант 7

Дан текст. В каждом слове каждого предложения для повторяющихся литер произведите следующую замену: повторные вхождения литер удалите, к первому вхождению литеры припишите число вхождений литеры в слово. Пример: ‘((aaabb ccccddd)(eeefggg hhkl)) преобразуется в ‘((a3b2 c4d3)(e3fg3 h2kl)).

Вариант 8

Дан текст. В каждом слове вставьте после заданного 3-буквенного сочетания заданное 2-буквенное.

Вариант 9

Дан текст. Вставьте заданное новое слово после каждого вхождения другого заданного слова.

Вариант 10

Дан текст. Запишите каждое предложение текста в порядке возрастания количества гласных букв в слове.

Вариант 11

Дан текст. Перепишите каждое предложение, расположив слова в обратном алфавитном порядке.

Вариант 12

Напишите программу, которая в каждом слове исходного текста меняет местами первую и последнюю буквы.

Контрольные вопросы

1 Что такое Test Driven Development?

2 Что позволяет TDD?

3 Что такое стек?

4 Каково значение последовательности действий, которую сокращенно можно описать в виде трех слов: красный, зеленый, рефакторинг?

5 Лабораторная работа № 5. WPF. Построение интерфейса. Диспетчеры компоновки

Цель работы: ознакомиться с различными диспетчерами компоновки.

Порядок выполнения работы

1 Изучить различные диспетчеры компоновки.
2 Выполнить практические задания.

Основные теоретические положения

Окно WPF-приложения обычно представлено корневым элементом Window. Дочерним элементом корневого элемента является диспетчер компоновки, который в свою очередь содержит любое количество элементов (в том числе вложенных диспетчеров компоновки), определяющих пользовательский интерфейс. Диспетчер компоновки является объектом класса, унаследованного от абстрактного класса.

Диспетчер компоновки Canvas.

Панель Canvas поддерживает абсолютное позиционирование содержимого пользовательского интерфейса. Если пользователь изменяет размер окна, делая его меньше, чем компоновка, обслуживаемая панелью Canvas, ее внутреннее содержимое становится невидимым до тех пор, пока контейнер вновь не увеличится до размера, равного или больше начального размера области Canvas.

Диспетчер компоновки WrapPanel.

Панель WrapPanel выводит дочерние элементы последовательно слева направо (либо сверху вниз, если для атрибута Orientation установлено значение “Vertical”) и при достижении границы окна переходит на новую строку (столбец). При изменении размеров окна панель перераспределяет компоненты таким образом, чтобы они находились в окне.

Диспетчер компоновки StackPanel.

Панель StackPanel располагает содержащиеся в нем элементы управления либо в вертикальном столбце (по умолчанию), либо в горизонтальной строке (если в атрибут Orientation записано значение «Vertical»). Если в панель StackPanel добавлено больше элементов управления, чем может быть отображено по ширине/высоте StackPanel, лишние элементы обрезаются и не отображаются.

Диспетчер компоновки DockPanel.

Панель DockPanel пристыковывает дочерние элементы к различным сторонам панели: Top, Bottom, Left, Right. Атрибут LastChildFill по умолчанию имеет значение True, что означает, что последний дочерний элемент управления будет занимать всё оставшееся пространство панели.

Диспетчер компоновки Grid.

Подобно HTML-таблице, панель Grid может состоять из набора ячеек, каждая из которых имеет свое содержимое. При определении панели Grid

выполняются следующие шаги:

– определение и конфигурирование каждого столбца;

– определение и конфигурирование каждой строки;

– назначение содержимого каждой ячейке сетки с использованием синтаксиса присоединяемых свойств.

Практические задания

1 Разработайте приложение WPF со следующим графическим интерфейсом (рисунок 5.1).

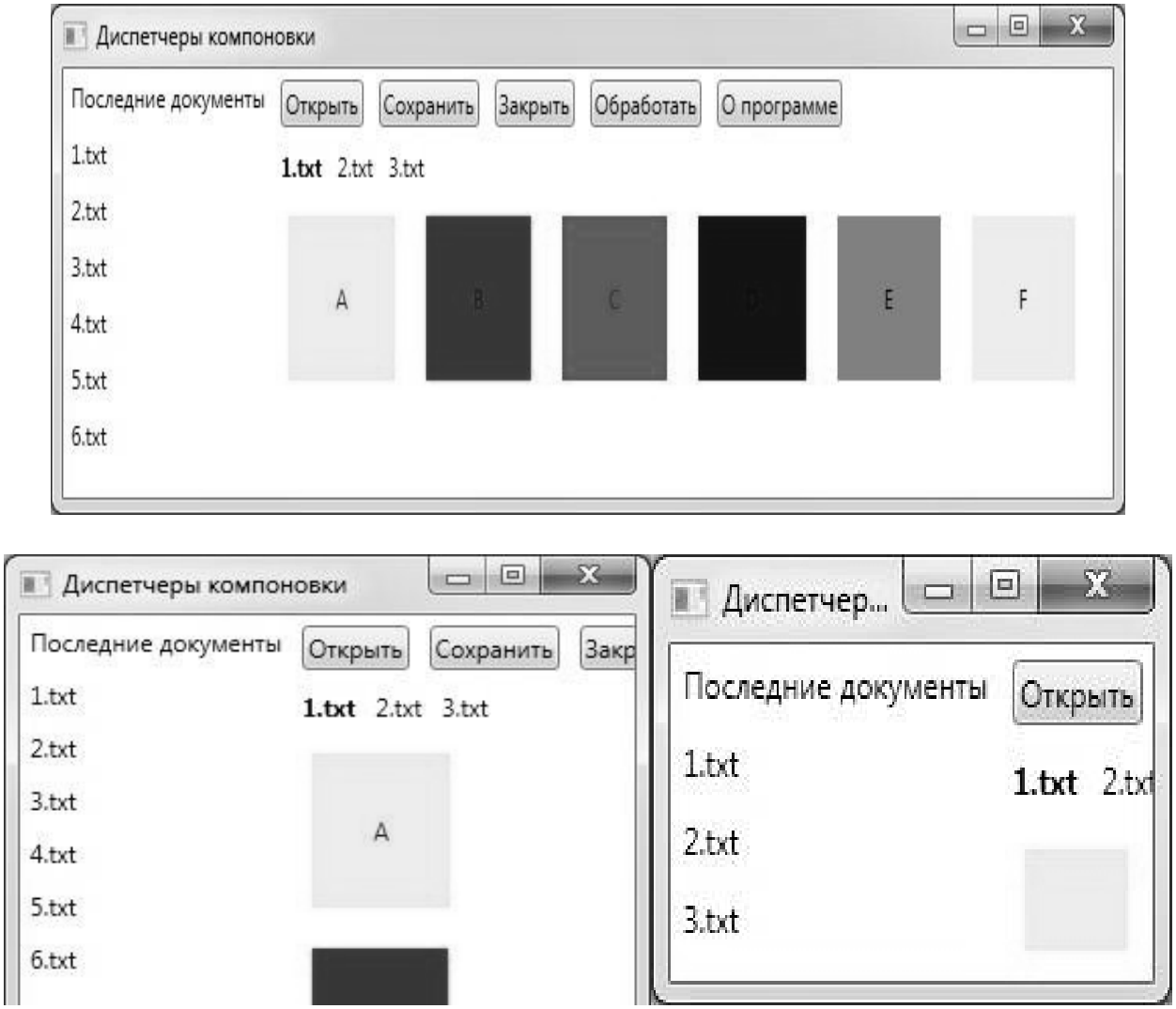

Рисунок 5.1 – Интерфейс приложения для задания 1

2 Разработайте приложение WPF со следующим графическим интерфейсом (рисунок 5.2).

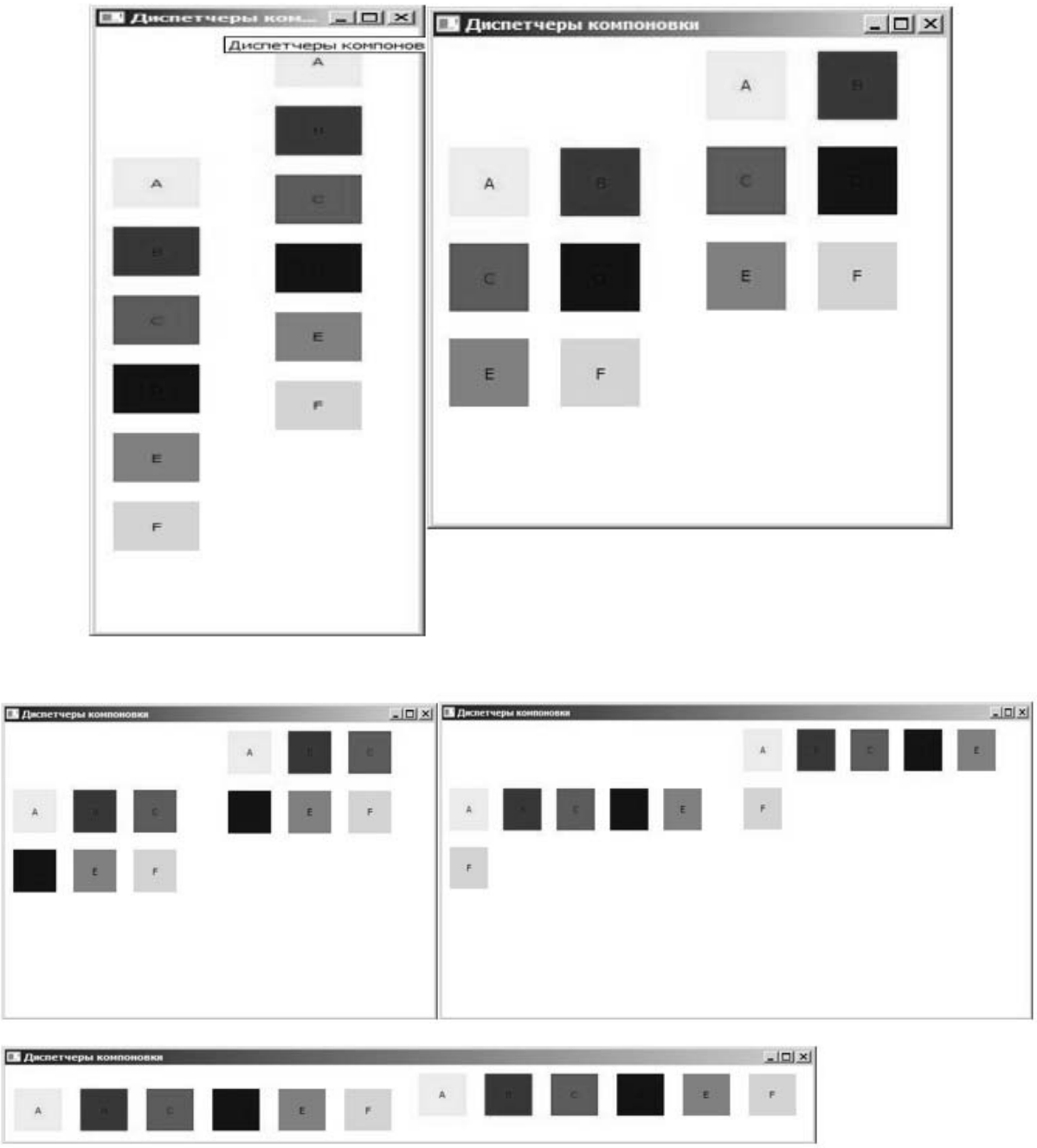

Рисунок 5.2 – Интерфейс приложения для задания 2

Контрольные вопросы

1 Какие существуют диспетчеры компоновки?
2 Для чего используются присоединяемые свойства?
3 Чем StackPanel отличается от WrapPanel и что между ними общего?
4 Какие возможности предоставляет диспетчер компоновки Canvas?

6 Лабораторная работа № 6. Основные элементы управления WPF

Цель работы: изучить основные элементы управления WPF.

Порядок выполнения работы

1 Изучить стили и шаблоны элементов управления.
2 Разобрать события и взаимодействие элементов.
3 Выполнить практические задания.

Основные теоретические положения

Элемент управления Button (кнопка) представляет собой обычную кнопку.
Отличительные особенности:
– событие Click – нажатие на кнопку. В атрибуте Click указывается название функции-обработчика этого события;
– свойство IsCancel. Возможные значения: True, False. Если записано True, то кнопка будет срабатывать при нажатии на кнопку Esc в данном окне, т. е. когда пользователь хочет закрыть окно без выполнения каких-либо действий.

Элемент управления ToggleButton (переключаемая кнопка) представляет собой кнопку, которая может находиться в двух состояниях: нажатом и отжатом.

Класс RadioButton является наследником от класса ToggleButton и наследует его свойства и события.

Свойство GroupName – название группы зависимых переключателей. В одном окне может быть несколько групп зависимых переключателей с разными названиями групп.

Элемент управления ComboBox (выпадающий список) представляет собою выпадающий список, элементы которого определены с помощью элементов ComboBoxItem.

Элемент управления ListBox (список) представляет собою список, элементы которого определены с помощью элементов ListBoxItem.

Элемент Slider представляет собою ползунок с минимальным значением Minimum, максимальным значением Maximum и текущим значением Value.

Меню в WPF представлено классом Menu, который может включать в себя набор объектов MenuItem. Каждый объект MenuItem в свою очередь может включать в себя другие объекты MenuItem и объекты Separator (разделитель).

Панель инструментов в WPF представлена классом ToolBar, который в качестве содержимого может включать в себя коллекцию любых других элементов. Панели инструментов обычно используются как альтернативный способ активизации пунктов меню.

Строка состояния в WPF представлена классом StatusBar, который в качестве содержимого может включать в себя коллекцию любых других элементов, в том числе StatusBarItem.

Элемент управления InkCanvas позволяет рисовать и редактировать линии с помощью мыши или пера. Размеры элемента управления можно задать с помощью свойств Width и Height. Свойства пера (цвет, ширину и высоту) можно настроить с помощью свойства DefaultDrawingAttributes.

Обработчики событий. Для добавления обработчика для какого-либо события объекта необходимо в открывающем теге элемента написать имя события и через знак «=» имя функции-обработчика либо выбрать команду «Новый обработчик события».

Наиболее часто используемые события:

– Click – происходит при нажатии на элемент управления;

– MouseMove – происходит, когда указатель мыши совершает движение по этому элементу;

– MouseEnter – происходит, когда указатель мыши входит в границы данного элемента.

Практические задания

Задание 1

Разработайте WPF-приложение с меню, панелью инструментов и строкой состояния. С помощью пунктов меню пользователь может изменять цвет фона окна, получить информацию о разработчике, а также закрыть окно. Кнопки панели инструментов дублируют команды меню. При наведении на пункты меню или кнопки панели инструментов в строке состояния отображается информация об этих элементах управления.

Задание 2

Разработайте WPF-приложение «Графический редактор» с выпадающим списком для выбора цвета кисти, ползунком для выбора размеров кисти и зависимыми переключателями для выбора режима работы: «рисование», «редактирование», «удаление».

Контрольные вопросы

1 Какие основные элементы управления предоставляет WPF и какие задачи они помогают решать при разработке графических пользовательских интерфейсов?

2 Какие свойства и события элементов управления в WPF позволяют взаимодействовать с ними и управлять их поведением?

3 Как создать интерфейс в WPF, используя элементы управления, и настроить их внешний вид с помощью XAML и кода?

4 Какие элементы управления используются для создания выпадающих списков и списков в WPF? Чем они отличаются друг от друга?

7 Лабораторная работа № 7. Привязка данных в WPF

Цель работы: научиться использовать механизм привязки данных в WPF.

Порядок выполнения работы

1 Изучить концепции и основы привязки данных.
2 Выполнить практическое задание.

Основные теоретические положения

Привязка данных (data binding) в графической системе WPF представляет собою отношение, которое сообщает WPF о необходимости извлечения данных из свойства исходного объекта (Source) и использования её для задания значения некоторого свойства целевого объекта (Target) (и, в некоторых случаях, наоборот) [4].

Объектом-источником может быть как элемент WPF, так и объект ADO.NET или пользовательский объект, хранящий данные. В данной лабораторной работе рассматривается связывание элементов управления WPF.

Практические задания

Задание 1

Проверьте реакцию среды разработки на неверные значения параметров ElementName и Path. Проанализируйте сообщения, которые выводятся в окне вывода (Вид → Вывод) при построении и при запуске приложения.

Задание 2

Запустите приложение со следующим XAML-кодом:

```
<TextBox x:Name="t1" />
<TextBox x:Name="t2" Text="{Binding ElementName=t1, Path=Text}" />
<Slider x:Name="slider1" />
<Slider x:Name="slider2" Value="{Binding ElementName=slider1, Path=Value}" />
```

Определите различие в поведении полей t1 и t2 и модифицируйте код, чтобы устранить его.

Задание 3

Дополните задание 2 текстовым полем ввода TextBox, в котором пользователь может ввести размер шрифта, и задайте выражения привязки таким образом, чтобы значение ползунка, текст текстового поля и размер шрифта текстового блока соответствовали друг другу.

Задание 4

Модифицируйте приложения, разработанные в предыдущей лабораторной работе: удалите как можно больше обработчиков событий и реализуйте ту же

функциональность приложения с помощью привязки данных.

Подсказки: свойство EditingMode (тип данных InkCanvasEditingMode) элемента управления InkCanvas нельзя напрямую связать с текстовым свойством выпадающего списка ComboBox или списка ListBox, т. к. в этом случае будет несовпадение типов. Для привязки данных необходимо, чтобы тип элементов списка совпадал с типом свойства EditingMode. Для этой цели необходимо добавить в ресурсы окна приложения (элемент Windows.Resources) массив (элемент x:Array) элементов типа InkCanvasEditingMode (артибут x:Type), данному ресурсу следует задать ключ (атрибут x:Key), который необходимо указать в свойстве ItemSource списка ListBox или выпадающего списка ComboBox.

Контрольные вопросы

1 Что такое механизм привязки данных в WPF и какие преимущества он предоставляет при разработке графических приложений?

2 Какие основные элементы используются при привязке данных в WPF, и как они взаимодействуют с данными?

3 Какие ключевые атрибуты и параметры применяются для установки привязки данных, такие как ElementName и Path, и как они влияют на процесс привязки?

4 Каким образом можно обработать ситуации, когда возникают ошибки при привязке данных, и какие сообщения об ошибках могут быть выведены в окне вывода среды разработки?

5 Что представляют собой выражения привязки данных и как они позволяют связывать данные между различными элементами интерфейса?

6 Каким образом можно использовать привязку данных для синхронизации значений разных элементов управления, таких как TextBox, Slider и текстовый блок?

8 Лабораторная работа № 8. Триггеры в WPF-приложениях

Цель работы: научиться использовать триггеры в WPF-приложениях.

Порядок выполнения работы

1 Ознакомиться с концепцией триггеров в WPF.

2 Выполнить практические задания.

Основные теоретические положения

С помощью триггеров можно автоматизировать процесс внесения простых изменений в стили, каковой обычно требует написания рутинной логики обработки событий [5]. Триггеры определяются в коллекции Triggers стиля:

<Style ...>

```
    <Style.Triggers>
      <!-- Триггеры  -->
    </Style.Triggers>
  </Style>
```

Простой триггер (Trigger) срабатывает в случае, когда заданное свойство текущего элемента управления принимает заданное значение. При срабатывании изменяет значения свойств элемента управления или применяет анимацию с использований свойств текущего элемента управления.

Триггер привязки (DataTrigger) отличается от простого триггера тем, что проверяет связанное свойство другого элемента управления.

Триггер события (EventTrigger) срабатывает при возникновении в текущем элементе управления заданного события. При срабатывании применяет анимацию с использований свойств текущего элемента управления.

Множественный триггер (MultiTrigger) отличается наличием нескольких условий. Триггер срабатывает только в том случае, когда все условия выполняются. При срабатывании изменяет значения свойств элемента управления или применяет анимацию с использований свойств текущего элемента управления.

Множественный триггер привязки (MultiDataTrigger) отличается от множественного триггера тем, что позволяет проверять связанные свойства других элементов управления.

Практические задания

Задание 1

Рассмотрите случай, когда для одного и того же элемента управления срабатывают сразу несколько триггеров, устанавливающих для одного и того же свойства различные значения, и определите правило, по которому определяется приоритет применения элементов Setter этих триггеров.

Задание 2

Разработайте WPF-приложение с двумя многострочными текстовыми полями, кнопками «Открыть», «Очистить», «Закрыть» и выпадающим списком для задания внешнего вида текстовых полей. Задайте для текстовых полей одинаковый градиентный фон. Кнопка «Закрыть» должна быть доступна только в том случае, если в обоих текстовых полях нет текста. Задайте для кнопок различный внешний вид при наведении курсора и при нажатии на них. Внешний вид текстовых полей (тип шрифта, размер шрифта, цвет шрифта) должен меняться в зависимости от значения, выбранного в выпадающем списке.

Дополнительное задание

Реализуйте задание 2 с различными типами триггеров.

Контрольные вопросы

1 Что такое триггер?
2 Для чего используются триггеры в WPF-приложениях?
3 Какие существуют типы триггеров?
4 Отличие множественного триггера привязки от триггера привязки.

9 Лабораторная работа № 9. Анимация в WPF-приложениях

Цель работы: научиться использовать анимацию в WPF-приложениях.

Порядок выполнения работы

1 Изучить типы анимаций в WPF.
2 Ознакомиться с интерактивной анимацией и управлением событиями.
3 Выполнить практические задания.

Основные теоретические положения

Все классы анимации объявлены в пространстве имен System.Windows.Media.Animation. Имена классов анимации начинаются с имени типа свойства, для которого предназначена данная анимация [6].

Набор классов **<Тип>Animation** предназначен для линейного изменения значения свойства от начального (или текущего) до конечного за указанный промежуток времени. Конечное значение свойства может быть задано явно либо как приращение к начальному (или текущему) значению свойства.

Набор классов **<Тип>AnimationUsingKeyFrames** предназначен для анимации с использованием ключевых кадров. Разработчик задает набор значений для изменяемого свойства, временные характеристики и способ перехода между этими значениями.

Набор классов **<Тип>AnimationUsingPath** предназначен для изменения значения свойства в соответствии с геометрическим путем.

Практические задания

Задание 1

Разработайте WPF-приложение «Убегающая кнопка»: при наведении курсора мыши на кнопку она смещается на некоторое расстояние от курсора. Событие наведения курсора мыши – MouseEnter.

Задание 2

Разработайте WPF-приложение «Пульсар», изображающее круг, меняющий цвет по следующей схеме (рисунок 9.1).

Используйте элемент «Эллипс» с радиальной градиентной заливкой RadialGradientBrush для свойства Fill(заливка).

Рисунок 9.1 – Интерфейс приложения для задания 2

Контрольные вопросы

1 Чем обеспечивается анимация в WPF-приложениях?
2 Какие существуют типы анимации?
3 Свойства типа StoryBoard.
4 Свойства типа Animation.

10 Лабораторная работа № 10. Разработка приложений ASP.NET по шаблону MVC

Цель работы: ознакомиться с шаблоном MVC (Model-View-Controller) для разработки веб-приложений с использованием ASP.NET.

Порядок выполнения работы

1 Ознакомиться с основами шаблона MVC.
2 Выполнить практическое задание.

Основные теоретические положения

Общие сведения о ASP.NET MVC.

Схема архитектуры Model-View-Controller (MVC) разделяет приложение на три основных компонента: модель, представление и контроллер. Платформа ASP.NET MVC представляет собой альтернативу схеме веб-форм ASP.NET при создании веб-приложений. Платформа ASP.NET MVC является легковесной платформой отображения с широкими возможностями тестирования и, подобно приложениям на основе веб-форм, интегрирована с существующими функциями ASP.NET, например с главными страницами и проверкой подлинности на основе членства. Платформа MVC определяется в сборке System.Web.Mvc.

Поддержка разработки через тестирование.

В дополнение к упрощению сложных структур схема MVC также облегчает тестирование приложений по сравнению с веб-приложениями ASP.NET на основе веб-форм. Например, в веб-приложении ASP.NET на основе веб-форм один класс используется для отображения вывода и для ответа на ввод пользователя. Создание автоматических тестов для приложений ASP.NET на основе веб-форм может представлять сложности, т. к. для тестирования отдельной страницы следует создать экземпляр класса страницы, всех дочерних элементов управления и других зависимых классов приложения. Большое

число экземпляров классов, необходимое для запуска страницы, усложняет создание тестов для отдельных частей приложения. Из-за этого тестирование приложений ASP.NET на основе веб-форм может быть сложнее тестирования приложения MVC. Более того, для тестирования приложения ASP.NET необходим веб-сервер. Платформа MVC разделяет компоненты и активно использует интерфейсы, что позволяет тестировать отдельные элементы вне остальной структуры.

Причины для создания приложения MVC.

Следует внимательно продумать вопрос о создании веб-приложения на основе платформы ASP.NET MVC или на основе модели веб-форм ASP.NET. Платформа MVC не заменяет собой модель веб-форм. Обе модели можно использовать для веб-приложений (при наличии существующих приложений на основе веб-форм они будут продолжать работу в нормальном режиме).

Перед использованием платформы MVC или модели веб-форм для определенного веб-сайта следует взвесить все преимущества каждого из подходов.

Платформа ASP.NET MVC имеет следующие преимущества:

– облегчает управление сложными структурами путем разделения приложения на модель, представление и контроллер;

– не использует состояние просмотра и серверные формы. Это делает платформу MVC идеальной для разработчиков, которым необходим полный контроль над поведением приложения;

– использует схему основного контроллера, при которой запросы веб-приложения обрабатываются через один контроллер. Это позволяет создавать приложения, поддерживающие расширенную инфраструктуру маршрутизации;

– обеспечивает расширенную поддержку разработки на основе тестирования;

– хорошо подходит для веб-приложений, поддерживаемых крупными коллективами разработчиков, а также веб-разработчикам, которым необходим высокий уровень контроля над поведением приложения.

Возможности платформы ASP.NET MVC.

Платформа ASP.NET MVC предоставляет следующие возможности.

1 Разделение задач приложения (логика ввода, бизнес-логика и логика пользовательского интерфейса), широкое возможности тестирования и разработки на основе тестирования. Все основные контракты платформы MVC основаны на интерфейсе и подлежат тестированию с помощью макетов объекта, которые имитируют поведение реальных объектов приложения. Приложение можно подвергать модульному тестированию без запуска контроллеров в процессе ASP.NET, что ускоряет тестирование и делает его более гибким. Для тестирования возможно использование любой платформы модульного тестирования, совместимой с .NET Framework.

2 Расширяемая и дополняемая платформа. Компоненты платформы ASP.NET MVC можно легко заменить или настроить. Разработчик может подключать собственный механизм представлений, политику маршрутизации URL-адресов, сериализацию параметров методов действий и другие компо-

ненты. Платформа ASP.NET MVC также поддерживает использование моделей контейнера внедрения зависимости (DI) и инверсии элемента управления (IOC). Модель внедрения зависимости позволяет внедрять объекты в класс, а не ожидать создания объекта самим классом. Модель инверсии элемента управления указывает на то, что если один объект требует другой объект, то первые объекты должны получить второй объект из внешнего источника (например, из файла конфигурации).

3 Расширенная поддержка маршрутизации ASP.NET. Этот мощный компонент сопоставления URL-адресов позволяет создавать приложения с понятными URL-адресами, которые можно использовать в поиске. URL-адреса не должны содержать расширения имен файлов и предназначены для поддержки шаблонов именования URL-адресов, обеспечивающих адресацию, оптимизированную для поисковых систем (SEO) и для передачи репрезентативного состояния (REST).

4 Поддержка использования разметки в существующих файлах страниц ASP.NET (ASPX), элементов управления (ASCX) и главных страниц (MASTER) как шаблонов представлений. Вместе с платформой ASP.NET MVC можно использовать существующие функции ASP.NET, например вложенные главные страницы, встроенные выражения (<%= %>), декларативные серверные элементы управления, шаблоны, привязку данных, локализацию и т. д.

5 Поддержка существующих функций ASP.NET. ASP.NET MVC позволяет использовать такие функции, как проверка подлинности с помощью форм и Windows, проверка подлинности по URL-адресу, членство и роли, кэширование вывода и данных, управление состоянием сеанса и профиля, наблюдение за работоспособностью, система конфигурации и архитектура поставщика.

Шаблон проекта MVC.

В состав платформы ASP.NET MVC входит шаблон проекта Visual Studio, который позволяет создавать веб-приложения, структура которых соответствует шаблону MVC. Этот шаблон создает новое веб-приложение MVC, конфигурация которого предусматривает все необходимые папки, шаблоны элементов и записи файла конфигурации.

При создании нового веб-приложения MVC Visual Studio предоставляет возможность создания двух проектов одновременно. Первый проект является веб-проектом, в котором реализуется приложение. Второй проект представляет собой проект модульного теста, в котором возможно создание модульных тестов для компонентов MVC первого проекта.

Для тестирования приложений ASP.NET MVC можно использовать любую платформу модульного тестирования, совместимую с платформой .NET Framework. Visual Studio Professional Edition поддерживает тестирование проектов в MSTest.

Практические задания

Разработайте MVC-приложение для своего варианта.

Вариант 1

Разработайте MVC-приложение со всеми функциями (CRUD – create, read, update, delete) для работы с базой данных о студентах для их распределения по местам практики: фамилия, год рождения, пол, группа, факультет, средний балл, место работы, город.

Вариант 2

Разработайте MVC-приложение со всеми функциями (CRUD) для работы с базой данных об автомобилях: номер, год выпуска, марка, цвет, состояние, фамилия владельца, адрес.

Вариант 3

Разработайте MVC-приложение со всеми функциями (CRUD) для работы с базой данных о квартирах, предназначенных для продажи: район, этаж, площадь, количество комнат, сведения о владельце, цена.

Вариант 4

Разработайте MVC-приложение со всеми функциями (CRUD) для работы с базой данных о книгах, купленных библиотекой: название, автор, год издания, адрес автора, адрес издательства, цена, книготорговая фирма.

Вариант 5

Разработайте MVC-приложение со всеми функциями (CRUD) для работы с базой данных о сотрудниках, имеющих компьютер: фамилия, номер комнаты, название отдела, данные о компьютерах, дата приобретения ПК.

Вариант 6

Разработайте MVC-приложение со всеми функциями (CRUD) для работы с базой данных о заказах, полученных сотрудниками фирмы: фамилия, сумма заказа, наименование товара, название фирмы клиента, фамилия заказчика.

Вариант 7

Разработайте MVC-приложение со всеми функциями (CRUD) для работы с базой данных об оценках, полученных студентами на экзаменах: фамилия, группа, предмет, номер билета, оценка, преподаватель.

Вариант 8

Разработайте MVC-приложение со всеми функциями (CRUD) для работы с базой данных об авторах веб-сайта и их статьях: имя, адрес, учетная запись, пароль, тема, заголовок, дата публикации.

Вариант 9

Спроектируйте структуру базы данных о списке рассылки и подписчиках: тема и содержание письма, дата отправки, имена и адреса подписчиков, их учетные записи и пароли.

Вариант 10

В базе данных содержится информация о туристических поездках: страна, город, изображение городской достопримечательности, количество дней, дата поездки, класс отеля, цена.

Контрольные вопросы

1 Что представляет собой шаблон MVC (Model-View-Controller) в контексте разработки веб-приложений?

2 Какие основные компоненты включает в себя архитектура MVC и как они взаимодействуют друг с другом?

3 Как создать контроллер в ASP.NET MVC, и какие действия он может выполнять?

4 Как настроить маршруты в ASP.NET MVC и за что они отвечают?

5 Какие преимущества применения шаблона MVC при разработке веб-приложений?

11 Лабораторная работа № 11. Разработка веб-приложений ASP.NET с использованием Docker и шаблона MVC

Цель работы: овладеть навыками разработки веб-приложений с использованием технологий ASP.NET, Docker и архитектурного шаблона MVC.

Порядок выполнения работы

1 Создать проект ASP.NET с использованием шаблона MVC.

2 Настроить Docker-контейнер для развертывания приложения.

3 Разработать функциональность приложения.

4 Контейнеровать приложение с использованием Docker.

5 Выполнить практическое задание.

Основные теоретические положения

Создание Dockerfile.

Dockerfile – это текстовый файл, который содержит инструкции для создания Docker-контейнера [8]. Он определяет, какую операционную систему использовать, какие зависимости установить, какие файлы приложения скопировать в контейнер и какие команды выполнить при запуске контейнера.

Создание docker-compose.yml файла.

docker-compose.yml – это файл, который определяет контейнеры и службы, связанные с вашим веб-приложением. В этом файле описываете, какие контейнеры должны быть созданы, как они взаимодействуют между собой и какие параметры конфигурации они имеют.

Сборка Docker-контейнера.

Сборка Docker-контейнера начинается с выполнения команды docker build. Docker использует Dockerfile для создания образа контейнера. В процессе сборки контейнера выполняются команды, определенные в Dockerfile. Например, можете установить зависимости, скопировать файлы приложения и выполнить другие необходимые действия.

Запуск контейнера.

После сборки контейнера можете запустить его с помощью команды docker run. Контейнер становится активным и веб-приложение становится доступным по указанным портам.

Практическое задание

1 Создайте проект ASP.NET с использованием шаблона MVC.

2 Настройте Docker-контейнер.

3 Контейнеризуйте приложение с Docker.

Контрольные вопросы

1 Как создать Docker-контейнер для приложения ASP.NET и какие компоненты Docker необходимы для этого?

2 Как настроить взаимодействие между представлениями и контроллерами в шаблоне MVC?

3 Какие основные шаги необходимо выполнить для успешной контейнизации веб-приложения с использованием Docker?

4 Какие команды Docker используются для создания, запуска и управления контейнерами?

5 Какие преимущества предоставляет использование Docker при разработке и развертывании веб-приложений?

6 Как обеспечить связь между Docker-контейнером и приложением ASP.NET?

12 Лабораторная работа № 12. Подготовка среды EXPO на React Native и первое приложение

Цель работы: получить навыки настройки среды для отладки приложений на React Native.

Порядок выполнения работы

1 Изучить теоретические положения.

2 Выполнить практическое задание.

Основные теоретические положения

Материал для изучения:
– https://docs.expo.dev/tutorial/create-your-first-app/;
– https://reactnative.dev/docs/environment-setup.

Практическое задание

1 Перейдите на сайт https://snack.expo.io/ и установите на телефон приложение Expo. Отсканируйте QR-код и синхронизируйте Expo-редактор и ваше устройство.

2 Используя стрелочную функцию (или класс), задайте компонент, который выведет на экран устройства Hello World.

3 Загрузите созданную программу на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами).

Контрольные вопросы

1 Можно ли писать в React Native все на JS?
2 Есть ли в React Native объект кнопка?
3 Что можно делать на React Native?
4 В чем разница между React и React Native?

13 Лабораторная работа № 13. Стили в React Native

Цель работы: получить навыки работы со стилями в React Native.

Порядок выполнения работы

1 Изучить стилей в React Native.
2 Выполнить практическое задание.

Основные теоретические положения

С React Native стилизуете свое приложение с помощью JavaScript. Все основные компоненты принимают свойство с именем style. Имена и значения стилей обычно соответствуют тому, как CSS работает в интернете, за исключением того, что имена пишутся с использованием верблюжьего регистра, например, backgroundColora не background-color [9].

Свойство style может быть простым старым объектом JavaScript. Это то, что обычно используем для примера кода. Также можете передать массив стилей – последний стиль в массиве имеет приоритет, поэтому можете использовать его для наследования стилей.

Один из распространенных шаблонов – заставить компонент принимать style свойство, которое, в свою очередь, используется для стилизации

подкомпонентов. Можете использовать это, чтобы сделать стили «каскадными», как в CSS.

Практическое задание

Используя стартовый шаблон https://snack.expo.io/, оформите контент таким же образом (рисунок 13.1).

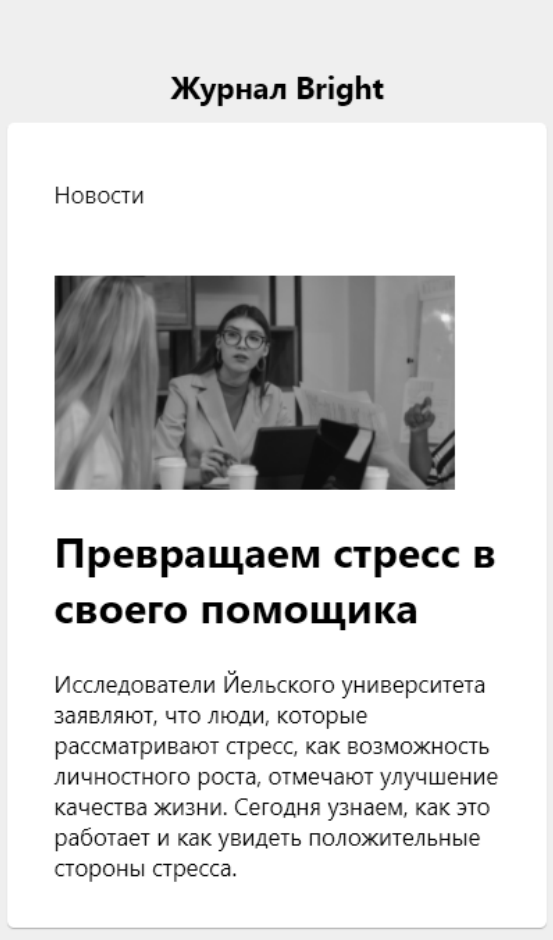

Рисунок 13.1 – Интерфейс приложения практического задания

Загрузите созданную программу на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)_2.

Для оформления можно использовать стили:

```
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';
const LotsOfStyles = () => {
   return (
     <View style={styles.container}>
       <Text style={styles.red}>just red</Text>
       <Text style={styles.bigBlue}>just bigBlue</Text>
       <Text style={[styles.bigBlue, styles.red]}>bigBlue, then red</Text>
       <Text style={[styles.red, styles.bigBlue]}>red, then bigBlue</Text>
```

```
      </View>
    );
};
const styles = StyleSheet.create({
  container: {
    marginTop: 50,
  },
  bigBlue: {
    color: 'blue',
    fontWeight: 'bold',
    fontSize: 30,
  },
  red: {
    color: 'red',
  },
});
export default LotsOfStyles;
```

Контент можно взять с сайта http://brightmagazine.ru/.

Контрольные вопросы

1 Приведите аналоги следующих тегов html в React Native: , <p>, <img>, <button>.

2 С помощью чего стилизуется приложение?

3 Какая запись правильная: backgroundColora или background-color?

4 Что такое тип стиля в React Native?

14 Лабораторная работа № 14. Гибкие элементы в React Native

Цель работы: получить навыки работы с адаптивными объектами в React Native.

Порядок выполнения работы

1 Изучить гибкие элементы в React Native.

2 Выполнить практическое задание.

Основные теоретические положения

Высота и ширина.

Высота и ширина компонента определяют его размер на экране.

Фиксированные размеры.

Общий способ установки размеров компонента – добавление фиксированного width и height к стилю. Все размеры в React Native безразмерны и

представляют собой пиксели, не зависящие от плотности.

Гибкие размеры.

Используйте flex в стиле компонента, чтобы компонент динамически расширялся и сжимался в зависимости от доступного пространства. Обычно будете использовать flex: 1, что говорит компоненту заполнить все доступное пространство, равномерно распределенное между другими компонентами с одним и тем же родителем. Чем больше flex данное значение, тем выше соотношение пространства, которое займет компонент по сравнению с его братьями и сестрами.

После того как научитесь управлять размером компонента, следующим шагом будет научиться размещать его на экране .

Процентные размеры.

Если хотите заполнить определенную часть экрана, но не хотите использовать flexмакет, можете использовать процентные значения в стиле компонента. Подобно гибким размерам, для процентных размеров требуется родительский элемент с определенным размером.

Практическое задание

Используя официальную документацию и https://reactnative.dev/docs/height-and-width, реализуйте следующее (рисунок 14.1).

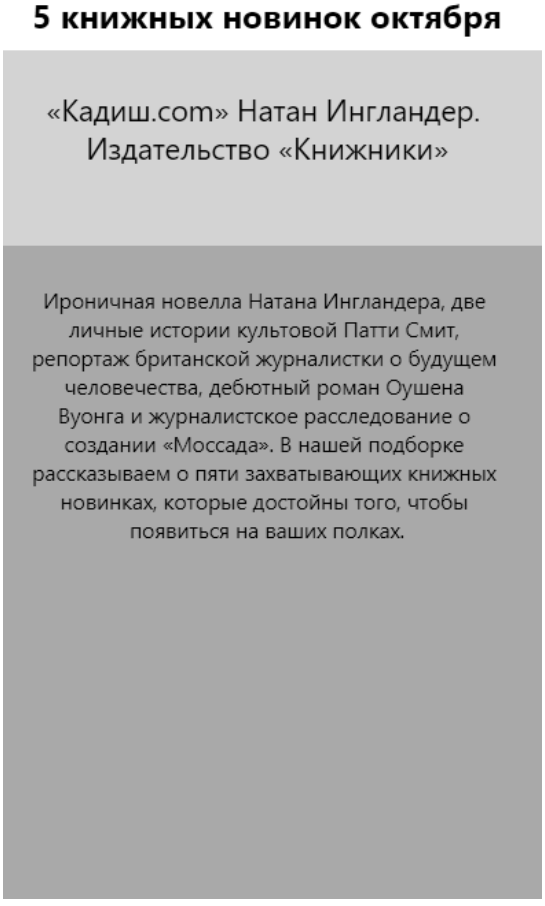

Рисунок 14.1 – Интерфейс приложения для практического задания

Загрузите созданную программу на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)_3.

Контрольные вопросы

1 Как в React Native сделать объекты фиксированной ширины и длины?

2 Что используется в стиле компонента, чтобы компонент динамически расширялся и сжимался в зависимости от доступного пространства?

3 Для чего используются процентные значения в стиле компонента?

4 Что будет, если у родителя нет фиксированного width и height?

15 Лабораторная работа № 15. Основные компоненты React Native

Цель работы: получить навыки работы с основными компонентами в React Native.

Порядок выполнения работы

1 Изучить основные компоненты в React Native.

2 Выполнить практические задание.

Основные теоретические положения

Основные компоненты и API.

React Native предоставляет ряд встроенных основных компонентов, готовых для использования в приложении. Можете найти их все на левой боковой панели (или в меню выше, если узкий экран). Если не знаете, с чего начать, взгляните на следующие категории:

– основные компоненты;

– пользовательский интерфейс;

– представления списка;

– для Android;

– для iOS.

Основные компоненты.

Большинство приложений в конечном итоге будут использовать один из этих основных компонентов:

– View (самый фундаментальный компонент для создания пользовательского интерфейса);

– Text (компонент для отображения текста);

– Image (компонент для отображения изображений);

– TextInput (компонент для ввода текста в приложение с помощью клавиатуры);

– ScrollView (предоставляет контейнер с прокруткой, который может содержать несколько компонентов и представлений);

– StyleSheet (предоставляет уровень абстракции, аналогичный таблицам стилей CSS);

Пользовательский интерфейс.

Эти общие элементы управления пользовательского интерфейса будут отображаться на любой платформе:

– Button (базовый компонент кнопки для обработки касаний, который должен хорошо отображаться на любой платформе);

– Switch (отображает логический ввод).

Представления списка.

В отличие от более общего ScrollView, следующие компоненты представления списка отображают только те элементы, которые в данный момент отображаются на экране. Это делает их эффективным выбором для отображения длинных списков данных:

– FlatList (компонент для рендеринга производительных прокручиваемых списков);

– SectionList (аналогично FlatList, но для секционированных списков).

Android компоненты и API.

Многие из следующих компонентов предоставляют оболочки для часто используемых классов Android.

– BackHandler (обнаружение нажатия аппаратных кнопок для обратной навигации);

– DrawerLayoutAndroid (отрисовывает DrawerLayoutна Android);

– PermissionsAndroid (предоставляет доступ к модели разрешений, представленной в Android M);

– ToastAndroid (создает оповещение Android Toast).

Практическое задание

1 Используя официальную документацию и https://reactnative.dev/docs/components-and-apis#user-interface, создайте экран с тремя и более основными компонентами (рисунок 15.1).

Рисунок 15.1 – Интерфейс приложения для практического задания 1

2 Сделайте экран осмысленным следующим образом (рисунок 15.2).

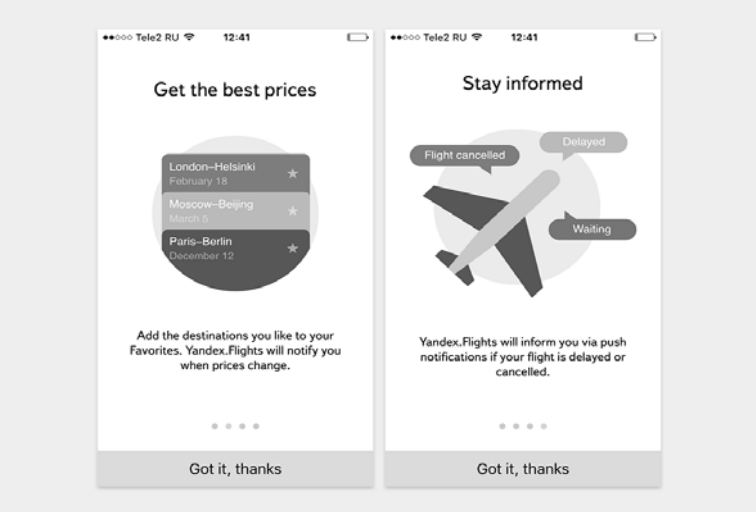

Рисунок 15.2 – Интерфейс приложения для практического задания 2

Загрузите созданную программу на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)_4.

Важно: в коде все используемые объекты должны быть импортированы: import { StyleSheet, Text, SafeAreaView, ScrollView, View, TextInput, Image, Button, Alert } from 'react-native'.

Контрольные вопросы

1 Какой компонент является базовым?
2 Какой компонент является пользовательским?
3 Какой компонент предоставляют оболочки для часто используемых классов Android?
4 Что является представлением списка?

16 Лабораторная работа № 16. Навигация в React Native

Цель работы: изучить методы навигации в приложениях, разработанных с использованием React Native.

Порядок выполнения работы

1 Изучить навигацию в React Native;
2 Выполнить практическое задание.

Основные теоретические положения

Мобильные приложения редко состоят из одного экрана. Управление представлением нескольких экранов и переходом между ними обычно осуществляется с помощью так называемого навигатора.

Установка и настройка.

Во-первых, нужно установить в свой проект:

npm install @react-navigation/native @react-navigation/native-stack.

Затем установите необходимые одноранговые зависимости. Нужно запускать разные команды в зависимости от того, является ли проект управляемым проектом Expo или простым проектом React Native.

Если есть управляемый проект Expo, установите зависимости с помощью expo:

npx expo install react-native-screens react-native-safe-area-context.

Если есть пустой проект React Native, установите зависимости с помощью npm:

npm install react-native-screens react-native-safe-area-context.

Для iOS с голым проектом React Native убедитесь, что установлены CocoaPods . Затем установите модули, чтобы завершить установку:

```
cd ios
pod install
cd ..
```

После установки можете получить предупреждения, связанные с одноранговыми зависимостями. Обычно они вызваны неправильными диапазонами версий, указанными в некоторых пакетах. Можете безопасно игнорировать большинство предупреждений, пока приложение собирается.

Теперь нужно обернуть все приложение в NavigationContainer. Обычно делаете это в своем файле ввода, например, index.js или App.js:

```
import * as React from 'react';
import {NavigationContainer} from '@react-navigation/native';
const App = () => {
  return (
    <NavigationContainer>
      {/* Rest of your app code */}
    </NavigationContainer>
  );
};
export default App;
```

Теперь готовы создать и запустить свое приложение на устройстве/симуляторе.

Практическое задание

Используя официальную документацию и https://reactnative.dev/docs/navigation, создайте три экрана вместе с навигацией по ним (рисунок 16.1).

Добавьте основные компоненты на экраны и создайте книгу контактов, галерею или любое другое простое приложение.

Загрузите созданную программу на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)_5.

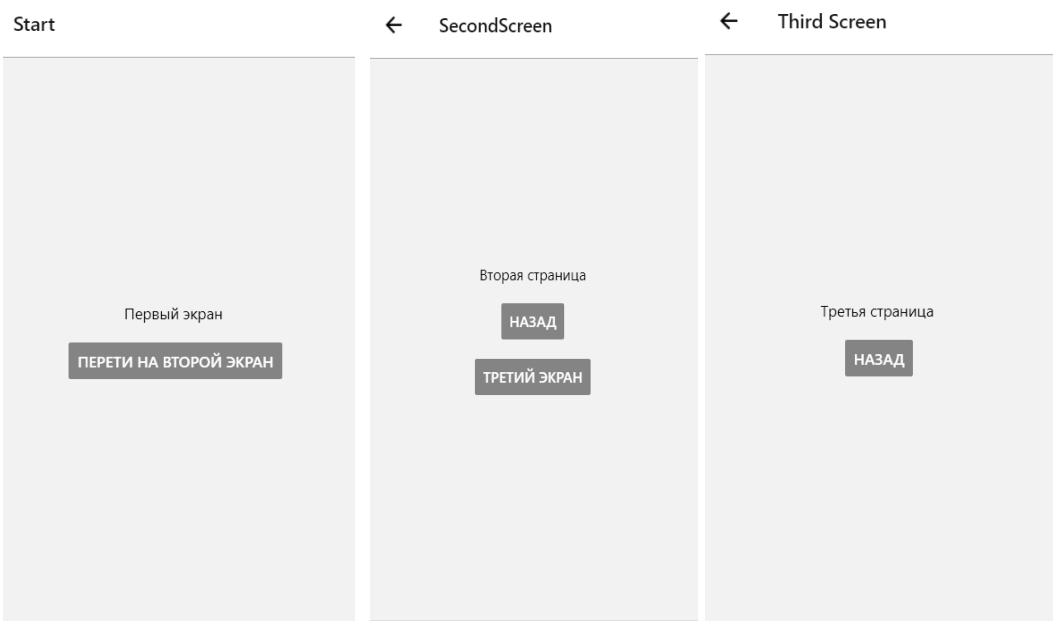

Рисунок 16.1 – Экраны приложения практического задания

Контрольные вопросы

1 Как установить зависимости для управляемого проекта Expo?
2 Что такое react Navigation?
3 Для чего используется навигация?
4 Во что оборачивают приложение?
5 Какие типы навигации можно сделать в приложении на React Native?

17 Лабораторная работа № 17. Анимация в React Native

Цель работы: изучить анимационные возможности в React Native.

Порядок выполнения работы

1 Изучить анимации в React Native.
2 Выполнить практические задание.

Основные теоретические положения

Реализация анимации в React Native.

Библиотека Animated предназначена для того, чтобы сделать анимацию плавной, мощной и простой в создании и обслуживании. Animated фокусируется на декларативных отношениях между входными и выходными данными, настраиваемых преобразованиях между ними и start/ stop – методах для управления выполнением анимации на основе времени.

Основной рабочий процесс для создания анимации заключается в создании файла Animated.Value, подключении его к одному или нескольким атрибутам

стиля анимированного компонента, а затем внесении обновлений через анимацию с помощью Animated.timing().

Можете использовать два типа значений Animated:

1) Animated.Value() – для отдельных значений;
2) Animated.ValueXY() – для векторов.

Animated.Value может привязываться к свойствам стиля или другим свойствам, а также может быть интерполирован. Один Animated.Value может управлять любым количеством свойств.

Практическое задание

1 Используя официальную документацию и https://reactnative.dev/docs/animated, ознакомьтесь с синтаксисом:

```
const FadeInView = (props) => {
  const fadeAnim = useRef(new Animated.Value(0)).current  // Initial value for opacity: 0

  useEffect(() => {
    Animated.timing(
      fadeAnim,
      {
        toValue: 1,
        duration: 10000,
      }
    ).start();
  }, [fadeAnim])

  return (
    <Animated.View                 // Special animatable View
      style={{
        ...props.style,
        opacity: fadeAnim,         // Bind opacity to animated value
      }}
    >
      {props.children}
    </Animated.View>
  );
}
```

2 Ознакомьтесь с основными принципами анимации: https://infogra.ru/ui/12-printsipov-primeneniya-animatsii-v-polzovatelskih-interfejsah.

3 Создайте анимацию для слайдера с различными картиками, добавьте другие компоненты на экран, используя созданные велкомскрины (рисунок 17.1).

4 Создайте анимированный лаучскрин.

5 Поделитесь ссылкой на проект в Expo. Загрузите созданное приложение на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)_6.

Рисунок 17.1 – Интерфейс приложения практического задания

Контрольные вопросы

1 Что такое параллакс-эффект?
2 Для чего предназначена библиотека Animated?
3 Какие существуют типы анимации Animated?
4 Если одна анимация из композиции остановлена или прервана, что происходит с остальными?

18 Лабораторная работа № 18. Приложение формы регистрации и авторизации

Цель работы: овладеть навыками создания формы регистрации в React Native.

Порядок выполнения работы

1 Изучить способ создания формы регистрации.
2 Выполнить практическое задание.

Основные теоретические положения

Ресурсы для выполнения
– векторный банк и редактор: https://icons8.com/illustrations/style--doobry;
– таблица веб-цветов: https://colorscheme.ru/html-colors.html;
– таблица иконок: https://zurb.com/playground/foundation-icon-fonts-3.

Практическое задание

1 Создайте форму регистрации с экраном альтернативной авторизации в React Native.

2 Добавьте всплывающее сообщение для кнопок: «Регистрация/Авторизация прошла успешна».

3 Поделитесь ссылкой на проект в Expo. Загрузите созданное приложение на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)_7.

Порядок выполнения:

1) создайте папки и файлы;

2) добавьте картинку и шрифты;

3) напишите код для файла App.js, импортируя туда необходимые компоненты;

4) напишите код для файлов Login.js и Register.js;

5) получите поле ввода для пароля, взяв за основу разработанный код, изменив иконки https://zurb.com/playground/foundation-icon-fonts-3 и стиль для кнопки;

6) напишите код для файла навигации;

7) добавьте шрифты и состояния в файл App.js;

8) добавьте всплывающие сообщения, воспользовавшись Alert: https://reactnative.dev/docs/button

Контрольные вопросы

1 Какие основные свойства для выравнивания объектов на экране существуют?

2 Какие компоненты были использованы для создания полей ввода (регистрации, авторизации)?

3 С помощью чего стилизовались поля ввода (регистрации, авторизации)?

4 Какие типы анимации и для чего использовались при создании полей ввода (регистрации, авторизации)?

19 Лабораторная работа № 19. Установка профессиональной среды разработки и передача данных с формы

Цель работы: создать собственную среду разработки React Native.

Порядок выполнения работы

1 Изучить теоретический материал.
2 Выполнить практические задания.

Основные теоретические положения

Способы создания среды: https://www.reactnative.express/environment.

Практические задания

Задание 1

Перед установкой подразумевается, что Node.js, приложение Snack Expo и любая среда разработки (Sublime, Visual Code) уже установлены.

1 Следуйте руководству.

https://www.reactnative.express/environment/quick_start установите среду разработки React Native.

2 Добавьте функциональные элементы на экран, обновите приложение.

3 Поделитесь ссылкой на проект в Expo. Загрузите созданное приложение на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)_8.

Задание 2

1 Изучите реализованный пример: https://snack.expo.dev/49WY2dQMS.

2 Ознакомьтесь с описанием компонента ImageBackground https://docs.expo.dev/ui-programming/image-background/.

3 Используя реализованную навигацию для велком-скрина с регистрацией, создайте экран с приветствием, данные на который будут поступать с соседнего экрана.

4 Поделитесь ссылкой на проект в Expo. Загрузите созданное приложение на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)_9.

Контрольные вопросы

1 Какие общие принципы дизайна пользовательского интерфейса можно выделить как общие для Human Interface Design и Material Design?

2 Какие уникальные особенности и подходы к UI-дизайну существуют в Human Interface Design и Material Design, которые могут отличаться друг от друга?

20 Лабораторная работа № 20. Разработка мобильного приложения «Социальная сеть»

Цель работы: овладеть навыками создания мобильного приложения.

Порядок выполнения работы

1 Ознакомиться с теоретическими положениями.
2 Выполнить практическое задание.

Основные теоретические положения

Материал для изучения:
– https://www.youtube.com/watch?v=NZdq8VmvyEw&list=PL6sXCB6Pgqf9RbLaxmiSQQXp-a0_j56xe&index=18.

Практическое задание

1 Используйте созданный проект https://github.com/Minte-grace/React-Native-Pic-Stack-UI, разверните его у себя и ознакомьтесь с кодом.
2 Замените стиль и картинки, адаптируйте интерфейс под русский язык, добавьте постов (рисунок 20.1).
3 Добавьте кнопки https://docs.expo.dev/versions/v43.0.0/react-native/share/#reference для картинок.
4 Поделитесь ссылкой на проект в Expo. Загрузите созданное приложение на GitHub в репозиторий Student, используя формат в названии Фамилия (латинскими буквами)_10.

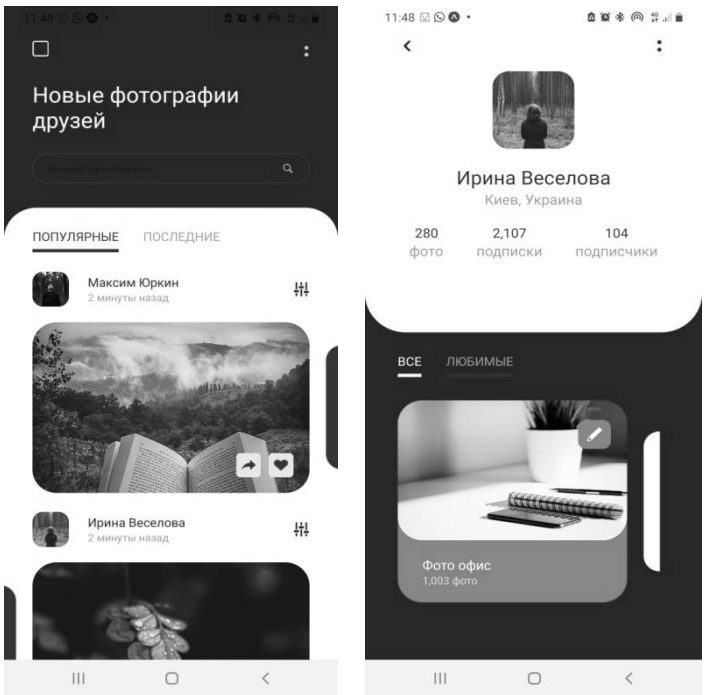

Рисунок 14 – Интерфейс приложения практического задания

Контрольные вопросы

1 Какие технологии и инструменты используются в данной лабораторной работе для разработки мобильного приложения?

2 Какие шаги необходимо предпринять для адаптации интерфейса приложения под русский язык и замены стилей и изображений?

Список литературы

1 **Липовача, М.** Изучай Haskell во имя добра!: практическое руководство / М. Липовача. – 2-е изд. – Москва: ДМК Пресс, 2023. – 492 с.

2 **Мак-Дональд, Д.** WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C 5.0 для профессионалов / Д. Мак-Дональд. – 4-е изд. – Москва: И. Д. Вильямс, 2019. – 1024 с.

3 **Dabit, N.** React Native in Action / N. Dabit. – London: Manning Publications, 2019. – 320 с.

4 **Freeman, A.** Pro ASP.NET Core MVC / A. Freeman. – New York: Apress, 2017. – 1017 с.