МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент, к.т.н.,
доцент кафедры ПМиП

_____ Е.А. Гренштейн

“___”_____2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____2020 г.

Разработка приложения для обучения корейскому языку для ОС MS Windows

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2020.308-275.ВКР

Научный руководитель,
к.ф.-м.н., доцент кафедры СП
_____А.В. Гренштейн

Автор работы,
студент группы КЭ-402
_____Е.С. Кузина

Ученый секретарь
(нормоконтролер)
_____И.Д. Володченко
“___”_____2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
09.02.2020

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-402
Кузиной Елены Сергеевной,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 24.04.2020 № 627)

Разработка приложения для обучения корейскому языку для ОС MS Windows.

2. **Срок сдачи студентом законченной работы:** 9.06.2020.
3. **Исходные данные к работе**

3.1. Что такое приложение UWP? URL: https://docs.microsoft.com/ru-ru/windows/uwp/get-started/universal-application-platform-guide.
3.2. Бьюли А. Изучаем SQL. // СПб: Символ-Плюс, 2007. С. 312
3.3. Обзор языка XAML. URL: https://docs.microsoft.com/ru-ru/windows/uwp/xaml-platform/xaml-overview.

4. **Перечень подлежащих разработке вопросов**

4.1. Выполнить анализ предметной области.
4.2. Спроектировать и разработать архитектуру приложения.
4.3. Спроектировать и реализовать приложение для ОС MS Windows.
4.4. Провести тестирование.

5. **Дата выдачи задания:** 03.02.2020.

Научный руководитель
к.ф.-м.н., доцент кафедры СП А.В. Гренштейн

Задание принял к исполнению Е.С. Кузина

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

На сегодняшний день все большей людей заинтересованы в изучении иностранных языков. Одними из самых изучаемых языков в мире являются: китайский, английский, индийский, испанский, арабский, малайский, русский, бенгальский, португальский, французский [1]. Для группы данных языков существует множество печатных и электронных самоучителей, так же существует множество приложений для самостоятельного изучения данных языков на платформах Android, Windows, iOS.

Среди молодого населения изучение корейского языка стало популярно не так давно, после увеличения популярности корейской музыкальной индустрии. Среди взрослого населения изучение корейского языка также стало популярным не так давно, вызван данный интерес развитием экономики в Южной Корее [2]. Всего за несколько поколений Южная Корея перешла из одной из самых бедных стран в мире в двенадцатую по величине [3]. Из – за недавно возникшего интереса у людей к изучению корейского языка существует не так много печатных и электронных изданий для изучения корейского языка, которые были бы переведены на русский язык. Та же ситуация и с приложениями для самостоятельного изучения языка, чаще всего в данных приложениях отсутствуют полноценные уроки, а сами они направлены на заучивание слов.

Цель и задачи

Целью данной работы является разработка приложения для ОС MS Windows для изучения корейского языка.

Для достижения указанной цели необходимо решить следующие задачи:

1) выполнить анализ предметной области;
2) провести обзор существующих решений;
3) провести проектирование архитектуры приложения;

4) реализовать приложение для платформы ОС MS Windows;

5) протестировать разработанное приложение.

Структура и объём работы

Работа состоит из введения, 5 разделов, заключения, списка библиографии, приложения. Объем работы составляет 61 страницу, объем библиографии 18 – источников.

Краткое содержание работы

Во введении описаны актуальность исследуемой темы, цели и задачи исследования, структура и объем работы.

В первой главе рассматриваются аналоги разрабатываемого приложения. Выделены их достоинства и недостатки. А также рассматриваются существующие средства для разработки.

Во второй главе описаны требования к разрабатываемой системе.

В третьей главе спроектирована диаграмма вариантов использования, приведена диаграмма классов и диаграмма деятельности, а также построена схема базы данных.

В четвертой главе описаны инструментальные средства разработки, взаимодействие компонентов приложения, приведены скриншоты интерфейса.

В пятой главе проведено функциональное тестирование разработанной системы.

В заключении перечислены основные результаты работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В данном разделе был выполнен обзор аналогов приложения, были указаны их достоинства и недостатки.

1.1. Обзор аналогов

Проведем анализ и выделим основные особенности уже используемых приложений, созданных для обучения корейскому языку. Для анализа выберем несколько наиболее популярных приложений для обучения корейскому языку.

6000 Слов – Учим Корейский Бесплатно с FunEasyLearn [4]

Приложение «6000 Слов – Учим Корейский Бесплатно с FunEasyLearn» подходит для использования людям с различными уровнями знания языка. В составе приложения доступно более 6000 слов и словосочетаний, которые носители языка чаще всего используют в повседневном общении. Все слова поделены на разные темы – от еды, работы и спорта до образования, досуга и многого другого. Данное приложение разработано для платформ Android, Windows, iOS.

Среди достоинств программы следует выделить следующее:

1) наличие аудиофайлов с правильным произношением слов и словосочетаний;
2) заучивание материала в форме игры.

К недостаткам приложения относится:

1) нет пояснений связанных с использованием грамматик;
2) нет возможности проверить полученные знания.

Учим языки с LinGo Play [5]

Приложение «Учим языки с LinGo Play» подходит для использования людям с различными уровнями знания языка. В составе приложения доступно большое количество уроков. Все уроки разделены на три вида: карточки – заучивание слов по карточкам с картинками, слова – заучивание глаголов, фразы – заучивание словосочетаний или устойчивых выражений.

Так же каждый урок разбит на темы. Однако, доступ к большинству уроков открывается только за дополнительную плату. Данное приложение разработано для платформы iOS.

Среди достоинств программы следует выделить следующее:

1) наличие аудиофайлов с правильным произношением слов и словосочетаний;
2) заучивание материала в форме игры;
3) заучивание правописания слов;
4) после каждого урока есть возможность пройти тест.

К недостаткам приложения относится:

1) нет пояснений связанных с использованием грамматик;
2) часть контента платная.

Корейский язык с Nemo [6]

Приложение «Корейский язык с Nemo» подходит для использования людям с различными уровнями знания языка. В составе приложения доступно большое количество слов и словосочетаний для заучивания. Все слова разделены на различные темы, пользователь сам решает с какой темы начать обучение. Все темы доступны без использования интернета. Данное приложение разработано для платформы Android, iOS.

Среди достоинств программы следует выделить следующее:

1) наличие аудиофайлов с правильным произношением слов и словосочетаний;
2) заучивание материала в форме игры;
3) заучивание правописания слов;
4) возможность использования, не имея доступа к интернету.

К недостаткам приложения относится:

1) нет пояснений связанных с использованием грамматик.

1.2. Обзор существующих средств для разработки платформы

На данный момент существует несколько платформ для создания

приложений для ОС MS Windows.

1. Visual Studio [7] с использованием языка разметки приложений XAML [8] и языка программирования C# [9].

2. JetBrains IntelliJ IDEA [10] с использованием языка программирования java.

3. PyCharm [11] с использованием языка программирования python.

Visual Studio

Интегрированная среда разработки Visual Studio – это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки.

JetBrains IntelliJ IDEA

Программное обеспечение JetBrains IntelliJ IDEA – это ведущая среда быстрой разработки на языке Java. IntelliJ IDEA представляет собой высокотехнологичный комплекс тесно интегрированных инструментов программирования, включающий интеллектуальный редактор исходных текстов с развитыми средствами автоматизации, мощные инструменты рефакторинга кода, встроенную поддержку технологий J2EE, механизмы интеграции со средой тестирования Ant/JUnit и системами управления версиями, уникальный инструмент оптимизации и проверки кода Code Inspection, а также инновационный визуальный конструктор графических интерфейсов.

PyCharm

PyCharm – это самая интеллектуальная Python IDE с полным набором средств для эффективной разработки на языке Python. Выпускается в двух

вариантах – бесплатная версия PyCharm Community Edition и поддерживающая больший набор возможностей PyCharm Professional Edition. PyCharm выполняет инспекцию кода на лету, автодополнение, в том числе основываясь на информации, полученной во время исполнения кода, навигацию по коду, обеспечивает множество рефакторингов.

Вывод

Анализ предметной области показывает, что на рынке представлено множество приложений, позволяющих пользователю изучать корейский язык. Несмотря на это, в большинстве приложений делается упор на заучивание слов корейского языка, но мало внимания уделяется грамматической части. Так же не во всех приложениях есть возможность проверить полученные знания. Для разработчиков также существует множество платформ, предоставляющих возможность для создания приложений для ОС MS Windows.

2. ТРЕБОВАНИЯ К ПЛАТФОРМЕ

2.1. Функциональные требования

Разрабатываемое приложение должно удовлетворять следующим функциональным требованиям.

1. Система должна предоставлять пользователю возможность переключаться между страницами в главном меню приложения.
2. Система должна предоставлять пользователю возможность переходить между имеющимися в приложении уроками.
3. Система должна предоставлять пользователю возможность проходить тест для определения уровня владением языка.
4. Система должна генерировать тестовые задания, полученные из базы данных.
5. Система должна генерировать словарь слов, полученных из базы данных.
6. Система должна генерировать словарь грамматик, полученный из базы данных.
7. Система должна предоставлять пользователю возможность добавлять слова и перевод слов в словарь слов.
8. Система должна предоставлять пользователю возможность осуществлять поиск слов по словарю слов на русском и корейском языках.
9. Система должна предоставлять пользователю возможность добавлять слова, примеры и грамматики в словарь грамматик.
10. Система должна предоставлять пользователю возможность осуществлять поиск по словарю грамматик на русском и корейском языках.

2.2. Нефункциональные требования

Разрабатываемое приложение должно удовлетворять следующим нефункциональным требованиям.

1. Приложение должно быть доступно на платформе ОС MS Windows.
2. Интерфейс приложения должен быть реализован на языке разметки

приложений XAML.

3. База данных приложения должна быть реализована с помощью программной библиотеки SQLite [12].

4. Приложение должно быть реализовано на языке программирования C#.

Вывод

В ходе анализа требований к платформе были выявлены функциональные и нефункциональные требования.

3. ПРОЕКТИРОВАНИЕ

3.1. Варианты использования системы

Для проектирования приложения был использован язык графического описания для объектного моделирования UML. Была построена модель взаимодействия внешнего актера с программной системой в виде диаграммы вариантов использования (use – case diagram) [13].

В ходе анализа разрабатываемого приложения были выявлены основные варианты использования, представленные на рисунке 1.

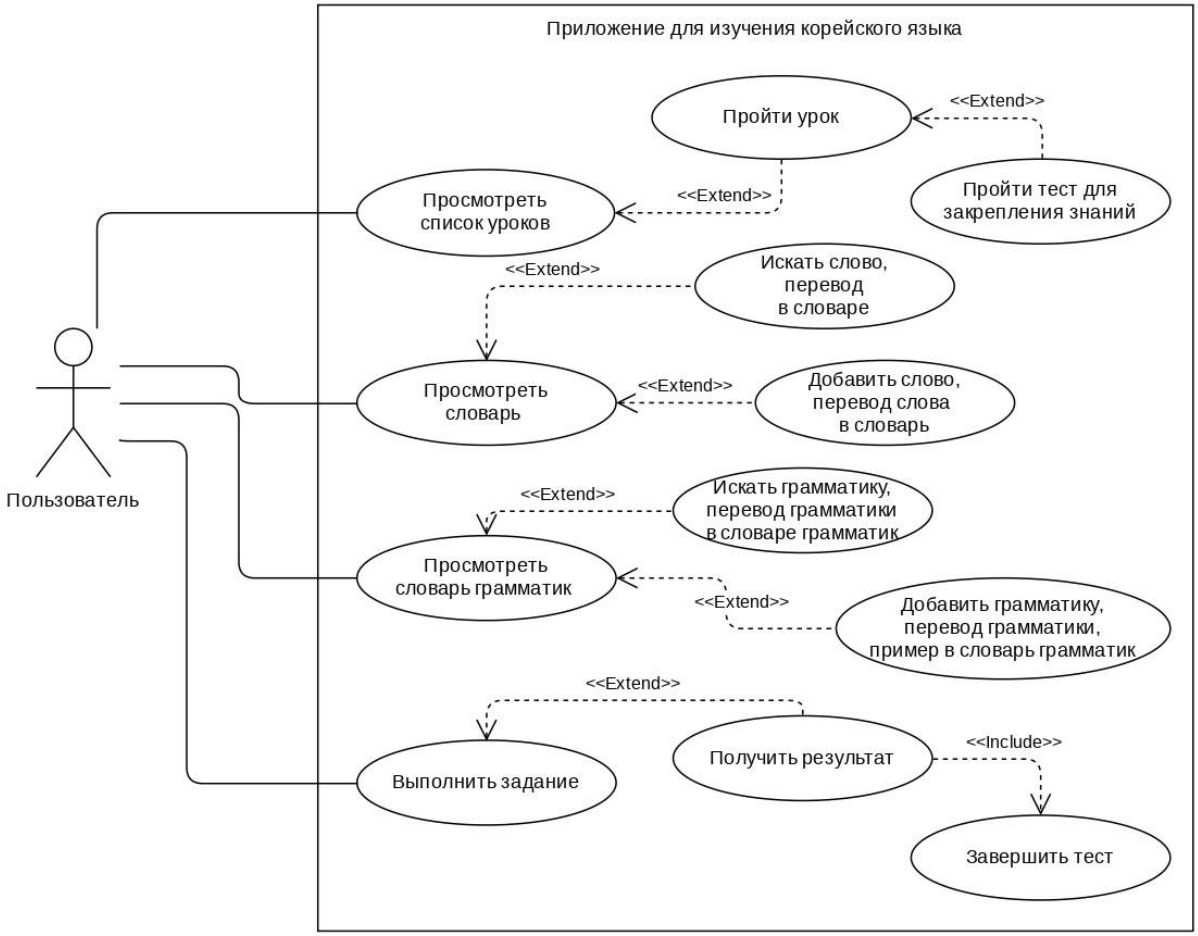

Рис. 1. Диаграмма вариантов использования

Для данной диаграммы основным актером, взаимодействующим с системой, является Пользователь. Для данного актера были определены следующие варианты использования:

1) *просмотреть список уроков* – в главном меню программы выбрать страницу «Уроки», после перехода на указанной странице будут находиться

все уроки доступные пользователю;

2) *пройти урок* – выбрать интересующий урок в списке уроков на странице «Уроки», ознакомиться с информацией урока, при желании пройти тест для закрепления знаний;

3) *пройти тест для закрепления знаний* – в конце каждого урока доступен тест для закрепления знаний о пройденном материале. Данный тест является не обязательным к прохождению;

4) *просмотреть словарь* – в главном меню программы выбрать страницу «Словарь», после перехода на указанной странице будут выведены: словарь слов корейского языка, поле для поиска слов в словаре, поля для добавления новых слов и перевода слов в словарь;

5) *искать слово или перевод слова в словаре* – в поле, находящемся на странице «Словарь», ввести искомое слово или его перевод, нажать кнопку для начала поиска;

6) *добавить новое слово и его перевод в словарь* – в поля, находящиеся на странице «Словарь», ввести значение нового слова и его перевод, нажать кнопку для добавления нового слова в словарь;

7) *посмотреть словарь грамматик* – в главном меню программы выбрать страницу «Словарь грамматик», после перехода на указанной странице будут выведены: словарь грамматик корейского языка, поле для поиска слов в словаре, поля для добавления новых слов, примеров и перевода слов в словарь;

8) *искать грамматику или перевод грамматики в словаре* – в поле, находящемся на странице «Словарь грамматик», ввести искомую грамматику или ее перевод, нажать кнопку для начала поиска;

9) *добавить новую грамматику, пример и ее перевод в словарь* – в поля, находящиеся на странице «Словарь грамматик», ввести значение новой грамматики, пример и ее перевод, нажать кнопку для добавления новой грамматики в словарь;

10) *выполнить задание* – в главном меню программы выбрать

страницу «Тест», после перехода на указанной странице будет выведено тестовое задание, выбрать вариант ответа;

11) *получить результаты теста* – результаты теста будут выведены системой после того, как пользователь пройдет тест;

12) *завершить тест* – тест считается пройденным, после того как пользователь ответит на указанные вопросы и в конце теста нажмет на кнопку для завершения теста, данная кнопка становиться доступной на последней странице теста.

3.2. Диаграмма классов

Для проектирования приложения был использован язык графического описания для объектного моделирования UML. Была построена диаграмма классов (class diagram) [14].

В ходе анализа разрабатываемого приложения были выявлены основные классы приложения, представленные на рисунке 2.

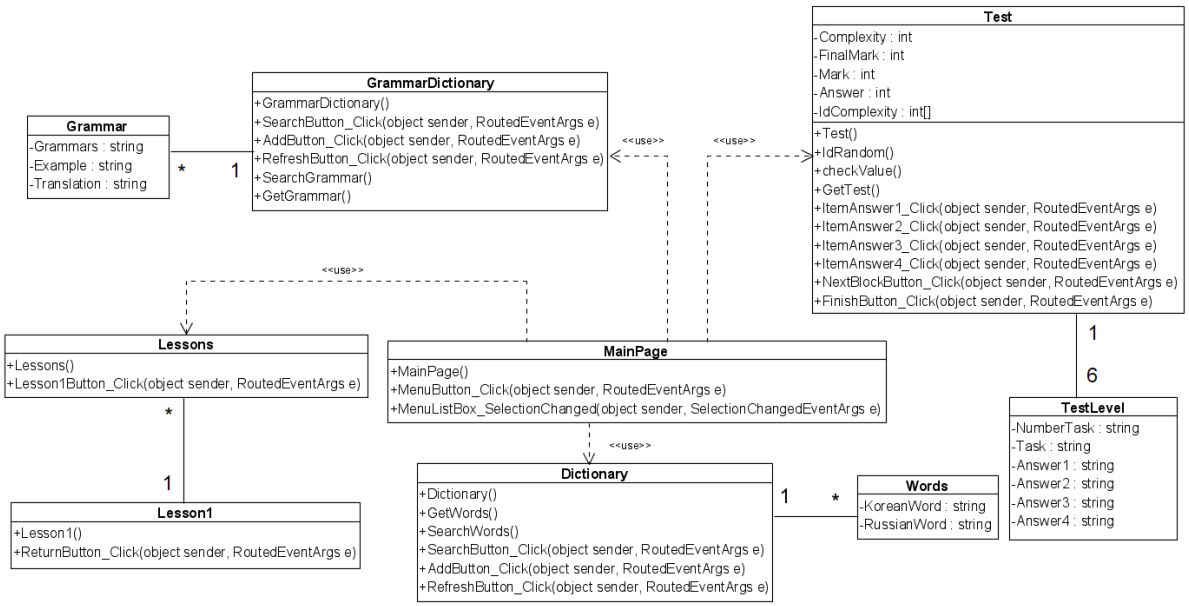

Рис. 2. Диаграмма классов приложения

Класс *MainPage* реализует основную логику приложения, в данном классе расположено главное меню приложения.

Класс *Lessons* содержит информацию обо всех уроках доступных

пользователю.

Класс *Lesson1* реализует интерфейс урока.

Класс *Dictionary* реализует основную логику словаря.

Класс *Words* предназначен для хранения и загрузки информации о словах в словаре.

Класс *GrammarDictionary* реализует основную логику словаря грамматик.

Класс *Grammar* предназначен для хранения и загрузки информации о грамматиках в словаре грамматик.

Класс *Test* реализует основную логику теста для определения уровня знания языка.

Класс *TestLevel* предназначен для хранения и загрузки информации о заданиях для каждого уровня знания языка.

3.3. Диаграммы деятельности

На основе требований к системе были разработаны диаграммы деятельности (activity diagram) [15]. Эти диаграммы показывают, как происходит процесс взаимодействия пользователя с системой.

Диаграмма деятельности перехода на страницу словаря и взаимодействие с ним представлена на рисунке 3.

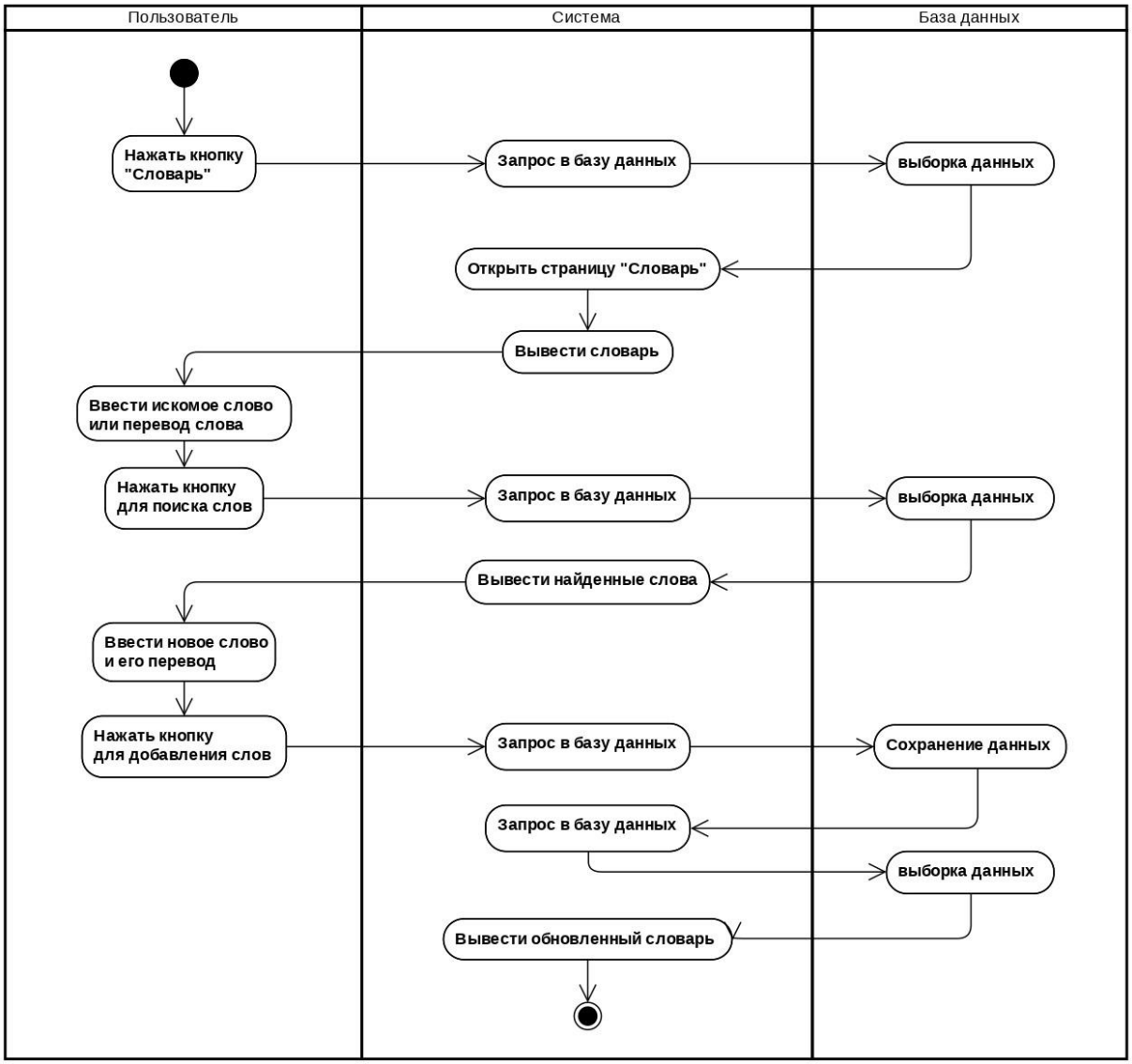

Рис.3. Диаграмма деятельности перехода на страницу словаря и взаимодействие с ним

Диаграмма деятельности перехода на страницу словаря грамматик и взаимодействие с ним представлена на рисунке 4.

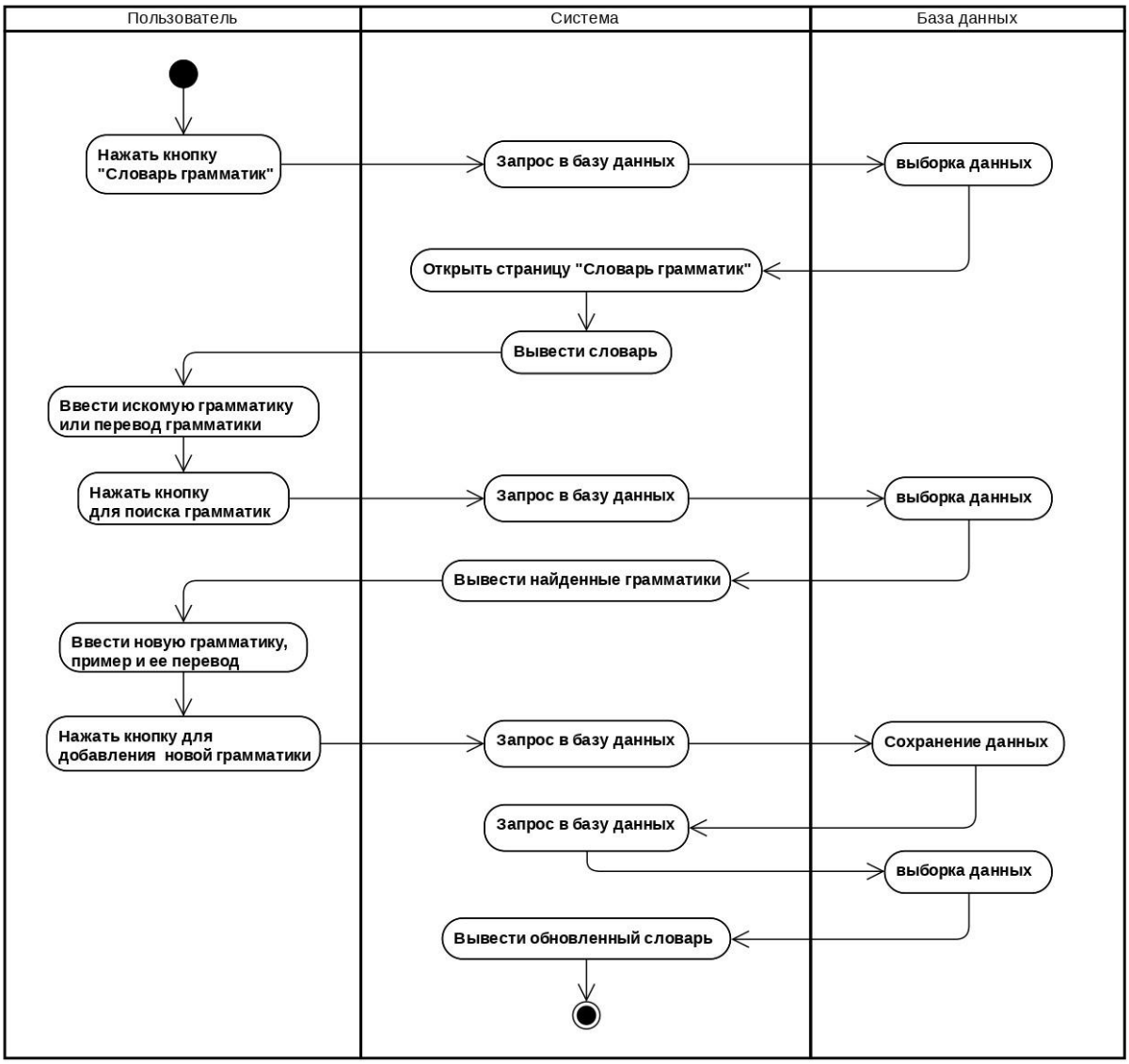

Рис.4. Диаграмма деятельности перехода на страницу словаря грамматик и взаимодействие с ним

Диаграмма деятельности перехода на страницу теста и взаимодействие с ним представлена на рисунке 5.

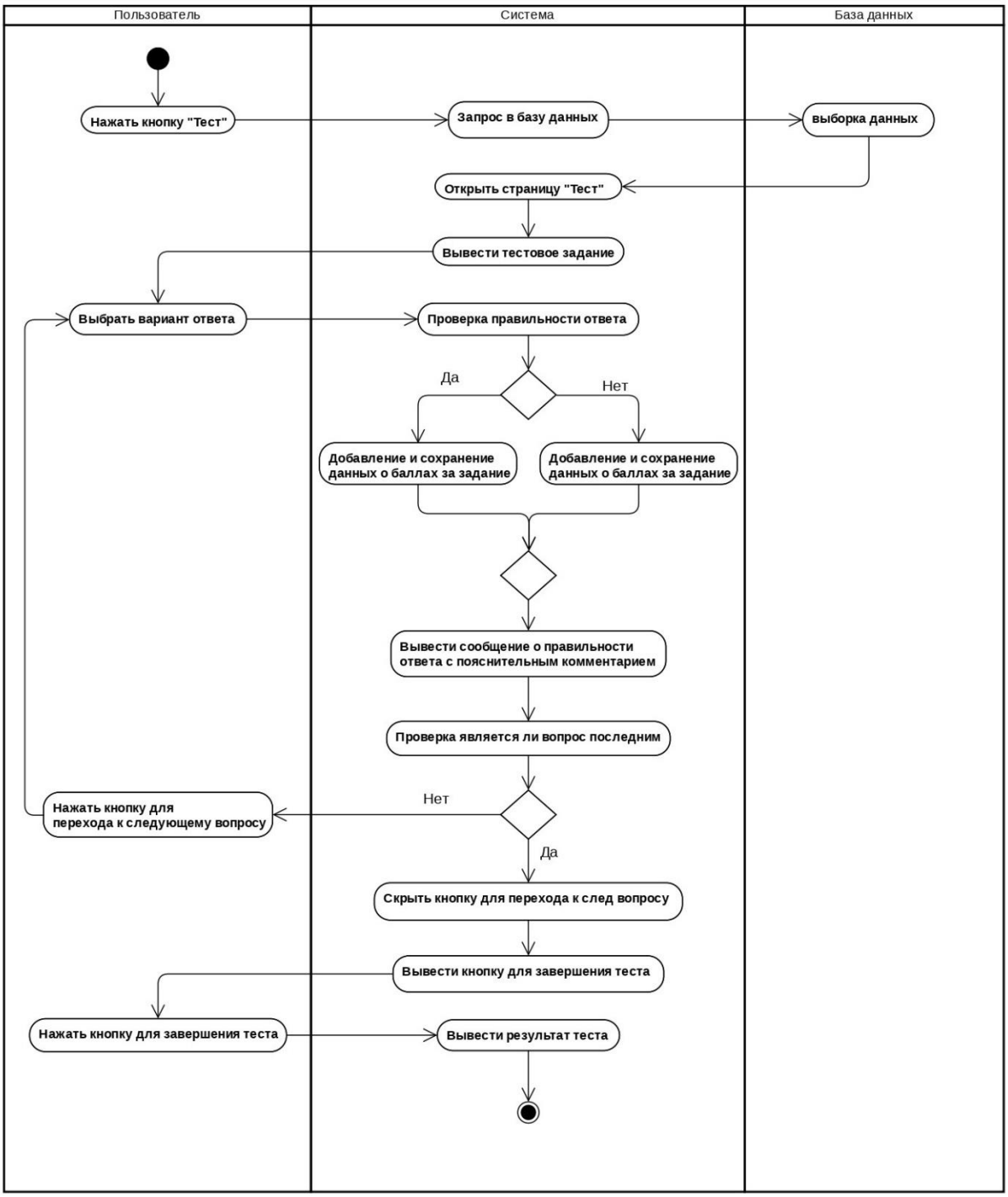

Рис.5. Диаграмма деятельности перехода на страницу теста и взаимодействие с ним

3.4. База данных

После сравнения различных СУБД в качестве используемой базы данных было решено использовать SQLite СУБД. В данной базе данных есть свои достоинства и недостатки, но после проведенного анализа, выявилось,

что достоинств в SQLite СУБД больше, в связи с этим было принято решение об использовании данной СУБД для реализации приложения. Все выявленные достоинства и недостатки представлены ниже.

Достоинства SQLite СУБД:

1) файловая структура – вся база данных состоит из одного файла, поэтому её очень легко переносить на разные электронно – вычислительные машины;

2) отсутствие необходимости настройки сервера СУБД;

3) полностью свободная лицензия;

4) кроссплатформенность;

5) высокая скорость простых операций выборки данных;

6) поддержка транзакций, триггеров, представлений (views), вложенных запросов;

7) очень экономичная, в плане ресурсов, архитектура;

8) безопасность. SQLite СУБД хранится в одном файле, права доступа к которому можно контролировать стандартными средствами ОС.

Недостатки SQLite СУБД:

1) отсутствие системы пользователей – более крупные СУБД включают в свой состав системы управления правами доступа пользователей. Обычно применения этой функции не так критично, так как эта СУБД используется в небольших приложениях;

2) отсутствие возможности увеличения производительности.

3.4.1. Схема базы данных

На рисунке 6 представлена схема базы данных, составленная на основе требований к разрабатываемой системе.

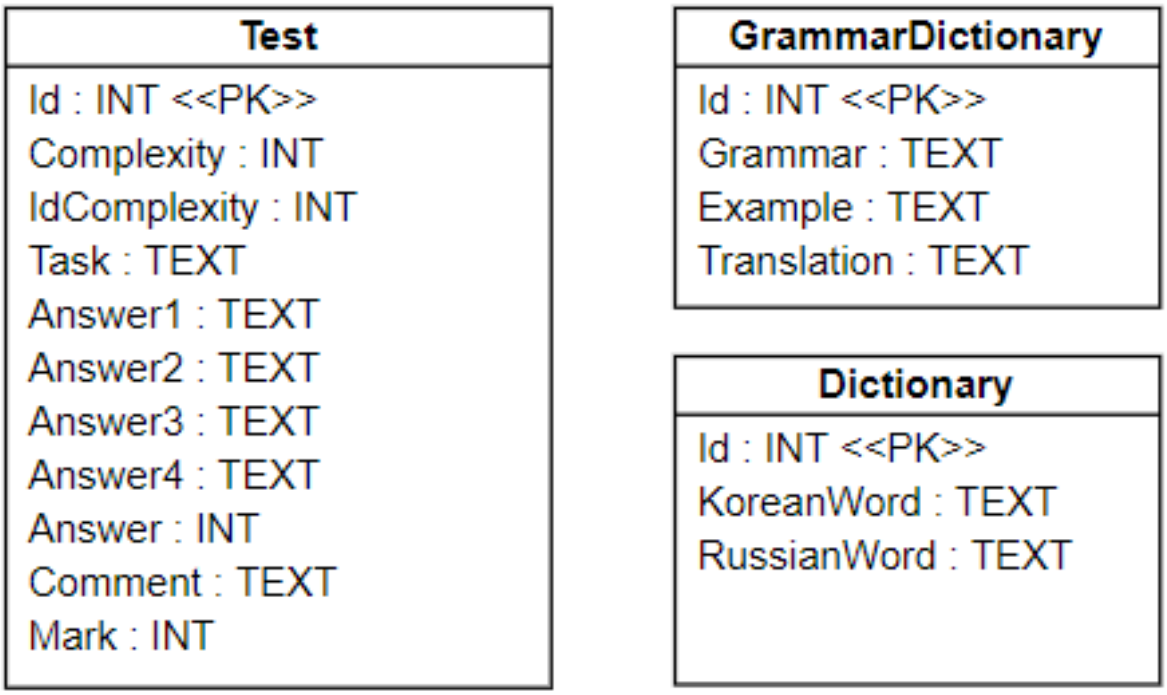

Рис. 6. Схема базы данных

Таблица *Dictionary* содержит информацию о словаре слов. Каждый кортеж состоит из следующих атрибутов.

1. *Id* содержит уникальный идентификатор слова.
2. *KoreanWord* содержит слово на корейском языке.
3. *RussianWord* содержит перевод слова на русский язык.

Таблица *GrammarDictionary* содержит информацию о словаре грамматик. Каждый кортеж состоит из следующих атрибутов.

1. *Id* содержит уникальный идентификатор грамматики.
2. *Grammar* содержит информацию о грамматике.
3. *Example* содержит пример, в котором употребляется грамматика.
4. *Translation* содержит перевод грамматики.

Таблицы *Test* содержат информацию о заданиях для определенного уровня владения языка и информацию о полученном балле за задание. Каждый кортеж состоит из следующих атрибутов.

1. *Id* содержит уникальный идентификатор задания.
2. *Complexity* содержит информацию о сложности тестового задания.

3. *IdComplexity* содержит идентификатор сложности задания.

4. *Task* содержит тестовое задание.

5. *Answer1* содержит один вариант ответа к тестовому заданию.

6. *Answer2* содержит один вариант ответа к тестовому заданию.

7. *Answer3* содержит один вариант ответа к тестовому заданию.

8. *Answer4* содержит один вариант ответа к тестовому заданию.

9. *Answer* содержит правильный ответ на задание.

10. *Comment* содержит комментарий к правильному ответу, который включает в себя разбор задания и его перевод на русский язык.

11. *Mark* содержит балл, получаемый за правильное выполнение задания.

Вывод

В главе проектирование с учетом назначения системы и обзора готовых решений для ее реализации, проведенных в первой главе, а так же с учетом функциональных и нефункциональных требований, приведенных во второй главе, представлены диаграмма вариантов использования, диаграмма классов, а так же диаграммы деятельности. Была разработана и описана схема базы данных.

4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

4.1. Программные средства реализации

Для разработки программной части системы был выбран язык программирования c#, для разработки интерфейса программы был выбран язык разметки XAML. Разработка велась в среде Microsoft Visual Studio 2019.

Для разработки были использованы следующие библиотеки: Microsoft.Data.Sqlite [16], Microsoft.NETCore.UniversalWindowsPlatform [17].

4.2. Реализация главной страницы

Для реализации главной страницы приложения был создан класс *MainPage*. В данном классе были реализованы два метода: *MenuButton_Click* и *MenuListBox_SelectionChanged*. Метод *MenuButton_Click* предназначен для открытия главного меню по нажатию кнопки. А метод *MenuListBox_SelectionChanged* предназначен для переходов между страницами приложения.

Реализация класса *MainPage* приведена на рисунке 7.

```
private void MenuButton_Click(object sender, RoutedEventArgs e)
        {
            MenuSplitView.IsPaneOpen = !MenuSplitView.IsPaneOpen;
        }

private void MenuListBox_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            if (ItemLessons.IsSelected)
            {
                MenuFrame.Navigate(typeof(Lessons));
                HeaderText.Text = "Уроки";
            }
            if (ItemDictionary.IsSelected)
            {
                MenuFrame.Navigate(typeof(Dictionary));
                HeaderText.Text = "Словарь";
            }
            if (ItemGrammarDictionary.IsSelected)
            {
                MenuFrame.Navigate(typeof(GrammarDictionary));
                HeaderText.Text = "Словарь грамматик";
            }
            if (ItemTest.IsSelected)
            {
                MenuFrame.Navigate(typeof(Test));
                HeaderText.Text = "Тест на уровень знания языка";
            }
        }
```

Рис. 7. Реализация класса *MainPage*

4.3. Реализация словаря слов и словаря грамматик

4.3.1. Реализация словаря слов

Для реализации словаря слов был создан класс *Dictionary*. В данном классе были реализованы три метода и два класса коллекций.

1. *SearchButton_Click.*

2. *AddButton_Click.*

3. *RefreshButton_Click.*

4. *ObservableCollection<Words>GetWords*.

5. *ObservableCollection<Words>SearchWords*.

Метод *SearchButton_Click* предназначен для поиска слов. Данный метод для поиска вызывает класс *ObservableCollection<Words>SearchWords*. В классе *ObservableCollection<Words>SearchWords* искомые слова, которые пользователь ввел в поле, передаются в базу данных в таблицу Dictionary в

виде запроса, все найденные слова записываются по порядку в коллекцию слов, а затем выводятся.

Реализация класса *ObservableCollection<Words>SearchWords* приведена на рисунке 8.

```
public ObservableCollection<Words> SearchWords()
        {
            var entries = new ObservableCollection<Words>();
            using (SqliteConnection db = new
SqliteConnection("Filename=Korean.db"))
            {
                db.Open();
                SqliteCommand selectCommand = new SqliteCommand("SELECT
* FROM Dictionary WHERE KoreanWord LIKE '%" + SearchTextBox.Text + "%'
OR RussianWord LIKE '%" + SearchTextBox.Text + "%'", db);
                SqliteDataReader query;
                try
                {
                    query = selectCommand.ExecuteReader();
                }
                catch (SqliteException error)
                {
                    return entries;
                }
                while (query.Read())
                {
                    var entrie = new Words();
                    entrie.KoreanWord = query.GetString(1);
                    entrie.RussianWord = query.GetString(2);
                    entries.Add(entrie);
                }
                db.Close();
            }
            return entries;
        }
```

Рис. 8. Реализация класса *ObservableCollection<Words>SearchWords*

Реализация метода *SearchButton_Click* приведена на рисунке 9.

```
private void SearchButton_Click(object sender, RoutedEventArgs e)
        {
            WordsList.ItemsSource = SearchWords();
        }
```

Рис. 9. Реализация метода *SearchButton_Click*

Метод *RefreshButton_Click* предназначен для обновления таблицы слов словаря. Данный метод вызывает класс *ObservableCollection<Words> GetWords*. В классе *ObservableCollection<Words>GetWords* передается в

базу данных запрос на вывод всех слов словаря, данные слова затем записываются в коллекцию и выводятся.

Реализация класса *ObservableCollection <Words> GetWords* представлена на рисунке 10.

```
public ObservableCollection<Words> GetWords()
        {
            var entries = new ObservableCollection<Words>();
            using (SqliteConnection db = new
SqliteConnection("Filename=Korean.db"))
            {
                db.Open();
                SqliteCommand selectCommand = new SqliteCommand("SELECT
* from Dictionary", db);
                SqliteDataReader query;
                try
                {
                    query = selectCommand.ExecuteReader();
                }
                catch (SqliteException error)
                {
                    return entries;
                }
                while (query.Read())
                {
                    var entrie = new Words();
                    entrie.KoreanWord = query.GetString(1);
                    entrie.RussianWord = query.GetString(2);
                    entries.Add(entrie);
                }
                db.Close();
            }
            return entries;
        }
```

Рис. 10. Реализация класса *ObservableCollection<Words>GetWords*

Метод *AddButton_Click* предназначен для добавления новых слов в словарь. В данном методе слова, введенные пользователем, передаются в базу данных в виде запроса, веденные слова сохраняются в базе данных, после этого для обновления таблицы метод вызывает класс *ObservableCollection<Words>GetWords* для вывода обновленного словаря.

Реализация метода *AddButton_Click* представлена на рисунке 11.

```
private void AddButton_Click(object sender, RoutedEventArgs e)
        {
            using (SqliteConnection db = new
SqliteConnection("Filename=Korean.db"))
            {
                db.Open();
                SqliteCommand insertCommand = new SqliteCommand();
                insertCommand.Connection = db;
                insertCommand.CommandText = "INSERT INTO Dictionary
(KoreanWord, RussianWord) VALUES (@KoreanWord, @RussianWord)";
                insertCommand.Parameters.AddWithValue("@KoreanWord",
AddKoreanTextBox.Text);
                insertCommand.Parameters.AddWithValue("@RussianWord",
AddRussianTextBox.Text);
                try
                {
                    insertCommand.ExecuteReader();
                }
                catch (SqliteException error)
                {
                    return;
                }
                db.Close();
            }
            WordsList.ItemsSource = GetWords();
        }
```

Рис. 11. Реализация метода *AddButton_Click*

4.3.2. Реализация словаря грамматик

Для реализации словаря грамматик был создан класс *GrammarDictionary*. В данном классе были реализованы три метода и два класса коллекций: *SearchButton_Click*, *AddButton_Click*, *RefreshButton_Click*, *ObservableCollection<Grammar>GetGrammar*, *ObservableCollection<Grammar>SearchGrammar*. Метод *SearchButton_Click* предназначен для поиска грамматик. Данный метод для поиска вызывает класс *ObservableCollection<Grammar>SearchGrammar*. В классе *ObservableCollection<Grammar>SearchGrammar* искомые грамматики, которые пользователь ввел в поле, передаются в базу данных в таблицу GrammarDictionary в виде запроса, все найденные грамматики записываются по порядку в коллекцию грамматик, а затем выводятся.

Реализация класса *ObservableCollection<Grammar>SearchGrammar* представлена на рисунке 12.

```
public ObservableCollection<Grammar> SearchGrammar()
        {
            var entries = new ObservableCollection<Grammar>();
            using (SqliteConnection db = new
SqliteConnection("Filename=Korean.db"))
            {
                db.Open();
                SqliteCommand selectCommand = new SqliteCommand("SELECT
* FROM GrammarDictionary WHERE Grammar LIKE '%" + SearchTextBox.Text +
"%' OR Translation LIKE '%" + SearchTextBox.Text + "%'", db);
                SqliteDataReader query;
                try
                {
                    query = selectCommand.ExecuteReader();
                }
                catch (SqliteException error)
                {
                    return entries;
                }
                while (query.Read())
                {
                    var entrie = new Grammar();
                    entrie.Grammars = query.GetString(1);
                    entrie.Example = query.GetString(2);
                    entrie.Translation = query.GetString(3);
                    entries.Add(entrie);
                }
                db.Close();
            }
            return entries;
        }
```

Рис. 12. Реализация класса

ObservableCollection<Grammar>SearchGrammar

Реализация метода *SearchButton_Click* представлена на рисунке 13.

```
private void SearchButton_Click(object sender, RoutedEventArgs e)
        {
            GrammarList.ItemsSource = SearchGrammar();
        }
```

Рис. 13. Реализация метода *SearchButton_Click*

Метод *RefreshButton_Click* предназначен для обновления таблицы грамматик. Данный метод вызывает класс *ObservableCollection<Grammar> GetGrammar*. В классе *ObservableCollection<Grammar>GetGrammar* передается в базу данных запрос на вывод всех грамматик словаря, данные грамматики затем записываются в коллекцию и выводятся.

Реализация класса *ObservableCollection<Grammar>GetGrammar* представлена на рисунке 14.

```
public ObservableCollection<Grammar> GetGrammar()
        {
            var entries = new ObservableCollection<Grammar>();
            using (SqliteConnection db = new
SqliteConnection("Filename=Korean.db"))
            {
                db.Open();
                SqliteCommand selectCommand = new SqliteCommand("SELECT
* from GrammarDictionary", db);
                SqliteDataReader query;
                try
                {
                    query = selectCommand.ExecuteReader();
                }
                catch (SqliteException error)
                {
                    return entries;
                }
                while (query.Read())
                {
                    var entrie = new Grammar();
                    entrie.Grammars = query.GetString(1);
                    entrie.Example = query.GetString(2);
                    entrie.Translation = query.GetString(3);
                    entries.Add(entrie);
                }
                db.Close();
            }
            return entries;
        }
```

Рис. 14. Реализация класса *ObservableCollection<Grammar>GetGrammar*

Метод *AddButton_Click* предназначен для добавления новых грамматик в словарь. В данном методе грамматика, введенная пользователем, передается в базу данных в виде запроса, веденная грамматика сохраняется в базе данных, после этого для обновления таблицы метод вызывает класс *ObservableCollection<Words>GetWords* для вывода обновленного словаря грамматик.

Реализация метода *AddButton_Click* представлена на рисунке 15.

```
private void AddButton_Click(object sender, RoutedEventArgs e)
        {
            using (SqliteConnection db = new
SqliteConnection("Filename=Korean.db"))
            {
                db.Open();
                SqliteCommand insertCommand = new SqliteCommand();
                insertCommand.Connection = db;
                insertCommand.CommandText = "INSERT INTO
GrammarDictionary (Grammar, Example, Translation) VALUES (@Grammar,
@Example, @Translation)";
                insertCommand.Parameters.AddWithValue("@Grammar",
AddGrammarTextBox.Text);
                insertCommand.Parameters.AddWithValue("@Example",
AddExampleTextBox.Text);
                insertCommand.Parameters.AddWithValue("@Translation",
AddTranslationTextBox.Text);
                try
                {
                    insertCommand.ExecuteReader();
                }
                catch (SqliteException error)
                {
                    return;
                }
                db.Close();
            }
            GrammarList.ItemsSource = GetGrammar();
        }
```

Рис. 15. Реализация метода *AddButton_Click*

4.4. Реализация теста

Для реализации теста был создан класс *Test*. Метод *IdRandom* с помощью функции Random генерирует набор случайных идентификаторов, которые записываются в массив. В методе *checkValue* сгенерированные идентификаторы проверяются на наличие повторений, если повторения есть, то в методе *IdRandom* идентификаторы перезаписываются до тех пор, пока повторяющихся идентификаторов не будет. Класс коллекций *ObservableCollection<TestLevel>GetTest* отвечает за передачу запроса в базу данных для вывода тестовых заданий с определенными идентификаторами, которые были сгенерированы в методе *IdRandom* и разбиение полученных данных на элементы коллекции, которые в дальнейшем будут выведены.

Реализация класса *ObservableCollection<TestLevel>GetTest* представлена на рисунке 16.

```
public ObservableCollection<TestLevel> GetTest()
        {
            var entries = new ObservableCollection<TestLevel>();
            using (SqliteConnection db = new
SqliteConnection("Filename=Korean.db"))
            {
                db.Open();
                SqliteCommand selectCommand = new SqliteCommand("SELECT
* from Test where Test.Complexity = " + Complexity + " and
Test.IdComplexity = " + IdComplexity[i] + "", db);
                SqliteDataReader query;
                try
                {
                    query = selectCommand.ExecuteReader();
                }
                catch (SqliteException error)
                {
                    return entries;
                }
                while (query.Read())
                {
                    var entrie = new TestLevel();
                    entrie.NumberTask = "Задание: " +
dbCounter.ToString() + " из 48";
                    entrie.Task = query.GetString(3);
                    entrie.Answer1 = query.GetString(4);
                    entrie.Answer2 = query.GetString(5);
                    entrie.Answer3 = query.GetString(6);
                    entrie.Answer4 = query.GetString(7);
                    Answer = query.GetInt32(8);
                    Comment = query.GetString(9);
                    Mark = query.GetInt32(10);
                    entries.Add(entrie);
                }
                db.Close();
            }
            i += 1;
            dbCounter += 1;
            return entries;
        }
```

Рис. 16. Реализация класса *ObservableCollection<TestLevel>GetTest*

Реализация методов *IdRandom* и *checkValue* представлена на рисунке 17.

```
public void  IdRandom()
        {
            Random random = new Random();
            i = 0;
            for (int k = 0; k < 2; k++)
            {
                do
                {
                    value = random.Next(1, 5);
                }
                while (checkValue() == false);
                IdComplexity[k] = value;
            }
        }
bool checkValue()
        {
            for (int k = 0; k < 2; k++)
            {
                if (IdComplexity[k] == value)
                    return false;
            }
            return true;
        }
```

Рис. 17. Реализация методов *IdRandom* и *checkValue*

Методы *ItemAnswer1_Click*, *ItemAnswer2_Click*, *ItemAnswer3_Click*, *ItemAnswer4_Click* отвечают за проверку правильности ответа пользователя. Листинги данных методов представлены в приложении А. Метод *NextBlockButton_Click* отвечает за переход к следующему заданию, когда пользователь нажимает соответствующую кнопку. Листинг данного метода представлен в приложении А. Метод *FinishButton_Click* отвечает за завершение теста, когда пользователь нажимает соответствующую кнопку и за вывод итогового результата с подсчитанными баллами.

Реализация метода *FinishButton_Click* представлена на рисунке 18.

```
private void FinishButton_Click(object sender, RoutedEventArgs e)
        {
            TestList.Visibility = Visibility.Collapsed;
            FinishTextBlock.Visibility = Visibility.Visible;
            if ((FinalMark >= 0) && (FinalMark <= 160))
            {
                FinishTextBlock.Text = "Вы набрали : " + FinalMark + "
баллов из 600 баллов \n Ваш примерный уровень знания корейского языка: 1
급";
            }
            if ((FinalMark >= 161) && (FinalMark <= 250))
            {
                FinishTextBlock.Text = "Вы набрали : " + FinalMark + "
баллов из 600 баллов \n Ваш примерный уровень знания корейского языка: 2
급";
            }
            if ((FinalMark >= 251) && (FinalMark <= 330))
            {
                FinishTextBlock.Text = "Вы набрали : " + FinalMark + "
баллов из 600 баллов \n Ваш примерный уровень знания корейского языка: 3
급";
            }
            if ((FinalMark >= 331) && (FinalMark <= 480))
            {
                FinishTextBlock.Text = "Вы набрали : " + FinalMark + "
баллов из 600 баллов \n Ваш примерный уровень знания корейского языка: 4
급";
            }
            if ((FinalMark >= 481) && (FinalMark <= 530))
            {
                FinishTextBlock.Text = "Вы набрали : " + FinalMark + "
баллов из 600 баллов \n Ваш примерный уровень знания корейского языка: 5
급";
            }
            if ((FinalMark >= 531) && (FinalMark <= 600))
            {
                FinishTextBlock.Text = "Вы набрали : " + FinalMark + "
баллов из 600 баллов \n Ваш примерный уровень знания корейского языка: 6
급";
            }
            FinishButton.Visibility = Visibility.Collapsed;
        }
```

Рис. 18. Реализация метода *FinishButton_Click*

4.5. Интерфейс

Для реализации пользовательского интерфейса приложения был использован язык разметки XAML.

Листинги реализации интерфейса готового приложения представлены в приложении Б.

На рисунке 19 представлен интерфейс стартового окна приложения, который представляет собой окно Windows – приложения, разделённого на три блока.

1. Верхняя панель, содержащая кнопку управления меню и блок с заголовком страницы.

2. Левая боковая панель, содержащая меню приложения.

3. Центральная панель, в которую по умолчанию выводится список уроков страницы «Уроки».

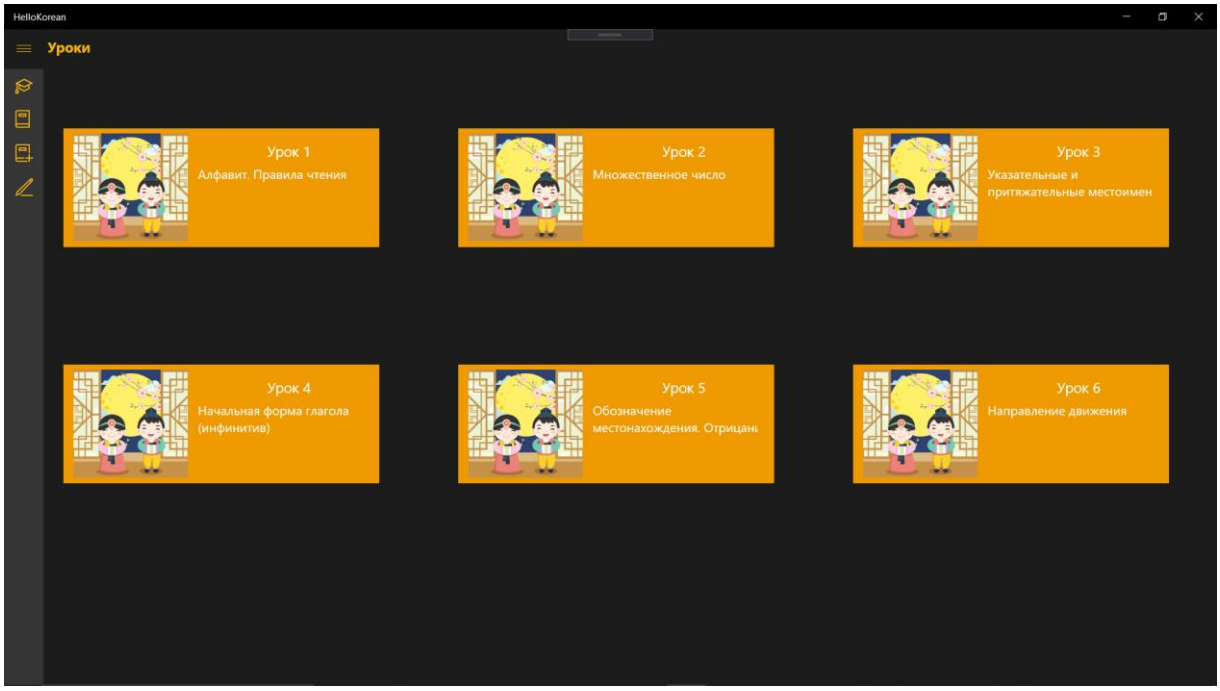

Рис. 19. Интерфейс стартового окна приложения

На рисунке 20 представлен интерфейс словаря, который представляет собой окно Windows – приложения, разделённого на четыре блока.

1. Верхняя панель, содержащая кнопку управления меню и блок с заголовком страницы.

2. Левая боковая панель, содержащая меню приложения.

3. Правая боковая панель, содержащая поле для ввода искомых слов, кнопку для поиска, поля для добавления новых слов в словарь, кнопку для

сохранения слов и кнопку обновления словаря, а также блок с сокращениями, использованными в словаре.

4. Центральная панель содержит словарь.

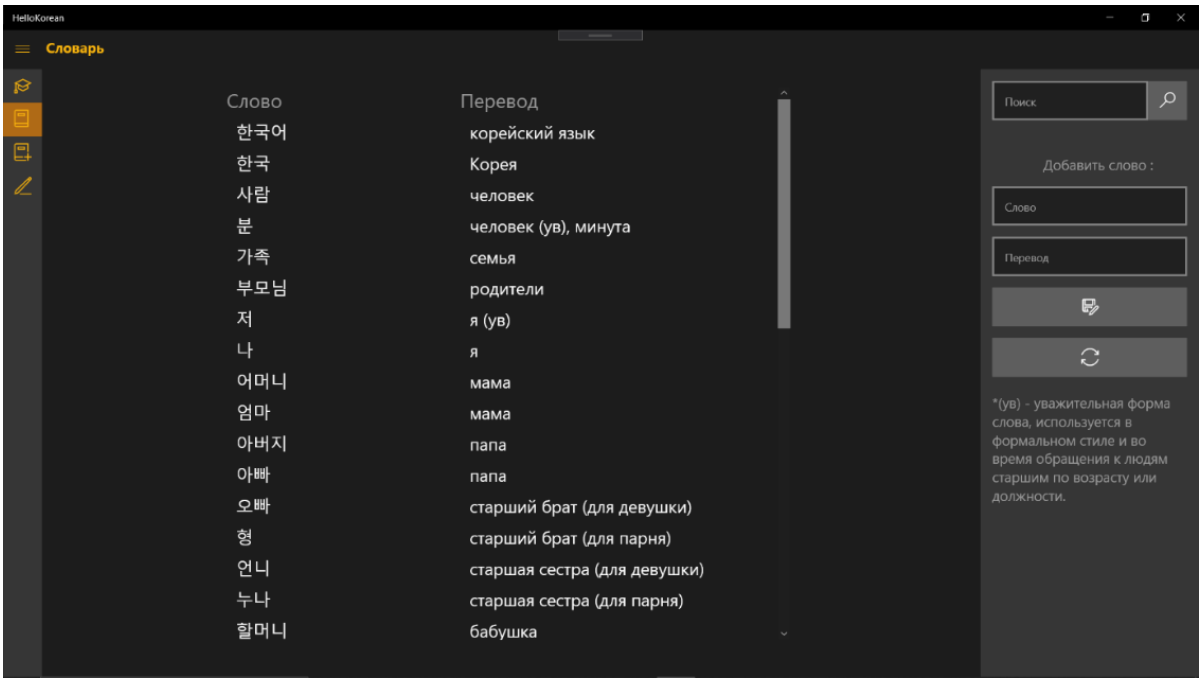

Рис. 20. Интерфейс словаря приложения

На рисунке 21 представлен интерфейс словаря грамматик, который представляет собой окно Windows – приложения, разделённого на четыре блока.

1. Верхняя панель, содержащая кнопку управления меню и блок с заголовком страницы.

2. Левая боковая панель, содержащая меню приложения

3. Правая боковая панель, содержащая поле для ввода искомых грамматик, кнопку для поиска, поля для добавления новых грамматик в словарь, кнопку для сохранения грамматик и кнопку обновления словаря грамматик, а также блок с сокращениями, использованными в словаре грамматик.

4. Центральная панель содержит словарь грамматик.

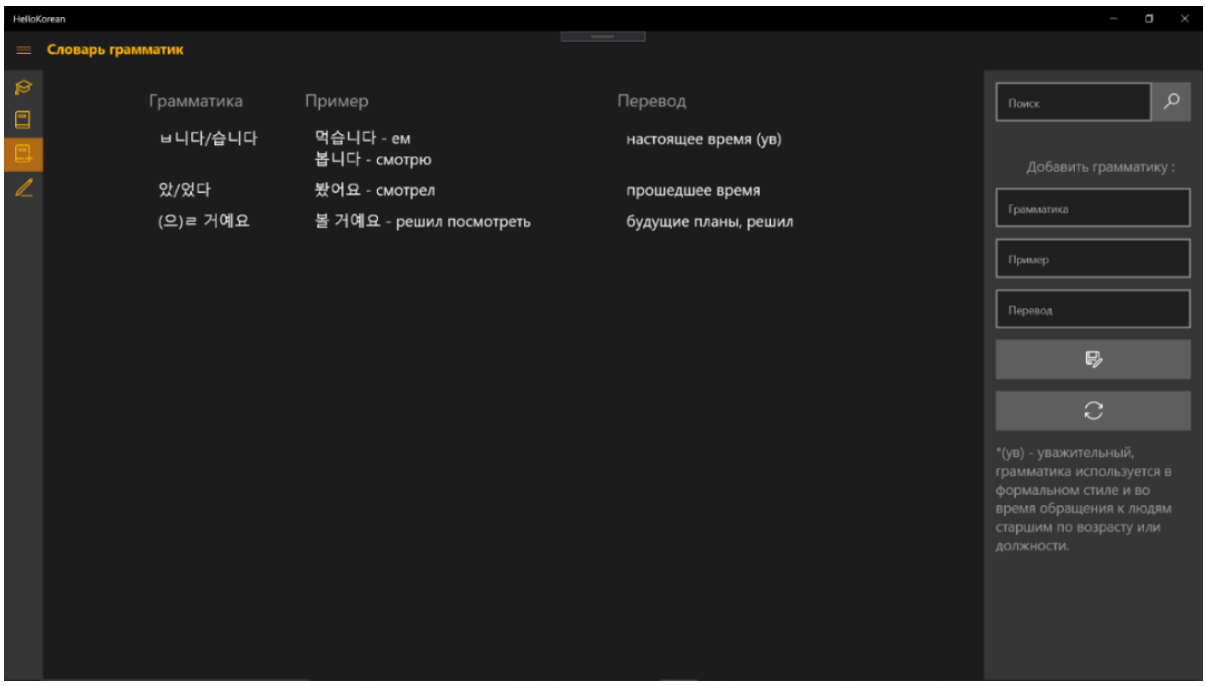

Рис. 21. Интерфейс словаря грамматик приложения

На рисунке 22 представлен интерфейс теста, который представляет собой окно Windows – приложения, разделённого на четыре блока.

1. Верхняя панель, содержащая кнопку управления меню и блок с заголовком страницы.

2. Левая боковая панель, содержащая меню приложения.

3. Правая боковая панель, содержащая блок, в который выводится информация о правильности выбранного ответа и поясняющая информация.

4. Центральная панель содержит блок с тестовым заданием, кнопку перехода к следующему заданию и поясняющий комментарий.

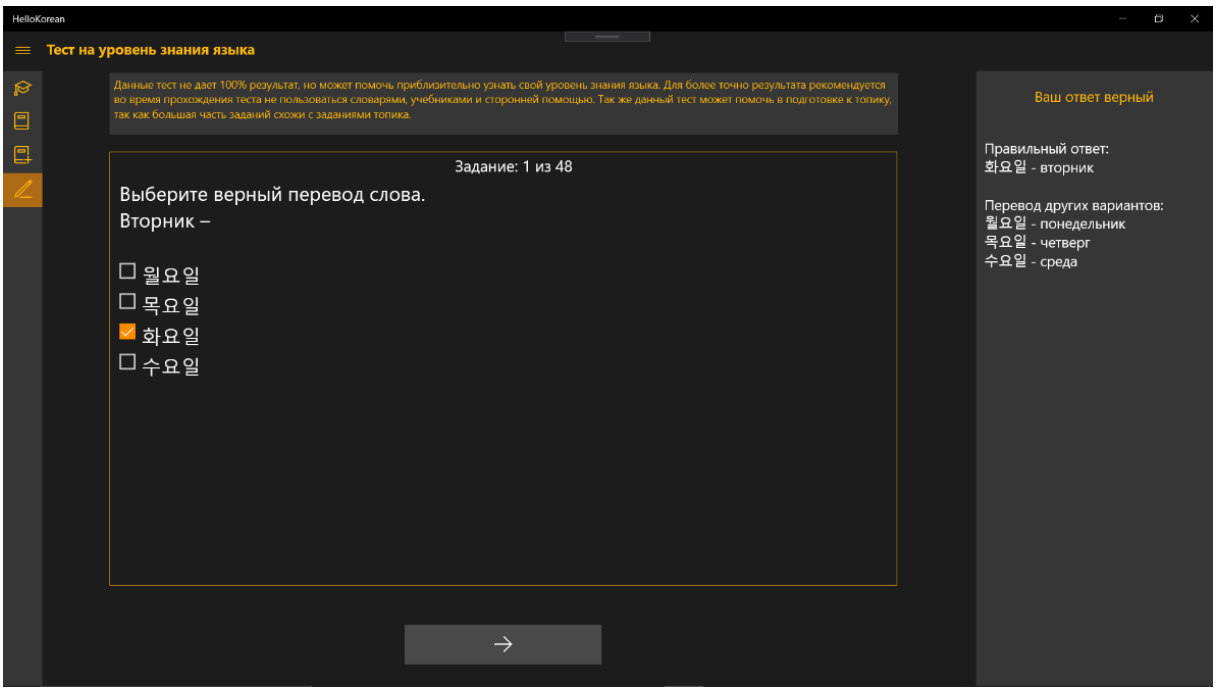

Рис. 22. Интерфейс теста приложения

На рисунке 23 представлен интерфейс урока 1, который представляет собой окно Windows – приложения, разделённого на четыре блока.

1. Верхняя панель, содержащая кнопку управления меню, блок с заголовком страницы и кнопку возврата назад.

2. Левая боковая панель, содержащая меню приложения.

3. Правая боковая панель, содержащая словарь урока.

4. Центральная панель содержит блок с текстом урока и кнопки для прослушивания верного произношения слов или букв корейского языка.

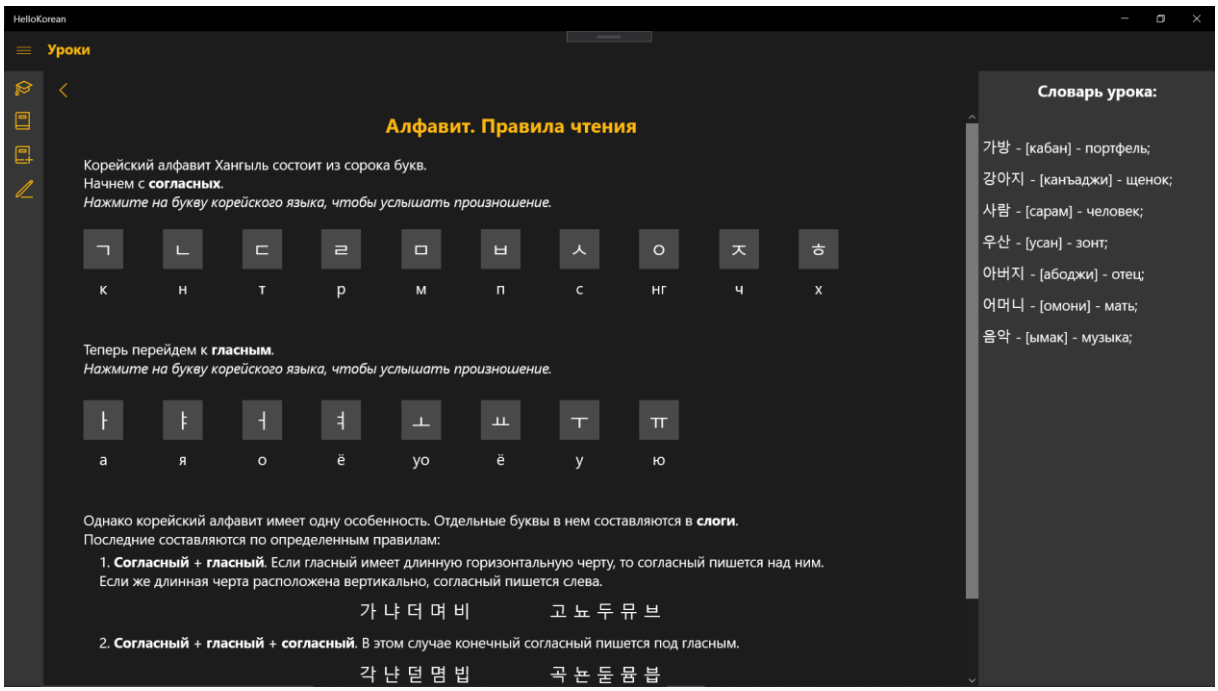

Рис. 23. Интерфейс урока

Вывод

В соответствии с задачами работы было реализовано приложение для ОС MS Windows. Был реализован пользовательский интерфейс, страница теста, страницы словаря слов и словаря грамматик, а также меню уроков и сами уроки.

5. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

Тестирование программного обеспечения [18] – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле тестирование – это одна из техник контроля качества, включающая в себя активности по планированию работ, проектированию тестов, выполнению тестирования и анализу полученных результатов.

5.1. Функциональное тестирование

Функциональное тестирование является одним из ключевых видов тестирования, задача которого – установить соответствие разработанного программного обеспечения исходным функциональным требованиям заказчика. Проведение функционального тестирования позволяет проверить способность информационной системы в определенных условиях решать задачи, нужные пользователям. Функциональное тестирование системы представлено в таблице 1.

Табл. 1. Функциональное тестирование системы

№	Действие	Ожидаемый результат	Результат теста
1	Пользователь нажимает кнопу меню	Меню раскрывается	Пройден
2	Пользователь нажимает кнопку «Уроки»	Открывается страница уроков и список уроков	Пройден
3	Пользователь нажимает кнопку «Словарь»	Открывается страница словаря и выводится словарь слов	Пройден
4	Пользователь вводит в поле для поиска слов слово на корейском языке и нажимает кнопку для поиска	Выводятся все слова содержащие искомое слово	Пройден

Окончание табл. 1

№	Действие	Ожидаемый результат	Результат теста
5	Пользователь вводит в поле для поиска слов перевод слова на и нажимает кнопку для поиска	Выводятся все слова с соответствующим переводом	Пройден
6	Пользователь вводит слово и его перевод, нажимает кнопку для добавления нового слова	Новое слово сохраняется в базу данных и выводится обновленный словарь	Пройден
7	Пользователь нажимает кнопку «Тест»	Открывается страница теста и выводится первое задание	Пройден
8	Пользователь нажимает на один из ответов к заданию	Выводится сообщение о правильности ответа и комментарий	Пройден
9	Пользователь нажимает кнопку для завершения теста	Тест завершается и выводится сообщение с результатами теста	Пройден

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы были решены следующие задачи:

1) выполнен анализ предметной области
2) произведен обзор существующих решений;
3) произведено проектирование приложения;
4) реализовано приложение для изучения корейского языка для ОС MS Windows;
5) протестировано приложение.

ЛИТЕРАТУРА

1. What are the Most Spoken Languages in the World? [Электронный ресурс] URL: https://www.fluentin3months.com/most-spoken-languages/ (Дата обращения: 01.06.2020).

2. 15 Best Languages to Learn in 2020. [Электронный ресурс] URL: https://www.tomedes.com/translator-hub/15-best-languages/ (Дата обращения: 01.06.2020).

3. Economy. Korea ranks 12th in world GDP ranking: World Bank. [Электронный ресурс] URL: http://www.theinvestor.co.kr/view.php?ud=20190707000187/ (Дата обращения: 01.06.2020).

4. 6000 Слов – Учим Корейский Бесплатно с FunEasyLearn. Официальный сайт. [Электронный ресурс] URL: https://www.microsoft.com/ru-ru/p/6000-Слов-Учим-Корейский-Бесплатно-с-funeasylearn/9nblggh5zckr?activetab=pivot:overviewtab/ (Дата обращения: 01.06.2020).

5. Приложение «Учим языки с LinGo Play». Официальный сайт. [Электронный ресурс] URL: https://apps.apple.com/ru/app/учить-языки-с-lingo-play/id969976197/ (Дата обращения: 01.06.2020).

6. Приложение «Корейский язык с Nemo». Официальный сайт. [Электронный ресурс] URL: https://apps.apple.com/ru/app/корейский-язык-с-nemo/id487077174/ (Дата обращения: 01.06.2020).

7. Visual Studio. Официальный сайт. [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019/ (Дата обращения: 01.06.2020).

8. Обзор языка XAML. Официальный сайт. [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/windows/uwp/xaml-platform/xaml-overview/ (Дата обращения: 01.06.2020).

9. Документация по C#. Официальный сайт. [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (Дата обращения: 01.06.2020).

10. JetBrains IntelliJ IDEA. Официальный сайт. [Электронный ресурс] URL: https://www.jetbrains.com/ru-ru/idea/ (Дата обращения: 01.06.2020).

11. PyCharm. Официальный сайт. [Электронный ресурс] URL: https://www.jetbrains.com/ru-ru/pycharm/ (Дата обращения: 01.06.2020).

12. What Is SQLite? Официальный сайт. [Электронный ресурс] URL: https://www.sqlite.org/index.html/ (Дата обращения: 01.06.2020).

13. Фаулер М. Основы UML. Краткое руководство по стандартному языку объектного моделирования, 3-е изд.// «Символ Плюс», 2005. –126 с.

14. Фаулер М. Основы UML. Краткое руководство по стандартному языку объектного моделирования, 3-е изд.// «Символ Плюс», 2005. – 52 с.

15. Фаулер М. Основы UML. Краткое руководство по стандартному языку объектного моделирования, 3-е изд.// «Символ Плюс», 2005. – 139 с.

16. Общие сведения о Microsoft.Data.Sqlite. Официальный сайт. [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/dotnet/standard/data/sqlite/?tabs=netcore-cli/ (Дата обращения: 01.06.2020).

17. Microsoft.NETCore.UniversalWindowsPlatform. Официальный сайт. [Электронный ресурс] URL: https://libraries.io/nuget/Microsoft.NETCore.UniversalWindowsPlatform/ (Дата обращения: 01.06.2020).

18. Тестирование программного обеспечения - основные понятия и определения. [Электронный ресурс] URL: http://www.protesting.ru/testing/ (Дата обращения: 01.06.2020).

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А. Листинг методов класса Test

Листинг А.1. Листинг методов *ItemAnswer1_Click, ItemAnswer2_Click, ItemAnswer3_Click, ItemAnswer4_Click*

```
private void ItemAnswer1_Click(object sender, RoutedEventArgs e)
        {
            Rectangle.Visibility = Visibility.Visible;
            if (Answer == 1)
            {
                AnswerTextBlock.Text = "Ваш ответ верный";
                CommentTextBlock.Text = Comment;
                FinalMark += Mark;
            }
            else
            {
                AnswerTextBlock.Text = "Ваш ответ неверный";
                CommentTextBlock.Text = Comment;
            }
        }

        private void ItemAnswer2_Click(object sender, RoutedEventArgs e)
        {
            Rectangle.Visibility = Visibility.Visible;
            if (Answer == 2)
            {
                AnswerTextBlock.Text = "Ваш ответ верный";
                CommentTextBlock.Text = Comment;
                FinalMark += Mark;
            }
            else
            {
                AnswerTextBlock.Text = "Ваш ответ неверный";
                CommentTextBlock.Text = Comment;
            }
        }

        private void ItemAnswer3_Click(object sender, RoutedEventArgs e)
        {
            Rectangle.Visibility = Visibility.Visible;
            if (Answer == 3)
```

```
            {
                AnswerTextBlock.Text = "Ваш ответ верный";
                CommentTextBlock.Text = Comment;
                FinalMark += Mark;
            }
            else
            {
                AnswerTextBlock.Text = "Ваш ответ неверный";
                CommentTextBlock.Text = Comment;
            }
        }

        private void ItemAnswer4_Click(object sender, RoutedEventArgs e)
        {
            Rectangle.Visibility = Visibility.Visible;
            if (Answer == 4)
            {
                AnswerTextBlock.Text = "Ваш ответ верный";
                CommentTextBlock.Text = Comment;
                FinalMark += Mark;
            }
            else
            {
                AnswerTextBlock.Text = "Ваш ответ неверный";
                CommentTextBlock.Text = Comment;
            }
        }
```

Листинг А.2. Листинг метода *NextBlockButton_Click*

```
        private void NextBlockButton_Click(object sender, RoutedEventArgs
e)
        {
            Rectangle.Visibility = Visibility.Collapsed;
            switch (dbCounter)
            {
                case 2:
                    Complexity = 1;
                    TestList.ItemsSource = GetTest();
                    AnswerTextBlock.Text = "";
                    CommentTextBlock.Text = "";
                    IdRandom();
```

```
            i = 0;
            break;
        case 3:
            Complexity = 2;
            TestList.ItemsSource = GetTest();
            AnswerTextBlock.Text = "";
            CommentTextBlock.Text = "";
            break;
        case 4:
            Complexity = 2;
            TestList.ItemsSource = GetTest();
            AnswerTextBlock.Text = "";
            CommentTextBlock.Text = "";
            IdRandom();
            i = 0;
            break;
        case 5:
            Complexity = 3;
            TestList.ItemsSource = GetTest();
            AnswerTextBlock.Text = "";
            CommentTextBlock.Text = "";
            break;
        case 6:
            Complexity = 3;
            TestList.ItemsSource = GetTest();
            AnswerTextBlock.Text = "";
            CommentTextBlock.Text = "";
            IdRandom();
            i = 0;
            break;
        case 7:
            Complexity = 4;
            TestList.ItemsSource = GetTest();
            AnswerTextBlock.Text = "";
            CommentTextBlock.Text = "";
            break;
        case 8:
            Complexity = 4;
            TestList.ItemsSource = GetTest();
            AnswerTextBlock.Text = "";
            CommentTextBlock.Text = "";
            IdRandom();
```

```
            case 9:
                Complexity = 5;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                break;
            case 10:
                Complexity = 5;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                IdRandom();
                i = 0;
                break;
            case 11:
                Complexity = 6;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                break;
            case 12:
                Complexity = 6;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                IdRandom();
                i = 0;
                break;
            case 13:
                Complexity = 7;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                break;
            case 14:
                Complexity = 7;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                IdRandom();
                i = 0;
                break;
```

```
            case 15:
                Complexity = 8;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                break;
            case 16:
                Complexity = 8;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                IdRandom();
                i = 0;
                break;
            case 17:
                Complexity = 9;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                break;
            case 18:
                Complexity = 9;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                IdRandom();
                i = 0;
                break;
            case 19:
                Complexity = 10;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                break;
            case 20:
                Complexity = 10;
                TestList.ItemsSource = GetTest();
                AnswerTextBlock.Text = "";
                CommentTextBlock.Text = "";
                IdRandom();
                i = 0;
                break;
```

```
                case 21:
                    Complexity = 11;
                    TestList.ItemsSource = GetTest();
                    AnswerTextBlock.Text = "";
                    CommentTextBlock.Text = "";
                    break;
                case 22:
                    Complexity = 11;
                    TestList.ItemsSource = GetTest();
                    AnswerTextBlock.Text = "";
                    CommentTextBlock.Text = "";
                    IdRandom();
                    i = 0;
                    break;
                case 23:
                    Complexity = 12;
                    TestList.ItemsSource = GetTest();
                    AnswerTextBlock.Text = "";
                    CommentTextBlock.Text = "";
                    break;
                case 24:
                    Complexity = 12;
                    TestList.ItemsSource = GetTest();
                    AnswerTextBlock.Text = "";
                    CommentTextBlock.Text = "";
                    IdRandom();
                    i = 0;
                    NextBlockButton.Visibility = Visibility.Collapsed;
                    FinishButton.Visibility = Visibility.Visible;
                    break;
            }
        }
```

ПРИЛОЖЕНИЕ Б. Листинги страниц XAML

Листинг Б.1. Листинг страницы *MainPage*

```
<Page
    x:Class="HelloKorean.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HelloKorean"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="#1C1C1C"
          HorizontalAlignment="Stretch">
        <Grid.RowDefinitions>
            <RowDefinition Height="50"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <Grid Grid.Row="0"
              Background="#1C1C1C">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="50"/>
                <ColumnDefinition Width="*"/>
            </Grid.ColumnDefinitions>
            <Button x:Name="MenuButton"
                    FontFamily="Segoe MDL2 Assets"
                    FontSize="18"
                    Content="&#xE700;"
                    Foreground="#FFB90F"
                    VerticalAlignment="Stretch"
                    HorizontalAlignment="Stretch"
                    Grid.Column="0"
                    Click="MenuButton_Click"
                    Background="{x:Null}"/>
            <TextBlock x:Name="HeaderText"
                       Grid.Column="1"
                       Margin="5 10 0 0"
                       Foreground="#FFB90F"
                       FontWeight="Bold"
                       FontSize="18"/>
        </Grid>
```

```
        <SplitView x:Name="MenuSplitView"
                   Grid.Row="1"
                   OpenPaneLength="200"
                   CompactPaneLength="50"
                   DisplayMode="CompactOverlay">
            <SplitView.Pane>
                <ListBox x:Name="MenuListBox"
                         Background="#363636"
                         SelectionChanged="MenuListBox_SelectionChanged">
                    <ListBoxItem x:Name="ItemLessons">
                        <StackPanel Orientation="Horizontal">
                            <TextBlock Style="{StaticResource ItemIcon}"
                                       Text="&#xE7BE;"/>
                            <TextBlock Text="Уроки"
                                       Foreground="#FFB90F"/>
                        </StackPanel>
                    </ListBoxItem>
                    <ListBoxItem x:Name="ItemDictionary">
                        <StackPanel Orientation="Horizontal">
                            <TextBlock Style="{StaticResource ItemIcon}"
                                       Text="&#xE82D;"/>
                            <TextBlock Text="Словарь"
                                       Foreground="#FFB90F"/>
                        </StackPanel>
                    </ListBoxItem>
                    <ListBoxItem x:Name="ItemGrammarDictionary">
                        <StackPanel Orientation="Horizontal">
                            <TextBlock Style="{StaticResource ItemIcon}"
                                       Text="&#xE82E;"/>
                            <TextBlock Text="Словарь грамматик"
                                       Foreground="#FFB90F"/>
                        </StackPanel>
                    </ListBoxItem>
                    <ListBoxItem x:Name="ItemTest">
                        <StackPanel Orientation="Horizontal">
                            <TextBlock Style="{StaticResource ItemIcon}"
                                       Text="&#xEC87;"/>
                            <TextBlock Text="Тест" Foreground="#FFB90F"/>
                        </StackPanel>
                    </ListBoxItem>
                </ListBox>
            </SplitView.Pane>
```

```
            <SplitView.Content>
                <Frame x:Name="MenuFrame" />
            </SplitView.Content>
        </SplitView>
    </Grid>
</Page>
```

Листинг Б.2. Листинг страницы *Dictionary*

```
<Page
    x:Class="HelloKorean.Dictionary"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HelloKorean"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    mc:Ignorable="d">
    <Grid Background="#1C1C1C">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="300"/>
        </Grid.ColumnDefinitions>
        <RelativePanel Grid.Column="0"
                       Width="1000"
                       Height="750"
                       Margin="85 2 0 20"
                       HorizontalAlignment="Left"
                       VerticalAlignment="Top">
            <ListView Name="WordsList"
                      RelativePanel.AlignHorizontalCenterWithPanel="True"
                      SelectionMode="Single"
                      ScrollViewer.VerticalScrollBarVisibility="Auto"
                      ScrollViewer.IsVerticalRailEnabled="True"
                      ScrollViewer.VerticalScrollMode="Enabled"
                      ScrollViewer.HorizontalScrollMode="Enabled"
                      ScrollViewer.HorizontalScrollBarVisibility="Auto"
                      ScrollViewer.IsHorizontalRailEnabled="True"
                      Margin="50 20 20 20">
                <ListView.HeaderTemplate>
                    <DataTemplate>
                        <StackPanel Orientation="Horizontal"  >
```

```
                    <TextBlock Text="Слово"
                               Width="300"
                               Foreground="#9C9C9C"
                               Margin="0 0 0 5"
                               FontSize="25" />
                    <TextBlock Text="Перевод"
                               Width="400"
                               Margin="0 0 0 5"
                               Foreground="#9C9C9C"
                               FontSize="25"/>
                </StackPanel>
            </DataTemplate>
        </ListView.HeaderTemplate>
        <ListView.ItemTemplate>
            <DataTemplate x:DataType="local:Words">
                <StackPanel Orientation="Horizontal"
                            Margin="0 5 0 5">
                    <TextBlock Name="ItemName"
                               Text="{x:Bind KoreanWord}"
                               Width="300"
                               Foreground="White"
                               FontSize="22"/>
                    <TextBlock Text="{x:Bind RussianWord}"
                               Width="400"
                               Foreground="White"
                               FontSize="22"/>
                </StackPanel>
            </DataTemplate>
        </ListView.ItemTemplate>
    </ListView>
</RelativePanel>
<RelativePanel Grid.Column="1"
               HorizontalAlignment="Right"
               Background="#363636">
    <Button x:Name="SearchButton"
            Click="SearchButton_Click"
            RelativePanel.AlignRightWithPanel="True"
            Width="50"
            Height="50"
            Margin="0 15 15 15"
            BorderThickness="0"
            FontFamily="Segoe MDL2 Assets"
```

```
        Content="&#xE094;"
        FontSize="20"/>
<TextBox x:Name="SearchTextBox"
         RelativePanel.LeftOf="SearchButton"
         Width="200"
         Height="50"
         Margin="15 15 0 15"
         Padding="14"
         PlaceholderText="Поиск"/>
<Button x:Name="AddButton"
        Click="AddButton_Click"
        RelativePanel.AlignRightWithPanel="True"
        Width="250"
        Height="50"
        Margin="0 280 15 0"
        BorderThickness="0"
        FontFamily="Segoe MDL2 Assets"
        Content="&#xE792;"
        FontSize="22"/>
<TextBlock Text="Добавить слово :"
           Foreground="#9C9C9C"
           FontSize="18"
           Margin="81 110 0 0"/>
<TextBox x:Name="AddKoreanTextBox"
         RelativePanel.AlignRightWithPanel="True"
         Width="250"
         Height="50"
         Margin="0 150 15 0"
         Padding="14"
         PlaceholderText="Слово"/>
<TextBox x:Name="AddRussianTextBox"
         RelativePanel.AlignRightWithPanel="True"
         Width="250"
         Height="50"
         Margin="0 215 15 0"
         Padding="14"
         PlaceholderText="Перевод"/>
<Button x:Name="RefreshButton"
        Click="RefreshButton_Click"
        RelativePanel.AlignRightWithPanel="True"
        Width="250"
        Height="50"
```

```
                    Margin="0 345 15 0"
                    BorderThickness="0"
                    FontFamily="Segoe MDL2 Assets"
                    Content="&#xE117;"
                    FontSize="22"/>
            <TextBlock Text="*(ув) - уважительная форма слова,
используется в формальном стиле и во время обращения к людям старшим по
возрасту или должности."
                       RelativePanel.AlignRightWithPanel="True"
                       TextWrapping="Wrap"
                       Width="250"
                       Foreground="#9C9C9C"
                       FontSize="18"
                       Margin="0 415 15 0"/>
        </RelativePanel>
    </Grid>
</Page>
```

Листинг Б.3. Листинг страницы *GrammarDictionary*

```
<Page
    x:Class="HelloKorean.GrammarDictionary"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HelloKorean"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="#1C1C1C">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="300"/>
        </Grid.ColumnDefinitions>
        <RelativePanel Grid.Column="0"
                       Width="1000"
                       Height="750"
                       Margin="85 2 0 20"
                       HorizontalAlignment="Left"
                       VerticalAlignment="Top">
            <ListView Name="GrammarList"
```

```
        SelectionMode="Single"
        ScrollViewer.VerticalScrollBarVisibility="Auto"
        ScrollViewer.IsVerticalRailEnabled="True"
        ScrollViewer.VerticalScrollMode="Enabled"
        ScrollViewer.HorizontalScrollMode="Enabled"
        ScrollViewer.HorizontalScrollBarVisibility="Auto"
        ScrollViewer.IsHorizontalRailEnabled="True"
        Margin="50 20 20 20">
    <ListView.HeaderTemplate>
        <DataTemplate>
            <StackPanel Orientation="Horizontal"
                        Margin="0 0 0 15" >
                <TextBlock Text="Грамматика"
                           Width="200"
                           Foreground="#9C9C9C"
                           FontSize="22" />
                <TextBlock Text="Пример"
                           Width="400"
                           Foreground="#9C9C9C"
                           FontSize="22"/>
                <TextBlock Text="Перевод"
                           Width="300"
                           Foreground="#9C9C9C"
                           FontSize="22"/>
            </StackPanel>
        </DataTemplate>
    </ListView.HeaderTemplate>
    <ListView.ItemTemplate>
        <DataTemplate x:DataType="local:Grammar">
            <StackPanel Orientation="Horizontal"
                        Margin="0 5 0 5">
                <TextBlock Name="ItemName"
                           Text="{x:Bind Grammars}"
                           Width="200"
                           Foreground="White"
                           FontSize="20"/>
                <TextBlock Name="ItemExample"
                           Text="{x:Bind Example}"
                           Width="400"
                           Foreground="White"
                           FontSize="20"/>
                <TextBlock Text="{x:Bind Translation}"
```

```
                                Width="400"
                                Foreground="White"
                                FontSize="20"/>
                    </StackPanel>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </RelativePanel>
    <RelativePanel Grid.Column="1"
                   HorizontalAlignment="Right"
                   Background="#363636">
        <Button x:Name="SearchButton"
                Click="SearchButton_Click"
                RelativePanel.AlignRightWithPanel="True"
                Width="50"
                Height="50"
                Margin="0 15 15 15"
                BorderThickness="0"
                FontFamily="Segoe MDL2 Assets"
                Content="&#xE094;"
                FontSize="20"/>
        <TextBox x:Name="SearchTextBox"
                 RelativePanel.LeftOf="SearchButton"
                 Width="200"
                 Height="50"
                 Margin="15 15 0 15"
                 Padding="14"
                 PlaceholderText="Поиск"/>
        <Button x:Name="AddButton"
                Click="AddButton_Click"
                RelativePanel.AlignRightWithPanel="True"
                Width="250"
                Height="50"
                Margin="0 345 15 0"
                BorderThickness="0"
                FontFamily="Segoe MDL2 Assets"
                Content="&#xE792;"
                FontSize="22"/>
        <TextBlock Text="Добавить грамматику :"
                   Foreground="#9C9C9C"
                   FontSize="18"
                   Margin="55 110 0 0"/>
```

```
            <TextBox x:Name="AddGrammarTextBox"
                     RelativePanel.AlignRightWithPanel="True"
                     Width="250"
                     Height="50"
                     Margin="0 150 15 0"
                     Padding="14"
                     PlaceholderText="Грамматика"/>
            <TextBox x:Name="AddExampleTextBox"
                     RelativePanel.AlignRightWithPanel="True"
                     Width="250"
                     Height="50"
                     Margin="0 215 15 0"
                     Padding="14"
                     PlaceholderText="Пример"/>
            <TextBox x:Name="AddTranslationTextBox"
                     RelativePanel.AlignRightWithPanel="True"
                     Width="250"
                     Height="50"
                     Margin="0 280 15 0"
                     Padding="14"
                     PlaceholderText="Перевод"/>
            <Button x:Name="RefreshButton"
                    Click="RefreshButton_Click"
                    RelativePanel.AlignRightWithPanel="True"
                    Width="250"
                    Height="50"
                    Margin="0 410 15 0"
                    BorderThickness="0"
                    FontFamily="Segoe MDL2 Assets"
                    Content="&#xE117;"
                    FontSize="22"/>
            <TextBlock Text="*(ув) - уважительный, грамматика
используется в формальном стиле и во время обращения к людям старшим по
возрасту или должности."
                       RelativePanel.AlignRightWithPanel="True"
                       TextWrapping="Wrap"
                       Width="250"
                       Foreground="#9C9C9C"
                       FontSize="18"
                       Margin="0 475 15 0"/>
        </RelativePanel>
    </Grid></Page>
```

Листинг Б.4. Листинг страницы *Test*

```
<Page
    x:Class="HelloKorean.Test"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HelloKorean"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    mc:Ignorable="d">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="400"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="100"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <RelativePanel Grid.Column="0"
                       Grid.Row="0"
                       Background="#363636"
                       Margin="85 2 0 20">
            <TextBlock Text="Данные тест не дает 100% результат, но может
помочь приблизительно узнать свой уровень знания языка. Для более точно
результата рекомендуется во время прохождения теста не пользоваться
словарями, учебниками и сторонней помощью. Так же данный тест может
помочь в подготовке к топику, так как большая часть заданий схожи с
заданиями топика."
                       TextWrapping="Wrap"
                       Margin="5"

RelativePanel.AlignHorizontalCenterWithPanel="True"
                       Foreground="#FFB90F"/>
        </RelativePanel>
        <RelativePanel x:Name="TestRelativePanel"
                       Grid.Column="0"
                       Grid.Row="1"
                       Margin="85 2 0 20"
```

```
                Width="1000"
                Height="650"
                HorizontalAlignment="Left"
                VerticalAlignment="Top">
        <ListView x:Name="TestList"
                  Height="550"
                  Width="1000"
                  BorderBrush="#FFB90F"
                  BorderThickness="1">
            <ListView.ItemTemplate>
                <DataTemplate x:DataType="local:TestLevel">
                    <StackPanel x:Name="TaskStackPanel"
                                Orientation="Vertical"
                                Height="550"
                                Width="1000">
                        <TextBlock x:Name="NumberTask"
                                   Text="{x:Bind NumberTask}"
                                   Foreground="White"
                                   FontSize="20"
                                   Margin="0 2 0 5"
                                   HorizontalAlignment="Center"/>
                        <TextBlock x:Name="ItemTask"
                                   Text="{x:Bind Task}"
                                   Foreground="White"
                                   FontSize="25"/>
                        <CheckBox x:Name="ItemAnswer1"
                                  Click="ItemAnswer1_Click"
                                  Content="{x:Bind Answer1}"
                                  Foreground="White"
                                  FontSize="25"/>
                        <CheckBox x:Name="ItemAnswer2"
                                  Click="ItemAnswer2_Click"
                                  Content="{x:Bind Answer2}"
                                  Foreground="White"
                                  FontSize="25"/>
                        <CheckBox x:Name="ItemAnswer3"
                                  Click="ItemAnswer3_Click"
                                  Content="{x:Bind Answer3}"
                                  Foreground="White"
                                  FontSize="25"/>
                        <CheckBox x:Name="ItemAnswer4"
                                  Click="ItemAnswer4_Click"
```

```
                                Content="{x:Bind Answer4}"
                                Foreground="White"
                                FontSize="25"/>
                    </StackPanel>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
        <Rectangle x:Name="Rectangle"
                   Fill="Transparent"
                   Width="1000"
                   Height="500"
                   Visibility="Collapsed"/>
        <Button x:Name="NextBlockButton"
                Click="NextBlockButton_Click"
                Width="250"
                Height="50"
                RelativePanel.AlignBottomWithPanel="True"
                RelativePanel.AlignHorizontalCenterWithPanel="True"
                FontFamily="Segoe MDL2 Assets"
                Content="&#xE0AB;"
                FontSize="22"/>
        <Button x:Name="FinishButton"
                Click="FinishButton_Click"
                Width="250"
                Height="50"
                Margin="275 370 225 0"
                FontFamily="Segoe MDL2 Assets"
                Content="&#xE001;"
                FontSize="22"
                RelativePanel.AlignBottomWithPanel="True"
                RelativePanel.AlignHorizontalCenterWithPanel="True"
                Visibility="Collapsed"/>
        <TextBlock x:Name="FinishTextBlock"
                   Visibility="Collapsed"
                   Margin="50 20 50 20"
                   Width="1000"
                   Height="300"
                   Foreground="White"/>
    </RelativePanel>
    <RelativePanel Grid.Column="1"
                   Grid.RowSpan="2"
                   HorizontalAlignment="Right"
```

```
                        Background="#363636">
            <TextBlock x:Name="AnswerTextBlock"
                        Foreground="#FFB90F"
                        FontSize="18"
                        Width="Auto"
                        Height="50"

RelativePanel.AlignHorizontalCenterWithPanel="True"
                        Margin="0 20 0 20"/>
            <TextBlock x:Name="CommentTextBlock"
                        Foreground="White"
                        Width="290"
                        Height="500"
                        FontSize="18"
                        Margin="10 85 0 20"/>
        </RelativePanel>
    </Grid>
</Page>
```