**Оглавление**

[Аннотация 4](#_Toc8739487)

[Введение 5](#_Toc8739488)

[1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ 6](#_Toc8739489)

[1.1 Основное функциональное назначение программного средства 6](#_Toc8739490)

[1.2 Полное наименование программного средства 6](#_Toc8739491)

[1.3 Условное обозначение программного средства 6](#_Toc8739492)

[1.4 Разработчики программного средства 6](#_Toc8739493)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 7](#_Toc8739494)

[2.1 Основание для разработки 7](#_Toc8739495)

[2.2 Назначение разработки 7](#_Toc8739496)

[2.3 Требование к программному средству 7](#_Toc8739497)

[2.4 Требования к программной документации 8](#_Toc8739498)

[2.5 Требования к эргономике и технической эстетике 8](#_Toc8739499)

[2.6 Стадии и этапы разработки 9](#_Toc8739500)

[2.7 Порядок контроля и приемки 10](#_Toc8739501)

[3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ 11](#_Toc8739502)

[3.1 Декомпозиция поставленной задачи 11](#_Toc8739503)

[3.2 Общая архитектура программного средства 12](#_Toc8739504)

[3.3 Разработка алгоритма решения задачи 15](#_Toc8739505)

[3.4 Реализация функционального назначения программного средства 16](#_Toc8739506)

[3.5 Структурная организация данных 17](#_Toc8739507)

[3.6 Разработка интерфейса ПС 22](#_Toc8739508)

[3.7 Описание структуры выходной информации 32](#_Toc8739509)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 33](#_Toc8739510)

[4.1 Назначение программного средства 33](#_Toc8739511)

[4.2 Условия выполнения программного средства 33](#_Toc8739512)

[4.3 Эксплуатация программного средства 33](#_Toc8739513)

[4.4 Сообщения пользователю 46](#_Toc8739514)

[5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА 47](#_Toc8739515)

[5.1 Определение верхней границы количества тестов 47](#_Toc8739516)

[5.2 Разработка тестовых вариантов 48](#_Toc8739517)

[5.3 Составление отчета о тестировании 55](#_Toc8739518)

[ЗАКЛЮЧЕНИЕ 56](#_Toc8739519)

[Оценка качества программного средства с помощью метрик 56](#_Toc8739520)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 59](#_Toc8739521)

[Приложение А Текст кода программного средства 61](#_Toc8739522)

[Приложение Б Спецификация 81](#_Toc8739523)

[Приложение B Диск с программным средством 83](#_Toc8739524)

**Аннотация**

На дипломную работу студента Карягина А.С. на тему: АИС «Школа английского языка “ABC”(подсистема работы с услугами)» Дипломная работа представлена пояснительной запиской на 83 листа машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В главе 1 содержится общие сведения о программном средстве АИС « Школа английского языка “ABC”(подсистема работы с услугами)». В главе 2 приведено техническое задание на проектирование БД согласно ГОСТ 19.201. Глава 3 содержит пояснительную записку к программному продукту согласно ГОСТ 19.404-79. В главе 4 содержится руководство пользователя согласно ГОСТ 19.504-79. Глава 5 содержит обоснование количества необходимых тестов и тестовые пакеты для оценки работоспособности программного продукта. В главе 6 приведены результаты ручного и автоматического тестирования программного продукта и выполнена оценка качества с помощью метрик.

В работе содержится общее описание программного средства, предназначенного для объекта проектирования – АИС Школа английского языка “ABC”(подсистема работы с услугами)», изложены требования к программному средству, описаны его архитектура, реализация функций, интерфейс, приведено руководство пользователя, обоснованы тестовые пакеты и выполнена оценка качества программного продукта с помощью метрик.

**Введение**

В дипломной работе была поставлена задача разработки АИС Школа английского языка “ABC”(подсистема работы с услугами)», который позволяет автоматизировать деятельность школы по автоматизации работы с услугами.

Для достижения поставленной цели был составлен основной алгоритм работы программы, а также разработаны алгоритмы вспомогательных процедур.

Для реализации алгоритмов был выбран язык программирования C#. Выбор обусловлен тем, что возможностей этого языка достаточно для достижения поставленной цели моей дипломной работы.

При разработке интерфейса программы, я учитывал наиболее нужные и важные функциональные возможности, которые должна выполнять данная программа, а также тип пользователей, которые с ней будут работать, в данном случае работник школы.

**1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ**

**1.1 Основное функциональное назначение программного средства**

Программа предназначена для автоматизации деятельности школы. Программа не требует владения современными информационными технологиями для эффективного использования данного программного средства, т.е. подходит для любого пользователя.

**1.2 Полное наименование программного средства**

АИС «Школа английского языка “ABC”(подсистема работы с услугами)»

**1.3 Условное обозначение программного средства**

«EnglishSchool 1.0»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 185 Карягин А.С.

**2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**2.1 Основание для разработки**

Работа ведётся на основании задания на дипломную работу по специальности 09.02.03 — «Программирование в компьютерных системах».

**2.2 Назначение разработки**

Программа «EnglishSchool 1.0» работает под управление операционной системы Windows 7, Windows 8, Windows 10. Объект проектирования – база данных и модуль обработки данных школы английского языка “ABC”(подсистема работы с услугами). Подсистема для работы с услугами компании включает в себя следующий функционал:

• просмотр списка услуг;

• добавление/удаление/редактирование данных об услугах;

• запись клиента на услугу;

• просмотр истории оказания услуг школы иностранных языков.

**2.3 Требование к программному средству**

«EnglishSchool 1.0» устанавливается на персональный компьютер, имеющий представленные минимальные системные требования:

Оперативная память: 4 Гб;

Процессор: Intel Core i / Phenom II X4 / AMD Athlon64 X2 4600 + 2.4 ГГц;

Видео карта: nVidia GeForce 8600 / ATI Radeon HD 2400;

Звуковая карта: Совместимая с DirectX;

Программный продукт «EnglishSchool 1.0» является кроссплатформенным продуктом, что позволяет не ограничиваться в работе с определенной версией ОС.

«EnglishSchool 1.0» написана на объектно-ориентированном языке программировании C# (си-шарп) в интегрированной среде разработки Visual Studio 2019. В качестве СУБД используется MS SQL. Программа не может подлежать сторонней оптимизации и редактированию программного кода.

**2.4 Требования к программной документации**

Состав программной документации должен включать в себя:

1. техническое задание;

2. сроки выполнения работ;

3. программу и методику испытаний;

4. эксплуатационные инструкции пользователю;

**2.5 Требования к эргономике и технической эстетике**

Система обеспечивает удобный для пользователей системный интерфейс, отвечающий следующим требованиям:

- единый стиль оформления для пользовательских интерфейсов;

- удобная, интуитивно понятная навигация в интерфейсе пользователя;

- взаимодействие пользователя с системой осуществляется на двух языках по выбору (русский и английский), для наибольшего удобства

Пользовательские интерфейсы системы спроектированы и разработаны с применением единых принципов графического представления информации и организации доступа к функциональным возможностям и сервисам. Разработан графический дизайн пользовательских интерфейсов, цветовые, шрифтовые и композиционные решения для отображения текстов, изображений, таблиц, гиперссылок, управляющих и навигационных элементов (меню, кнопок, форм и т.п.).

Экранные формы спроектированы с учетом требований унификации:

- все экранные формы пользовательского интерфейса выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;

- для обозначения сходных операций использованы сходные графические значки, кнопки и другие управляющие (навигационные) элементы. Термины, используемые для обозначения типовых операций (добавление информационной сущности, редактирование поля данных), а также последовательности действий пользователя при их выполнении унифицированы;

- внешнее поведение сходных элементов интерфейса (реакция на наведение указателя «мыши», переключение фокуса, нажатие кнопки) реализованы одинаково для однотипных элементов.

**2.6 Стадии и этапы разработки**

Разработка должна быть проведена в три стадии:

1. разработка технического задания;

2. рабочее проектирование;

3. внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;

2. разработка программной документации;

3. испытания программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;

2. определение и уточнение требований к техническим средствам;

3. определение требований к программе;

4. определение стадий, этапов и сроков разработки программы и документации на нее;

5. выбор языков программирования;

6. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с пунктом Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1. разработка, согласование и утверждение порядка и методики испытаний;

2. проведение испытаний;

3. корректировка программы и программной документации по результатам испытаний.

**2.7 Порядок контроля и приемки**

Сдача-приёмка работ производится поэтапно, в соответствии с рабочей программой и календарным планом. Сдача-приемка осуществляется комиссией, в состав которой входят представители ГАПОУ и работодатель. По результатам приемки подписывается акт приемочной комиссии.

В указанном подразделе, согласно таблице «Сроки выполнения работ» технического задания, будет описаны перечень участвующих организаций, место и сроки проведения работ, согласно п. 2.8 ГОСТ 34.602-89.

Порядок согласования и утверждения приемочной документации должен регламентироваться организационно-распорядительной документацией организации, принимающей участие в создании программного продукта. Согласно разделу «Приемка результатов разработки» ГОСТ 15.001-88 для согласования и утверждения приемочной документации создается приемочная комиссия (приказом).

Статус приемочной комиссии определяется Заказчиком до проведения испытаний.

**3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ**

**3.1 Декомпозиция поставленной задачи**

Для достижения цели работы была поставлена следующая задача:

Создать модуль “CodeDatabase” отвечающий за сохранение БД. Результат работы модуля - сохранение БД.

Поставленные задачи наглядно отображаются на диаграмме деятельности.

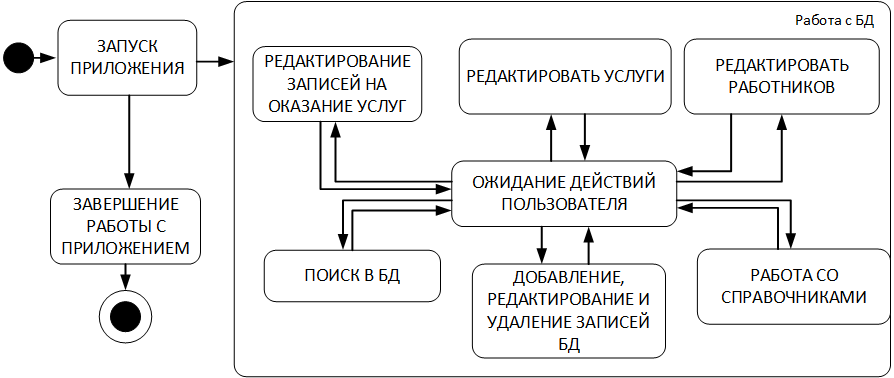


Рис.1 Декомпозиция поставленной задачи

**3.2 Общая архитектура программного средства**

Объектом моделирования является деятельность школы.

Рассматриваемые процессы: оказание услуг.

Объекты моделирования представлены на диаграмме классов, рис.2.

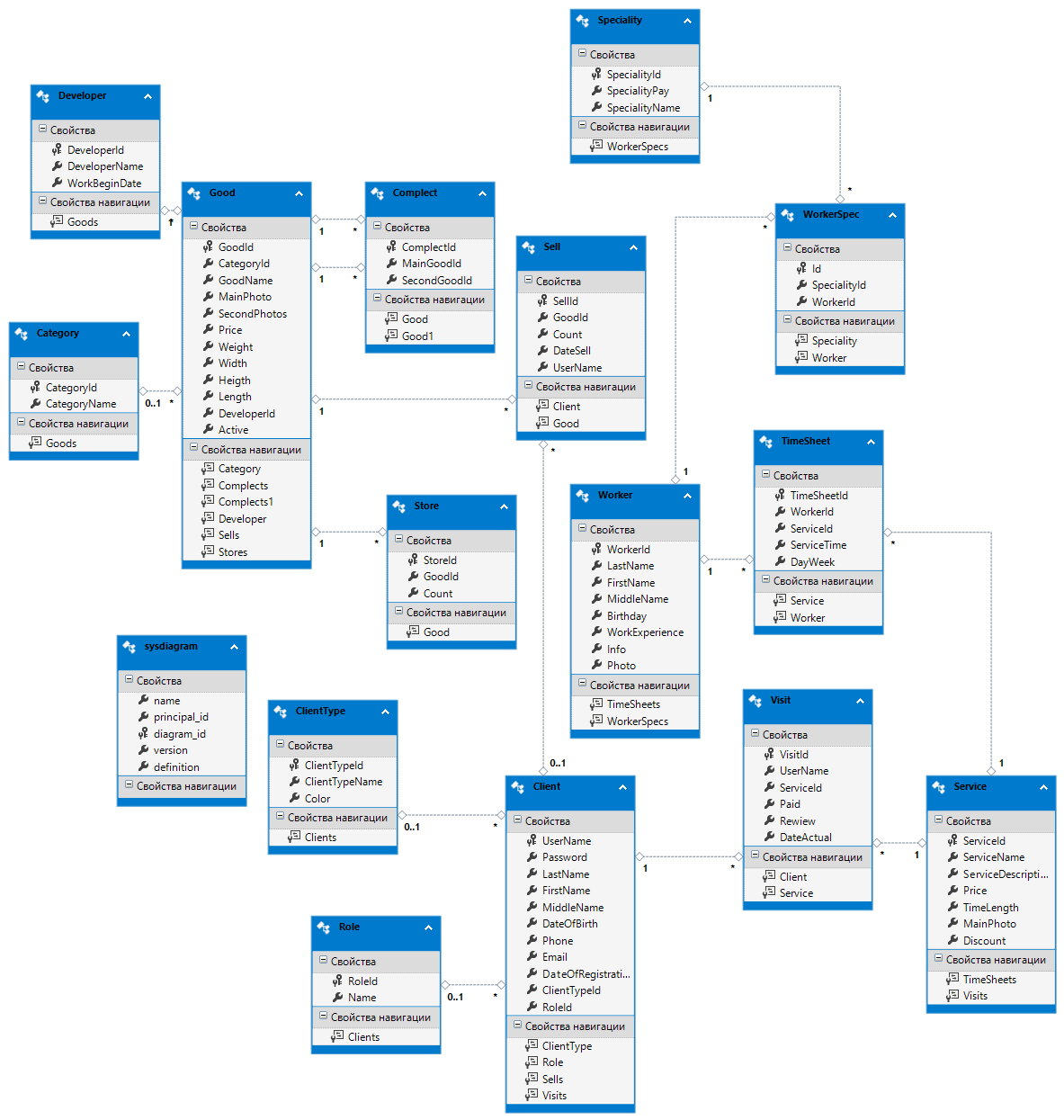


Рис.2 Диаграмма классов

По полученной декомпозиции задач была спроектирована следующая архитектура программного средства.

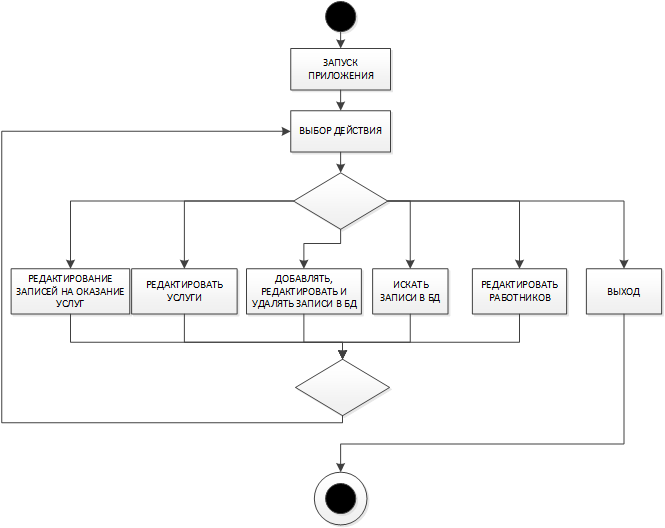


Рис.3 Схема событийно-ориентированной архитектуры

В «EnglishSchool 1.0» используется событийно-ориентированная архитектура.(Рис.3) Роль агента (источник событий) в программе выполняют: пользователи программы, за роль стоков (потребители событий) отвечают таблицы, входящие в базу данных. Например, когда пользователь выбирает определенное действие: добавить, удалить, редактировать, вывести в Excel и т.д. система осуществит выбранные действия, и база отреагирует соответствующим образом: запись добавлена, удалена, отредактирована, либо отчет выведен в Excel. Системная архитектура пользователя рассматривает это изменение состояния как событие, создаваемое, публикуемое, определяемое и потребляемое различными приложениями в составе архитектуры.

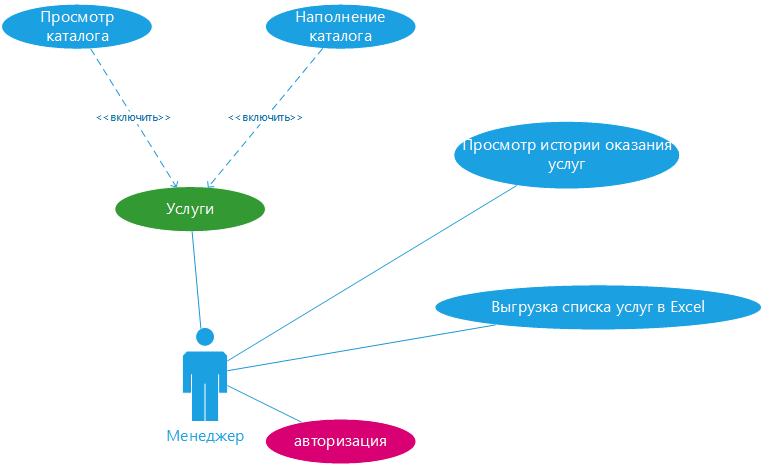


Рис.4 Диаграмма вариантов использования

**3.3 Разработка алгоритма решения задачи**

**Общий алгоритм**

****Рис.5 Алгоритм программы

**3.4 Реализация функционального назначения программного средства**

Программа «EnglishSchool 1.0» имеет следующий набор входных данных, такие как: услуги, работники, расписание.

Данные вводятся администратором в соответствующие поля ввода, снабженные всплывающими подсказками.

Выходными данными являются: списки услуг, расписание.

Данные вводятся пользователем в соответствующие поля вывода, снабженные всплывающими подсказками.

Выходные данные редактировать вручную невозможно, т.к. они служат в качестве отчетной информации для пользователя.

**3.5 Структурная организация данных**

Для создания БД необходимо определиться с данными, которые необходимы для полноценного функционирования системы. Все эти данные указаны в реляционной модели Информационная система «Организация проверки ассортимента магазина» представленная на рисунке. Любая реляционная база данных и называется реляционной, что характеризуется отношениями (relation) между таблицами. На рисунке изображены таблицы моей базы данных. При этом одна таблица является родительской (главной), а вторая – дочерней (подчиненной). Главной таблицей является «Услуга». Реляционная модель автоматизированной системы соответствует всем 12 правилам Кодда. (рис.6)

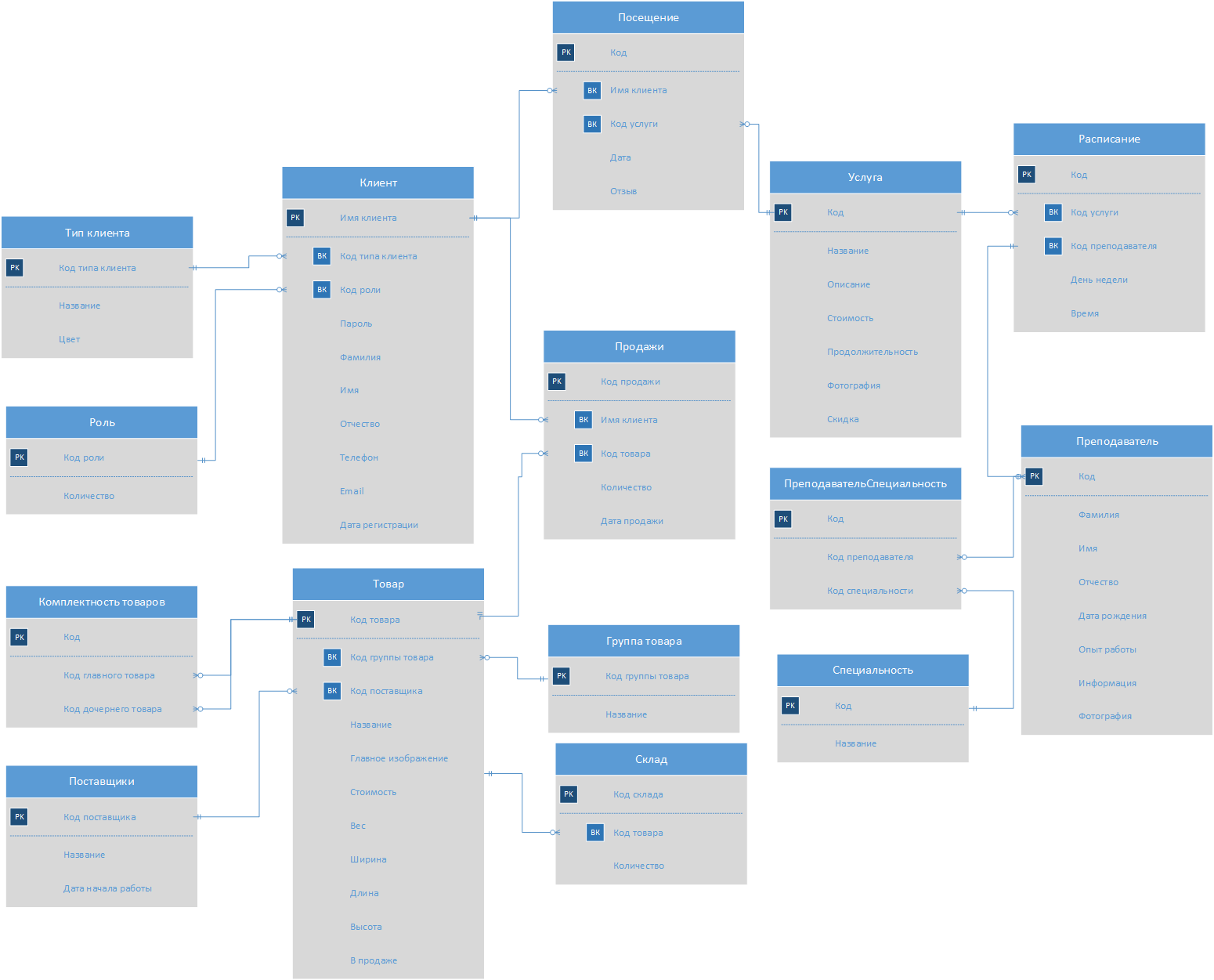


Рис.6 Реляционная модель

Первичный ключ в базе уникален, используется для организации отношений между таблицами, который не может иметь пустых и повторяющихся значений. Первичными ключами в базе являются поля: Код (Таблица Товар), Код (Таблица Группа товара), Код (Таблица Склад), Код (таблица Поставщик), Код (таблица Продажа), Код (Таблица Роль), Код (таблица Услуги), Код (таблица Посещение), Имя клиента (таблица Клиент), (таблица Услуга), Код (таблица Расписание), Код (таблица Преподаватель). Остальные ID-ключи являются внешними ключами.

Для организации более эффективной обработки данных применяется нормализация. Таблицы моей БД находятся в 3НФ:

* БД находится в форме -1НФ потому, что
  + Таблица не имеет повторяющихся записей;
  + Каждый атрибут отношения хранит одно-единственное значение и не является списком, ни множеством значений;
  + Таблица не имеет повторяющихся групп полей.
    - Вторая нормальная форма(2НФ):
      * Устранены атрибуты, зависящие только от части уникального (первичного) идентификатора, т.е. ID.
        + Третья нормальная форма(3НФ):

Отсутствуют атрибуты, зависящие от атрибутов, не входящих в уникальный (первичный) ключ.

На основе реляционной модели базы данных мною в MS SQL была построена следующая база данных. (Рис.7) В ней первичным ключом является поле таблицы “Good” – “GoodId”. Связь таблиц “Category” и “ Good ” по полю “CategoryID” один ко многим (1:М). Связь таблиц “Good” и “Store” по полю “GoodId” один ко многим (1:М). Связь таблиц “Good” и “Sell” по полю “GoodId” один ко многим (1:М). Связь таблиц “Good” и “Complect” по полям “MainGoodId” и “SecondGoodId” один ко многим (1:М). Связь таблиц “Client” и “Sell” по полю “UserName” один ко многим (1:М). Связь таблиц “ClientType” и “Client” по полю “ClientTypeName” один ко многим (1:М). Связь таблиц “Role” и “Client” по полю “RoleId” один ко многим (1:М). Связь таблиц “Client” и “Visit” по полю “UserName” один ко многим (1:М). Связь таблиц “Service” и “Visit” по полю “ServiceId” один ко многим (1:М). Связь таблиц “Service” и “TimeSheet” по полю “ServiceId” один ко многим (1:М). Связь таблиц “Worker” и “TimeSheet” по полю “WorkerId” один ко многим (1:М). Связь таблиц “Worker” и “WorkerSpec” по полю “WorkerId” один ко многим (1:М). Связь таблиц “Speciality” и “WorkerSpec” по полю “SpecialityId” один ко многим (1:М). Все связи между таблицами базы данных типа один ко многим.



Рис.7 Схема базы данных

Таблица “Service”(Услуга)



Рис.8 Таблица Service

Таблица “Speciality”(Сециальность)

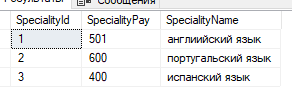


Рис.9 Таблица Speciality

Таблица “Worker”(Работники)

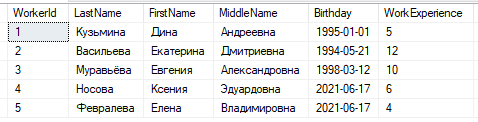


Рис.10 Таблица Worker

Таблица “TimeSheet”(Расписание)

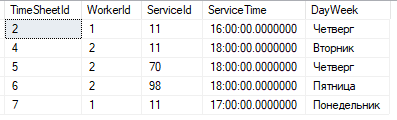


Рис.11 Таблица TimeSheet

Таблица “Visit”(Посещение)

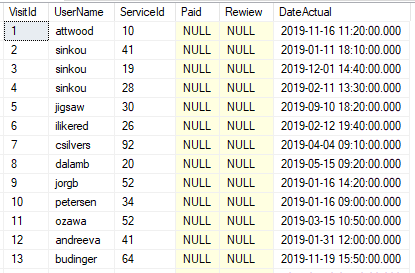


Рис.14 Таблица Visit

Таблица “WorkerSpec”(СпециализацияРаботников)

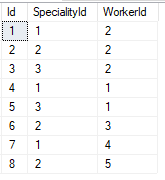


Рис.15 Таблица WorkerSpec

**3.6 Разработка интерфейса ПС**

Приложение предназначено для двух типов пользователей: администратора и клиента студии. Для входа в панель администратора необходимо ввести свои учётные данные. Для запуска приложения достаточно щелкнуть дважды левой кнопкой мыши по ярлыку.

После запуска программы пользователь приступит к работе с главной формой программы, на которой отображается каталог товаров в виде плиточного интерфейса. «EnglishSchool 1.01» обладает удобным интерфейсом, так что даже неопытному в обращение с пользовательскими компьютерными программами пользователю не составит труда быстро привыкнуть к работе с данным программным средством.

На рабочем экране расположены несколько кнопок для комфортной работы с приложением.

**Форма АИС «Школа английского языка “ABC”(подсистема работы суслугами)»**. Основная форма приложения (Рис.16). При разработке интерфейса этой формы было использовано множество различныx компонентов:

1. ComboBox – выпадающий список скидок;
2. ListBox – список услуг
3. Image – для отбражения изображения;
4. Buttons – кнопки для перехода на другие формы;
5. TextBox – поля для ввода



Button

ComboBox

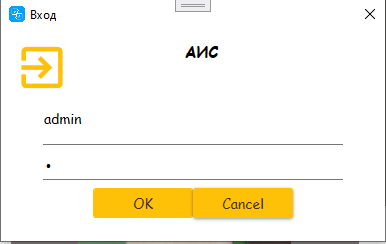
TextBox

ListBox

TextBox

Рис.16 АИС «Школа английского языка “ABC”(подсистема работы суслугами)».

**Форма входа в систему**. Интерфейс данной формы состоит из компонентов: Button – кнопки ОК, Cancel, Grid – сетка для аккуратного размещения компонентов, TextBox – поле для ввода логина, PasswordBox – поле для ввода пароля и TextBlock – текстовые метки. (Рис.17)



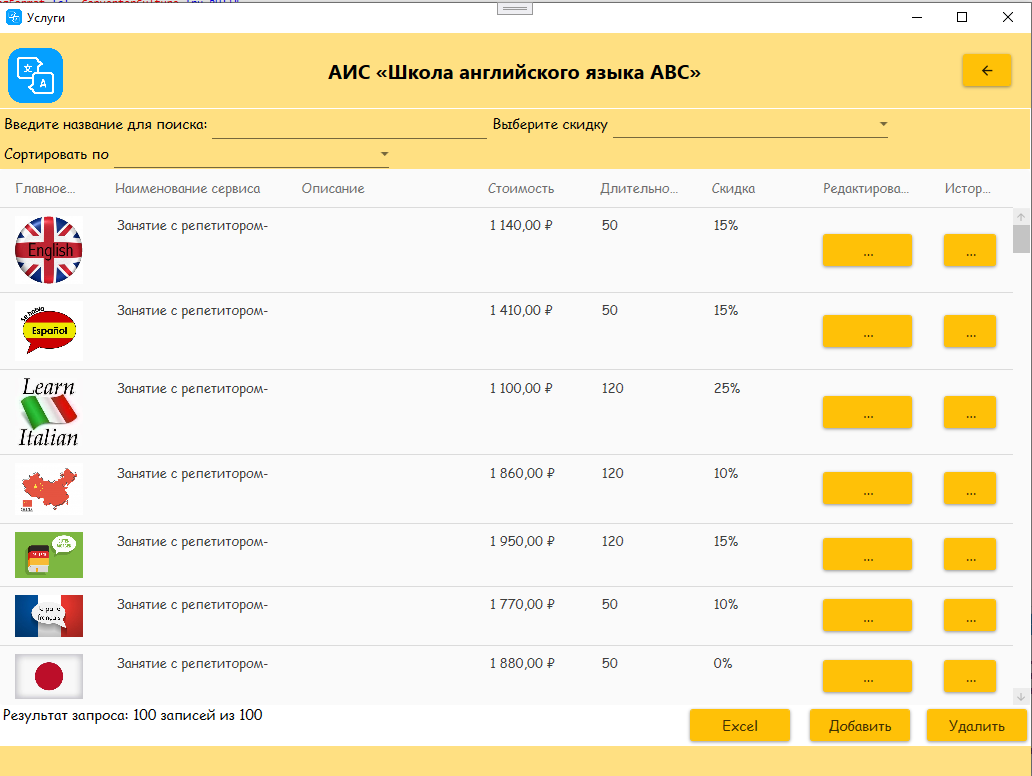
Buttons

TextBox

PasswordBox

Рис.17 Форма входа в систему

Страница **Услуги**. Интерфейс данной формы состоит из компонентов: DataGrid и двух кнопок: Сохранить и Удалить. (Рис.18)



Button – кнопки Добавить и Удалить

Button – кнопка Назад

Button – кнопки Печать в Excel

ComboBox

Компонент DataGrid - таблица

Рис.18 Страница Услуги

**Страница Добавление и редактирование**. Интерфейс данной формы состоит из компонентов: TextBox – поля ввода, ComboBox – выпадающие списки, Image – дляотображения изображения и Button. (Рис.19)



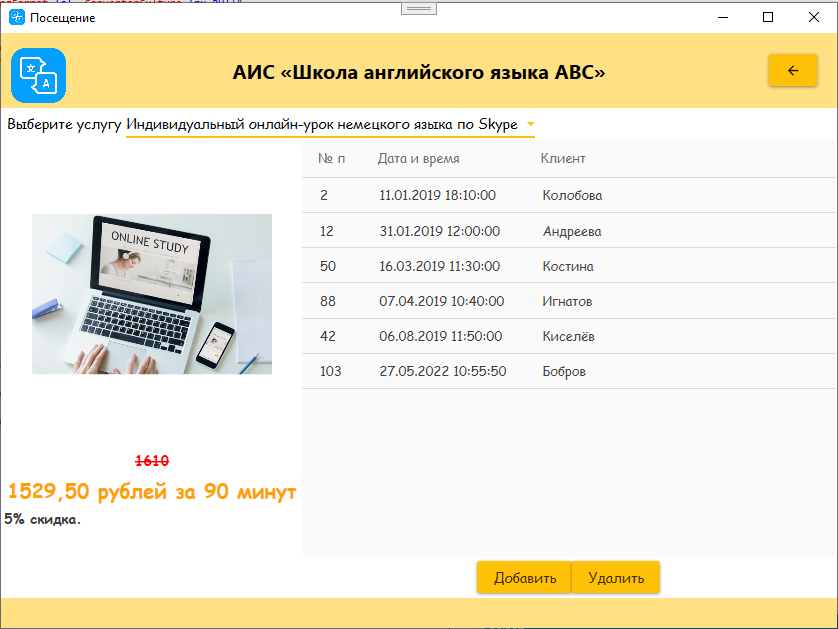
TextBox

Button

Image

Рис.19 Страница Добавление и редактирование

**Страница Продажи**. Интерфейс данной формы состоит из компонентов: DataGrid, Combobox, и кнопок: Добавить, Удалить, Назад. (Рис.20)



Button – кнопки Добавить и Удалить

Button – кнопка Назад

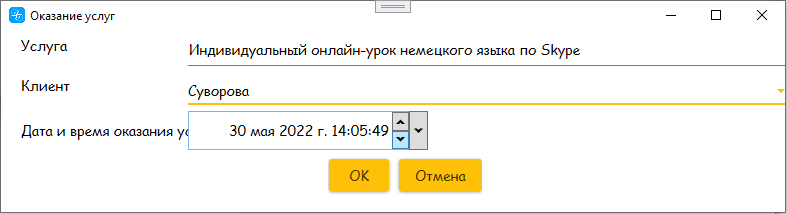
Компонент Combobox – выпадающий список

Компонент Image - изображение

Компонент DataGrid - таблица

Рис.20 Страница Продажи

**Окно добавления продаж**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода, DoubleUpDown – для ввода количества, DateTimePicker – компонент выбора даты и времени и кнопок Button. (Рис.21)



Button – кнопки Ок и Отмена

DateTimePicker

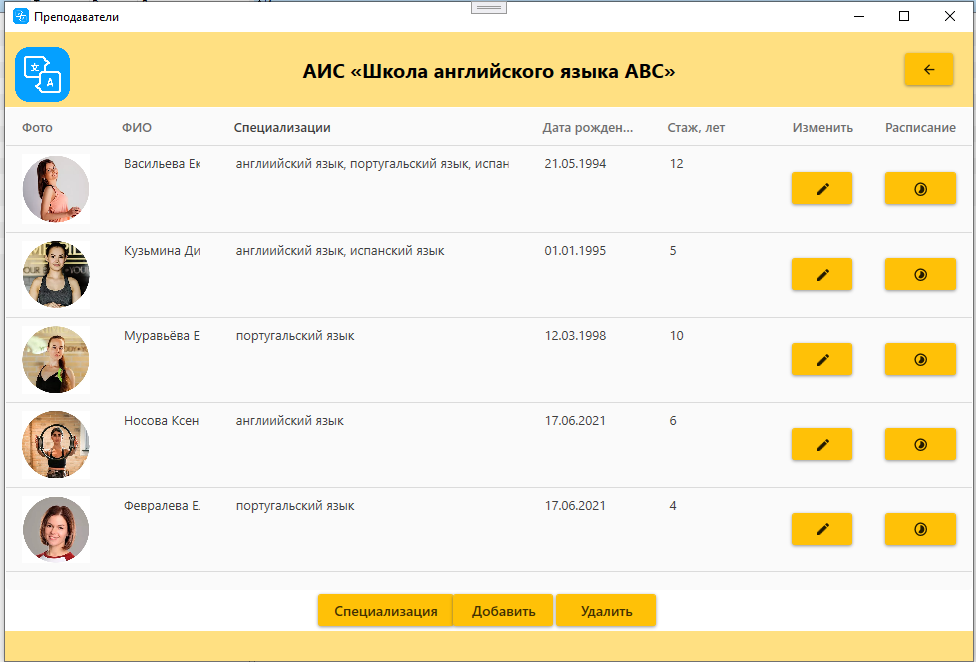
ComboBox

TextBox

Рис.21 Окно добавления

**Форма Преподаватели(**Рис.22**)**. При разработке интерфейса этой формы были использованы компоненты:

1. DataGrid – отображение списка автомастерских города в виде таблицы;
2. Image – для отображения изображения;
3. Buttons – кнопки для перехода к справочникам, добавления, редактирования и удаления записей;
4. TextBlock – текстовая метка;



Buttons

DataGrid

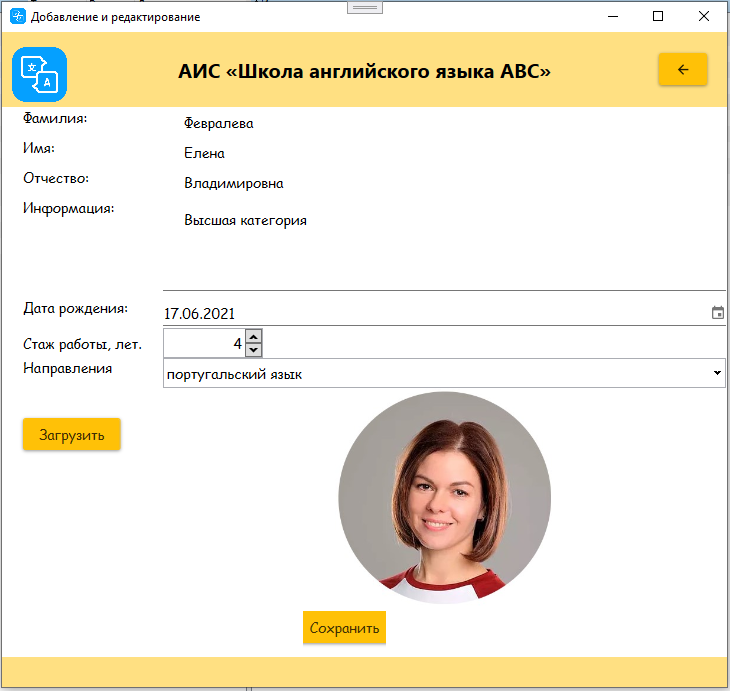
Image

TextBlock

Рис.22 Форма Преподаватели

**Страница добавления и редактирования(**Рис.23**)**. При разработке интерфейса этой формы были использованы компоненты:

1. CheckComboBox – выпадающий список;
2. Buttons – кнопки Сохранения, Загрузки и перехода назад;
3. DatePicker –поле для выбора даты;
4. IntegerUpDown –поле для целых чисел;
5. TextBox – поля для ввода текста
6. Image – для отображения изображения;
7. TextBlock – текстовые метки;



Buttons

Image

TextBlock

DatePicker

CheckComboBox

TextBox

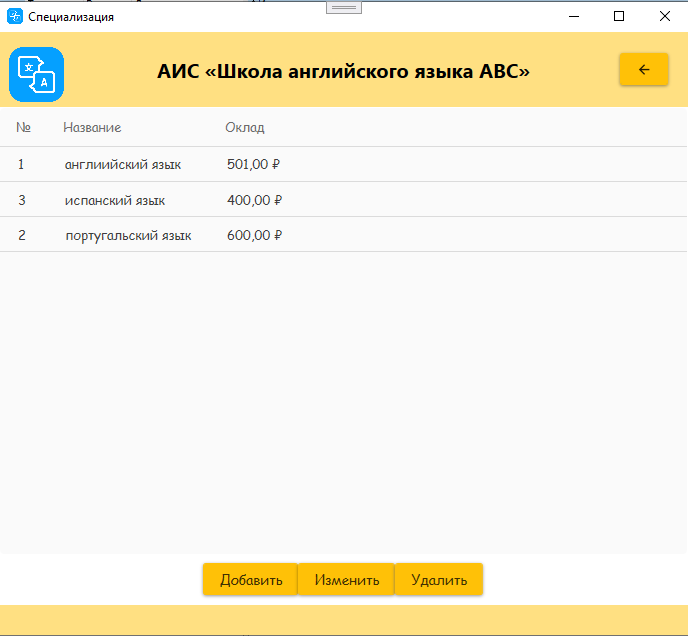
Button

DoubleUpDown

Grid

Рис.23 Страница добавления и редактирования

**Страница Направления**. Интерфейс данной формы состоит из компонентов: DataGrid и кнопок: Добавить, Изменить, Удалить, Назад. (Рис.24)



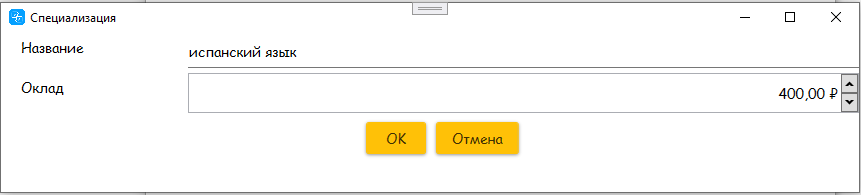
Button – кнопки Добавить, Изменить и Удалить

Button – кнопка Назад

Компонент DataGrid - таблица

Рис.24 Специализации

**Окно добавления и редактирования**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.25)

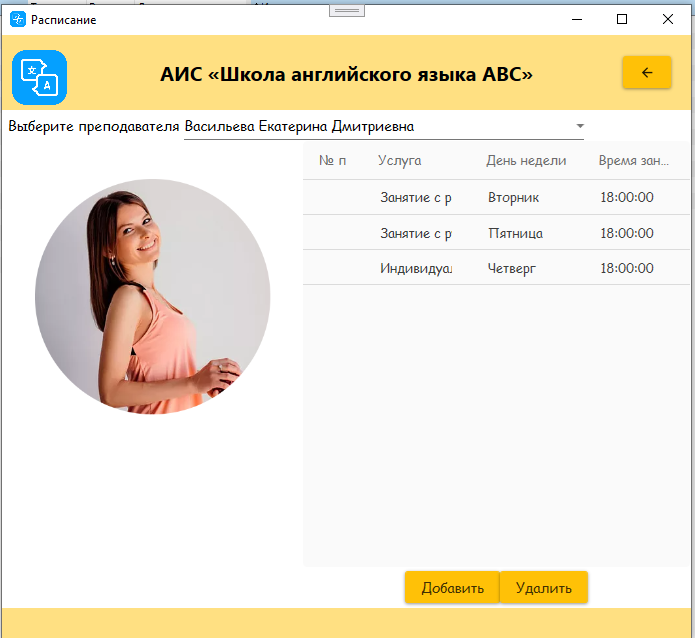


Button – кнопки Ок и Отмена

TextBox

Рис.25 Окно добавления и редактирования

**Страница Расписание**. Интерфейс данной формы состоит из компонентов: DataGrid, Combobox, и кнопок: Добавить, Изменить, Удалить, Назад. (Рис.26)



Button – кнопки Добавить, Удалить

Button – кнопка Назад

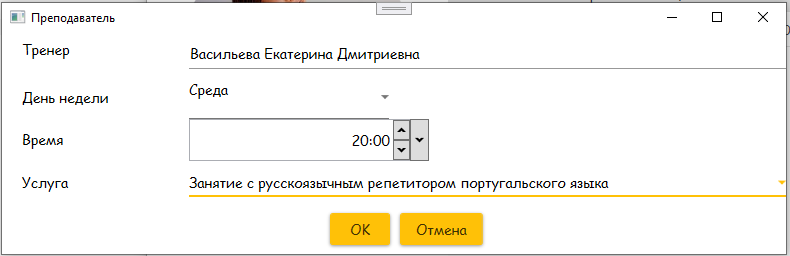
Компонент Combobox – выпадающий список

Компонент DataGrid - таблица

Компонент Image - изображение

Рис.26 Страница Расписание

**Окно добавления и редактирования**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода, TimePicker – для ввода времени, ComboBox – выпадающий список и кнопок Button. (Рис.27)



Button – кнопки Ок и Отмена

TimePicker

TextBox

ComboBox

Рис.27 Окно добавления и редактирования

**3.7 Описание структуры выходной информации**

Выходными данными являются списки услуг, отфильтрованные по разным критериям.

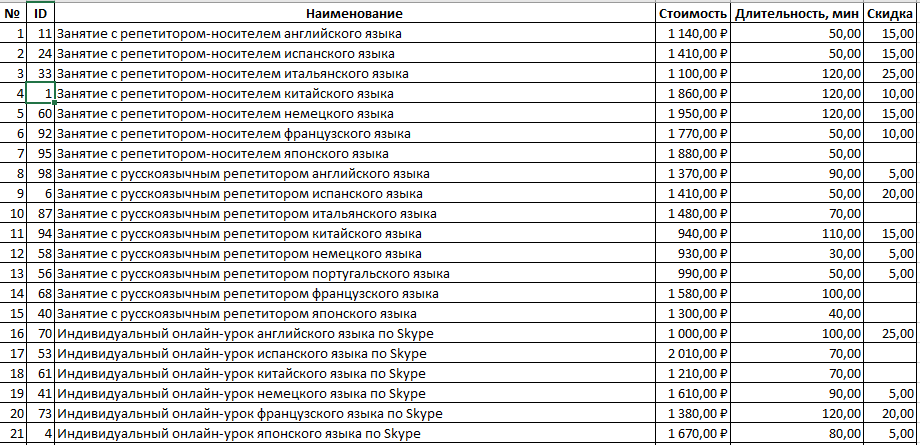


Рис.28 Список всех услуг

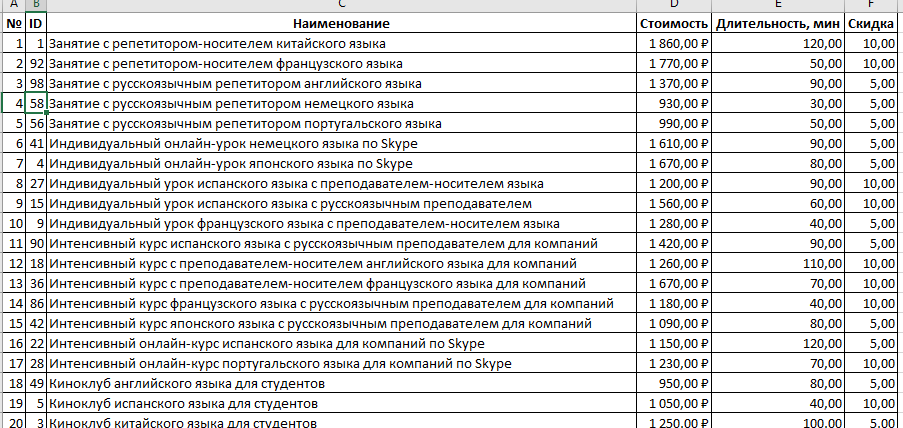


Рис.29 Список услуг

**4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**4.1 Назначение программного средства**

Назначением программного средства является автоматизация работы школы.

**4.2 Условия выполнения программного средства**

Условия, при соблюдении которых возможно выполнение работы с программным продуктом:

-Компьютер с установленной на нём «EnglishSchool 1.0».

Подготовительные действия:

-На иконке «EnglishSchool 1.01» рабочего стола произвести двойной щелчок левой кнопкой мыши.

**4.3 Эксплуатация программного средства**

Этот раздел содержит обзор приложений «EnglishSchool 1.0» и ее основных параметров.

***Активация продукта***

Активировать копию программы EnglishSchool 1.0 можно сразу после её установки, когда появится окно подтверждения подлинности Вашей версии. Ключ активации расположен внутри упаковки программного продукта. Для успешного выполнения активации ключ активации необходимо вводить именно в том виде, в котором он предоставлен.

Вводные сведения об интерфейсе пользователя

После запуска приложения отображается стартовое окно приложения (Рис.30). Главная форма представляет собой каталог услуг школы в виде списка. Каждая услуга отображается с краткой информацией: название, стоимость, скидка. Пользователь имеет возможность отфильтровать список по размеру скидки, а также отсортировать по стоимости.



Рис.30 Окно «АИС «Школа английского языка “ABC”(подсистема работы с услугами)»

Кликнув по кнопке «Закрыть» программа потребует подтверждение. (Рис.31)



Рис.31 Сообщение программы

Для входа в систему необходимо кликнуть на кнопку . Ввести логин и пароль администратора и нажать OK(Рис. 32).

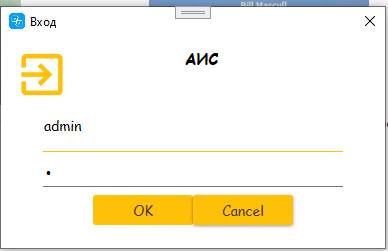


Рис.32 Форма входа в систему

Для того чтобы выйти из системы нажмите на кнопку , программа потребует потдверждения(Рис.33)

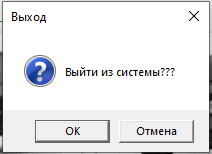


Рис.33 Сообщение программы

«Услуги»: эта страница предназначена для ввода, удаления и редактирования данных об услугах. Функционал страницы позволяет выгружать список услуг в Excel. Для каждой услуги можно открыть дополнительные страницы для редактирования записи, просмотра и редактирования сведений об истории оказания этой услуги. Пользователь имеет возможность отфильтровать список по размеру скидки, а также отсортировать по стоимости. (Рис.34)

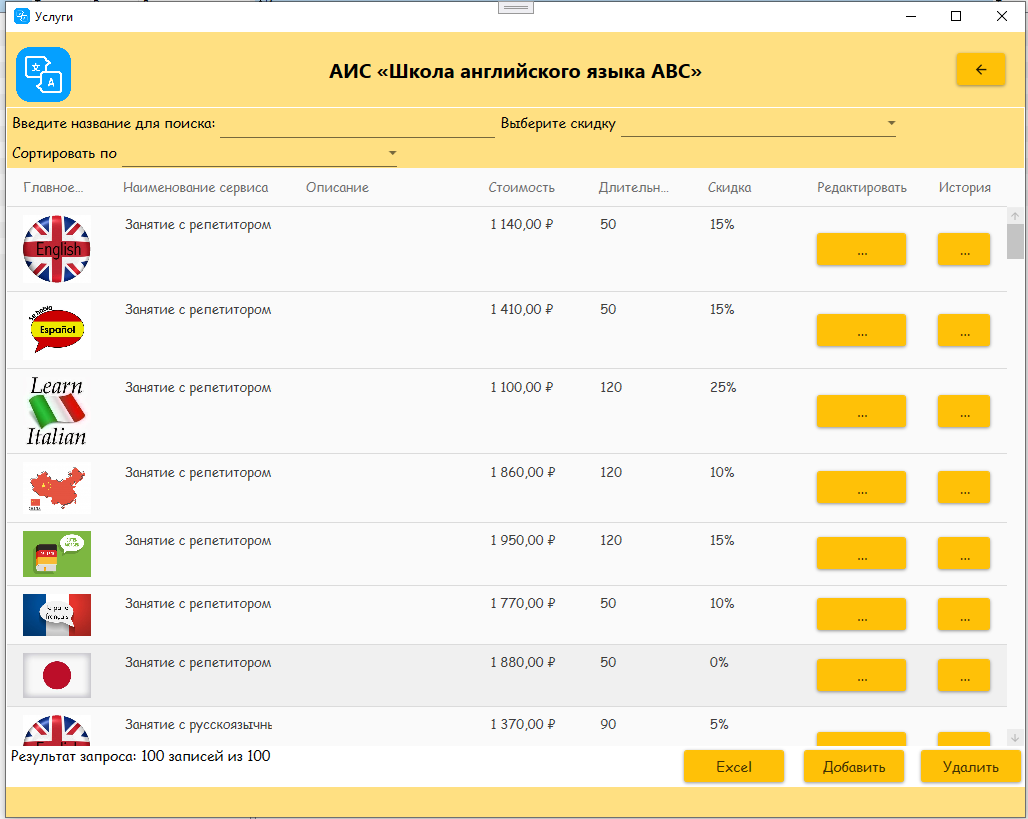


Рис.34 Страница Услуги.

«Добавление и редактирование»: эта страница предназначена для ввода, удаления и редактирования данных о товаре. (Рис.35)



Рис.35 Страница Услуги

«История оказания услуг»: эта страница предназначена для просмотра и удаления информации о продажах услуг. Добавление и редактирование осуществляется через дополнительную форму. (Рис.36)

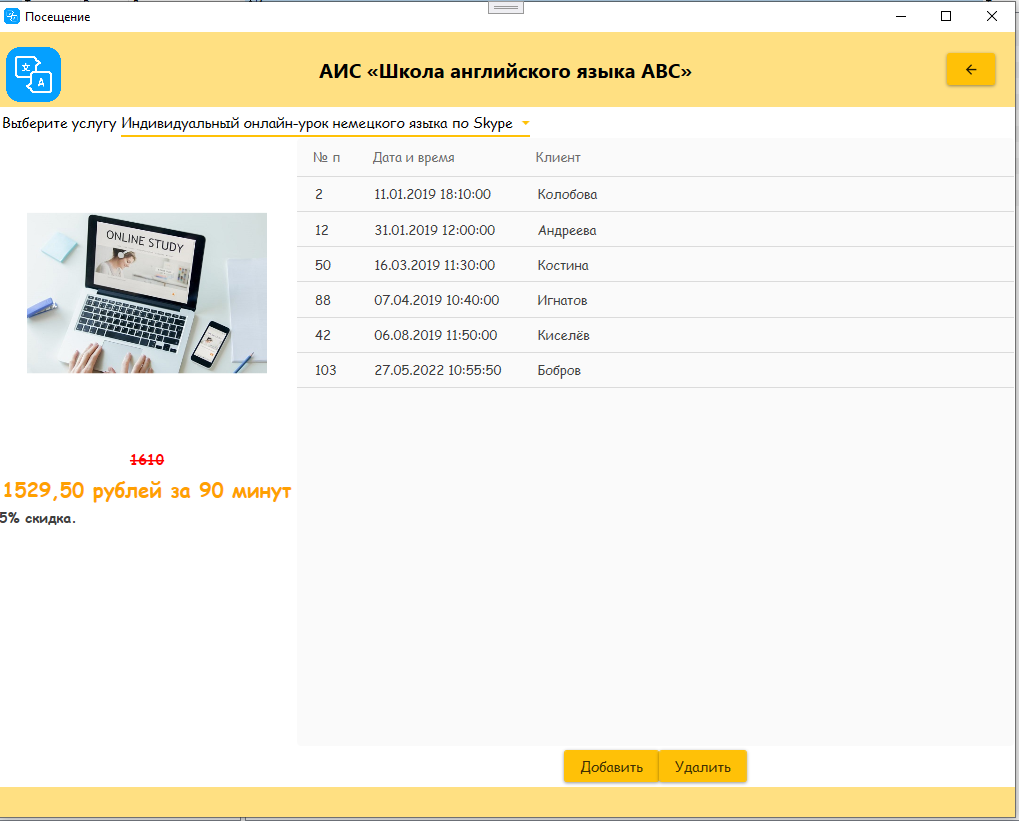


Рис.36 Страница История оказания услуг

«Оказание услуги»: эта форма предназначена для добавления и редактирования записи о продаже услуги. (Рис.37)

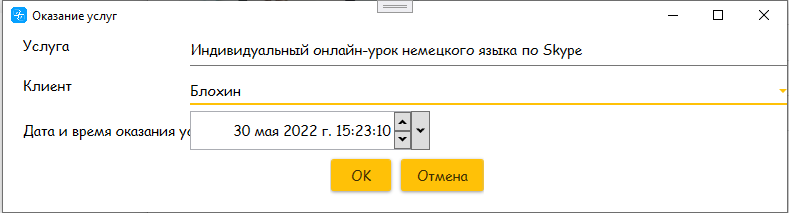


Рис.37 Форма Оказание услуги

«Специализация»: эта страница предназначена для просмотра и удаления информации о специализации преподавателей. Добавление и редактирование осуществляется через дополнительную форму. (Рис.38)

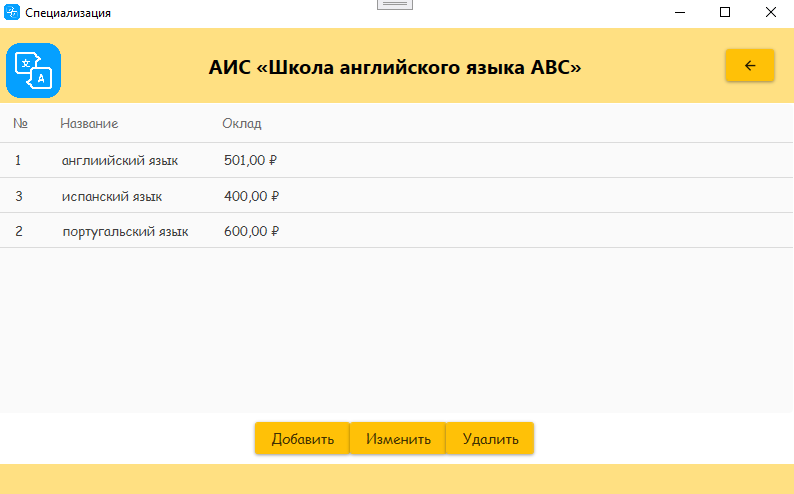


Рис.38 Страница Специализация

«Специализация»: эта форма предназначена для добавления и редактирования записи. (Рис.39)

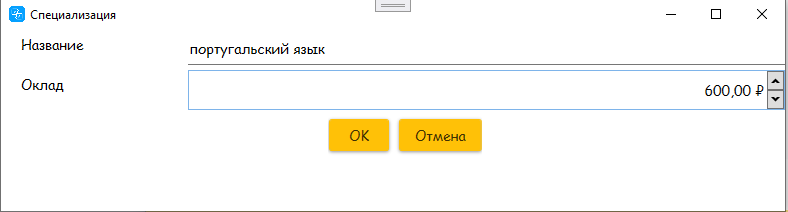


Рис.39 Форма Специализация

Страница «Преподаватели». Эта форма доступна только администратору. Для того, чтобы открыть эту форму нужно нажать на кнопку , расположенную на главной форме приложения в правом верхнем углу. Данная страница предназначена для отображения информации о преподавателях в табличном виде, открытия страницы редактирования, добавления, удаления выделенной записи, а также для перехода на другие страницы приложения. (Рис.40)

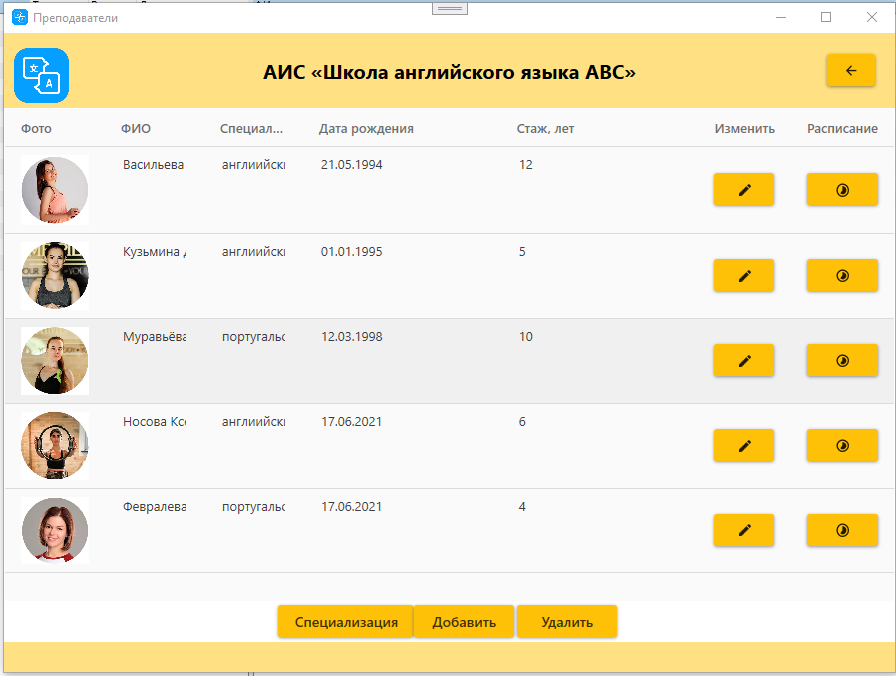


Рис.40 Страница Преподаватели

Страница Добавление и редактирование. (Рис.41) Для открытия этой страницы нужно на странице Преподаватели выбрать запись и нажать на кнопку Редактировать. Также данную страницу можно открыть в режиме добавления, нажав на соответствующую кнопку на предыдущей странице. Эта страница предназначена для добавления или редактирования сведений.

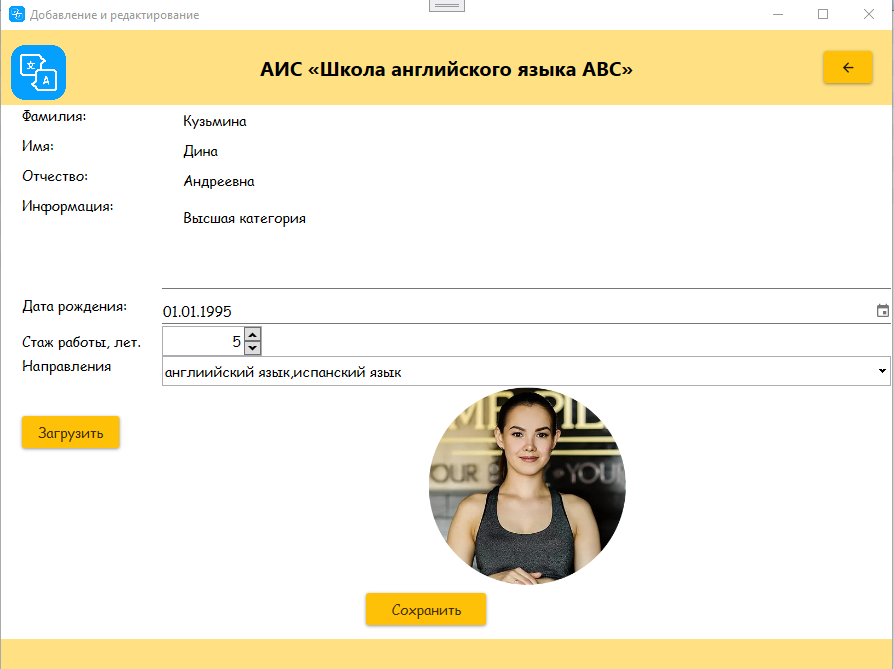


Рис.41 Страница Добавление и редактирование

«Расписание»: эта страница предназначена для просмотра и удаления информации о расписании занятий. Добавление и редактирование осуществляется через дополнительную форму. (Рис.42)

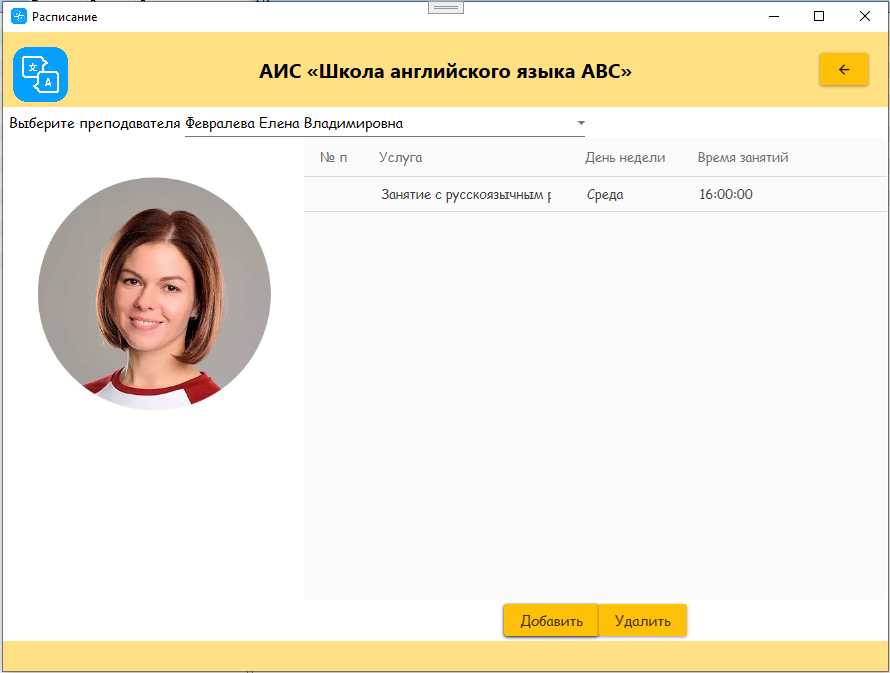


Рис.42 Страница Расписание

«Расписание»: эта форма предназначена для добавления и редактирования записи о времени проведения занятия. (Рис.43)

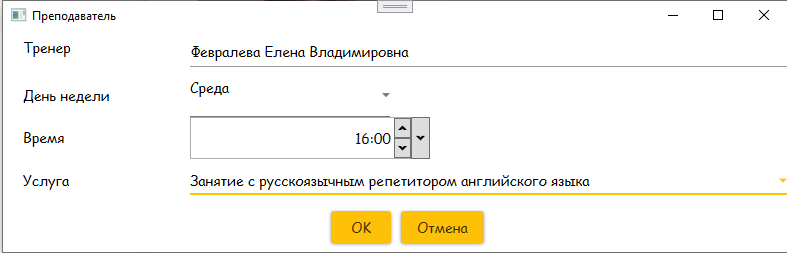


Рис.43 Форма Расписание

Работа с EnglishSchool 1.0

После авторизации в приложении Вы перейдете на главную форму программы – Каталог товаров. На страницах Услуги, Преподаватели, Расписание заполняем все необходимые для работы пункты.

После этого кликаем по кнопке «Добавить запись».

Если Вы желаете удалить любую введенную ранее информацию, Вам необходимо кликнуть по нужной кнопке.

Для того чтобы изменить запись, введите в редактируемые поля данные и нажмите кнопку сохранить. Данные формы редактированию и управлению со стороны пользователя не подлежат.

После того, как Вы закончите сеанс работы с программой, если Вы хотите выйти из EnglishSchool 1.0 кликните на соответствующей кнопке «Выход».

Основные кнопки EnglishSchool 1.0

Кнопки управления «Добавить», «Удалить запись», «Сохранить» расположены на каждой конкретной форме приложения.

Действия этих кнопок приведены в таблице 1:

Таблица 1 - Действие кнопок

|  |  |
| --- | --- |
| Добавить запись | После нажатия на эту кнопку выйдет вспомогательная форма для добавления записей |
| Удалить запись | Кнопка «Удалить запись» позволяет удалить выделенную запись. |
| Сохранить данные | Кнопка «Сохранить данные» добавляет запись введенную пользователем в БД, либо обновляет существующую. |
| Изменить | Открывает форму в режиме редактирования информации |
|  | Переход на предыдущую страницу |
|  | Перейти на страницу Расписание |
|  | Выгрузить информацию в Excel |

**4.4 Сообщения пользователю**

Для уверенной и комфортной работы пользователю важно понимать поведение программы, а программе, в свою очередь, поведение пользователя. Для информирования и уточнения намерений пользователей в программах используются сообщения.

При попытке удалить запись, пользователю выводится сообщение с запросом на подтверждение или отмену действия. (Рис.44)

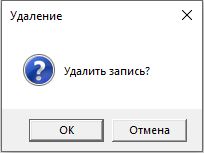


Рис.44 Сообщение Подтверждение удаления

После добавления или изменения записи, пользователю выводится сообщение. (Рис.45)

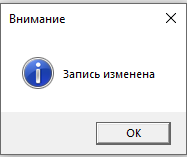


Рис.45 Сообщение Изменения сохранены

После Добавления записи, пользователю выводится сообщение. (Рис.46)

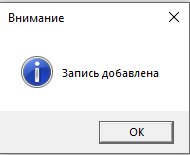


Рис.46 Сообщение Запись добавлена

При попытке удалить связанные записи, программа выводит пользователю сообщение о невозможности действия. (Рис.47)

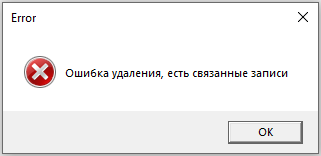


Рис.47 Сообщение Ошибка удаления

**5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА**

**5.1 Определение верхней границы количества тестов**

Для разработанного алгоритма был построен граф.

****

Рис.48 Потоковый граф

Необходимое количество тестов определяется по формуле V(G) = E-N+2, где E – это число ребер, а N–это число вершин данного графа. Согласно представленному графу E=34, N=25. Далее по формуле рассчитаем необходимое количество тестовых сценариев (тест-кейсов).

V(G)=34-25+2=11.

Для данного программного средства необходимо разработать не менее 11 тестовых сценариев.

Чтобы определить количество тестовых прогонов для программного продукта EnglishSchool 1.0 необходимо выделить требования. Из требований к приложению выделим поддержку 2-х операционных систем с одним основным языком локализации и выполнения 6-ти основных функций. Кроме того, приложение позволяет выполнять 5 функций по взаимодействию с системой (запуск приложения, выход из приложения, сохранение результатов в файл, работа с буфером и т.п.).

Таким образом, полное покрытие требований задаёт набор из 2\*1\*(6+5) =22 тестовых прогонов.

**5.2 Разработка тестовых вариантов**

Таблица 2 – Тестирование корректного добавления информации в Специализации

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Специализации | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Специализации | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = английский язык  Оклад = 500 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Специализации открыта | | Отображается форма Специализации |  |

Таблица 3 – Тестирование некорректного добавления информации в таблицу Специализации

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Специализации | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Специализации | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название =  Оклад = 500 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, так как поле пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Специализации открыта | | Отображается форма Специализации |  |

Таблица 4 – Тестирование корректного редактирования информации в таблице Специализации

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Специализации | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Специализации | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = английский язык  Оклад = 500 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Специализации  открыта | | Отображается форма Специализации |  |

Таблица 5 – Тестирование некорректного редактирования информации в таблице Специализации

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Специализации | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Специализации | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название =  Оклад = 500 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Специализации  открыта | | Отображается форма Специализации |  |

Таблица 6 – Тестирование корректного добавления информации в таблицу Преподаватель

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Преподаватель | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Преподаватель | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Фото= загружено  Фамилия = Васильева  Имя = Екатерина  Отчество = Васильевна  Дата рождения =21.05.1995  Специализация = Португальский  Стаж = 6 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Преподаватель  открыта | | Отображается форма Преподаватель |  |

Таблица 7 – Тестирование некорректного добавления информации в таблицу Преподаватель

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Преподаватель | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Преподаватель | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Фото= загружено, Фамилия = , Имя = , Отчество = Васильевна  Дата рождения =21.05.1995  Специализация =Португальский  Стаж = 6 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Преподаватель открыта | | Отображается форма Преподаватель |  |

Таблица 8 – Тестирование корректного редактирования информации в таблице Преподаватель

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Преподаватель | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Преподаватель | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные, Фото= загружено,  Фамилия = Васильева, Имя = Екатерина, Отчество = Васильевна, Дата рождения =21.05.1995, Направления = Португальский, Стаж = 6 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Преподаватель  открыта | | Отображается форма Преподаватель |  |

Таблица 9 – Тестирование некорректного редактирования информации в таблице Преподаватель

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Преподаватель | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Преподаватель | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Фото= загружено  Фамилия = , Имя =  Отчество = Васильевна  Дата рождения =21.05.1995  Специализация =Португальский  Стаж = 6 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Преподаватель  открыта | | Отображается форма Преподаватель |  |

Таблица 10 – Тестирование корректного добавления информации в таблицу Услуги

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Услуги | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуги | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Фото= загружено  Название = Урок англиийского,  стоимость = 1140 р, Длительность = 50мин, Скидка = 5 % | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуги  открыта | | Отображается форма Услуги |  |

Таблица 11 – Тестирование некорректного добавления информации в таблицу Услуги

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Услуги | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуги | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Фото= загружено  Название =,  стоимость = 1140 р, Длительность = 50мин, Скидка = 5 % | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуги  открыта | | Отображается форма Услуги |  |

Таблица 12 – Тестирование корректного редактирования информации в таблице Услуги

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Услуги | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуги | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Фото= загружено  Название =,  стоимость = 1140 р, Длительность = 50мин, Скидка = 5 % | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуги  открыта | | Отображается форма Услуги |  |

Таблица 13 – Тестирование некорректного редактирования информации в таблице Услуги

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Услуги | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Услуги | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Фото= загружено  Название = Урок англиийского,  стоимость = 1140 р, Длительность = 50мин, Скидка = 5 % | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Услуги  открыта | | Отображается форма Услуги |  |

**5.3 Составление отчета о тестировании**

Для форм проекта были созданы и проведены тесты, результаты сведены в таблицу:

Таблица 14 – Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Прохождение | | |
| Форма | Число ТП | пройдено | | не пройдено |
| Услуги | 4 | 4 | | 0 |
| Специализации | 4 | 4 | | 0 |
| Товары | 4 | 4 | | 0 |
| Продажи | 4 | 4 | | 0 |
| Преподаватель | 4 | 4 | | 0 |
| Профиль | 4 | 4 | | 0 |
| Дополнительные услуги | 4 | 4 | | 0 |
| ИТОГО | 24 | 24 | 0 | |

По результатам тестирования были выявлены и устранены следующие проблемы:

При выполнении теста таблица №8. Тестирование формы «Услуги», Некорректный запрос

**Ошибка**



Рис.51 Ошибка теста №8

**Причина**

Поле цена имеет тип вещественное число, было введено строковое значение.

**Принятые меры**

Ошибка устранена. В поле цена можно ввести только цифры и запятую

**ЗАКЛЮЧЕНИЕ**

**Оценка качества программного средства с помощью метрик**

Оценка качества программного средства проводится исходя из стандарта оценки качества ISО 9126.

На компьютере следующей конфигурации:

Установленная операционная система – Windows 10 Professional.

Процессор: Intel Pentium G2030 3.0 GHz.

ОЗУ: 4 Gb

Таблица 15 – Требования к количественным характеристикам качества программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Надежность** |  | | |
| *Завершенность:*  *-* наработка на отказ при отсутствии рестарта; | Часы | 10 | 10 |
| *Устойчивость:*  - относительные ресурсы на обеспечение надежности и рестарта. | % | 0,5 | 0,5 |
| *Восстанавливаемость:*  -длительность восстановления. | Минуты | 2 | 2 |
| *Доступность-готовность:*  -относительное время работоспособного функционирования. | Вероятность | 0,998 | 0,998 |
| **Эффективность** |  | | |
| *Временная эффективность:*  *-*время отклика - получение результатов на типовое задание;  - пропускная способность число типовых заданий, исполняемых в единицу времени. | Секунды  Число в минуту | 3  60 | 3  60 |
| *Используемость ресурсов:*  *-* относительная величина использования ресурсов ЭВМ при нормальном функционировании программного средства. |  | 0,001 | 0,001 |

Таблица 16 – Требования к качественным характеристикам программного средства.

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Практичность** |  | | |
| *Простота использования:*  **-**среднее время ввода заданий;  -среднее время отклика на задание. | Секунды  Секунды | 10  3 | 10  3 |
| *Изучаемость :*  - трудоемкость изучения применения ПС;  -продолжительность изучения;  -объем эксплуатационной документации; | Чел.-часы  Часы  Страницы | 5  2  15 | 5  2  15 |
| **Сопровождаемость** |  | | |
| *Изменяемость:*  -трудоемкость подготовки изменений;  -длительность подготовки изменений. | Чел.-часы  Часы | 5  5 | 5  5 |
| *Тестируемость:*  *-трудоемкость тестирования изменений;*  *- длительность тестирования изменений.* | Чел.-часы  Часы | 3  3 | 3  3 |
| **Мобильность** |  | | |
| *Адаптируемость:*  - трудоемкость адаптации;  -длительность адаптации. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Простота установки:*  -трудоемкость инсталляции;  -длительность инсталляции. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Замещаемость:*  -трудоемкость замены компонентов;  - длительность замены компонентов. | Чел.-часы  Часы | 0,2  0,2 | 0,2  0,2 |

Таблица 17 – Сложность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип объекта | Количество | Вес | | | Итого |
|  |  | Простой | Средний | Сложный |  |
| Экран | 17 | x1 | x2 | **x3** | =51 |
| Отчет | 3 | x2 | x5 | **x8** | =24 |

Таблица 18 – Сложность экрана

|  |  |  |  |
| --- | --- | --- | --- |
| Экраны | Количество клиентских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | Всего <8 | **Всего >8** |
|  | (<2 срв, <3 клт) | (2-3 срв, 3-5 клт) | (>3 срв, >5 клт) |
| <3 | Простой | Простой | Средний |
| 3-7 | Простой | Средний | Сложный |
| **>8** | Средний | Сложный | **Сложный** |

Таблица 19 – Сложность отчета

|  |  |  |  |
| --- | --- | --- | --- |
| Отчеты | Количество Услугиских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | Всего <8 | Всего >8 |
|  | (<2 срв, <3 клт) | (2-3 срв, 3-5 клт) | **(>3 срв, >5 клт)** |
| 0 или 1 | Простой | Простой | Средний |
| **2 или 3** | Простой | Средний | **Сложный** |
| >4 | Средний | Сложный | Сложный |

Преподавательность = 5000 тысяч строк кода/1 месяца=5000 тысяч строк кода/месяц

Качество = 25 ошибок /5 тысяч строк кода=5 ошибок/тысяч строк кода

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Г. Шилдт C#: Учебный курс. - Спб.: Питер, 2020. - 512с

2. Либерти, Д. Программирование на Си Шарп. - М.: Символ-плюс, 2018. - 684 с.

3. Ватсон К. Си Шарп/К.Ватсон.-М.:Лори, 2020.-862 с

4. Балена Ф., Димауро Д. Современная практика программирования на Visual Basic и Visual C#.-М.:Русская редакция,2020,604с.

5. Енин А., Енин Н. Локальная СУБД своими руками. Учимся на примерах.-М.:СОЛОН-ПРЕСС,2020,464с.,СД

6. Лабор В.В. Си Шарп: СоУслуги приложений для Windows/-Мн.: Харвест, 2018-384с.

7. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2020-512с.

8. ГОСТ 19.201-78.ЕСПД. Техническое задание. Требования к содержанию и оформлению

9. ГОСТ 19.102-77.ЕСПД. Стадии разработки

10. ГОСТ 19.505-79.ЕСПД. Руководство оператора. Требования к содержанию и оформлению

11. ГОСТ 19.002-80.ЕСПД. Схемы алгоритмов и программ. Правила выполнения

12. ГОСТ 19.404-79.ЕСПД. Пояснительная записка. Требования к содержанию и оформлению

13. Владимир Биллинг. Основы программирования на С#. 2018 г.

[Электронный ресурс http://mexalib.com/view/42671]

14. Никита Культин. Основы программирования в MicrosoftVisualC# 2020. 2020 г.

[Электронный ресурс http://www.litres.ru/nikita-kultin/osnovy-programmirovaniya-v-microsoft-visual-c-2010-2]

15. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2020.

[Электронный ресурс http://www.twirpx.com/file/27852]

16. Йен Гриффитс. Программирование на С#. 2021 г.

[Электронный ресурс http://rutracker.org/forum/viewtopic.php?t=4772853]

**Приложение А**

«АИС «Школа английского языка “ABC”

(подсистема работы с услугами)»

Текст кода программного средства

460.ЗМК.00005-01 12 01

**Текст кода программного средства**

**App.config**

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<configSections>

<!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />

</configSections>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

<entityFramework>

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />

<providers>

<provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />

</providers>

</entityFramework>

<connectionStrings>

<add name="AssortmentBdEntities" connectionString="metadata=res://\*/Models.AssortmentModel.csdl|res://\*/Models.AssortmentModel.ssdl|res://\*/Models.AssortmentModel.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=.\SQLEXPRESS;initial catalog=AssortmentBd;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

</connectionStrings>

</configuration>

**App.xaml**

<Application x:Class="WpfAssortmentCheck.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:WpfAssortmentCheck"

StartupUri="windows/LoginWindow.xaml">

<!--строчка StartUpUrl отвечает за запускаемое со стартом приложения окно-->

<!--Первое запускаемое окно будет главным в приложении.-->

<Application.Resources>

<!--цвета приложения для соответсвия стилю-->

<SolidColorBrush x:Key="main" Color="#FFF"/>

<SolidColorBrush x:Key="additional" Color="#FFE7FABF"/>

<SolidColorBrush x:Key="akcent" Color="#FF04A0FF"/>

<!--Стиль для окна-->

<Style TargetType="Window" x:Key="base\_window">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

<Setter Property="FontSize" Value="14"/>

<Setter Property="MinHeight" Value="600"/>

<Setter Property="MinWidth" Value="800"/>

<Setter Property="Background" Value="{StaticResource main}"/>

</Style>

<Style TargetType="Window" x:Key="base\_window\_1">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

<Setter Property="FontSize" Value="14"/>

<Setter Property="Background" Value="{StaticResource main}"/>

</Style>

<!--стиль для страницы-->

<Style TargetType="Page" x:Key="base\_page">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

<Setter Property="Background" Value="{StaticResource main}"/>

<Setter Property="FontSize" Value="14"/>

</Style>

<!--стиль для полей ввода TextBox-->

<Style TargetType="TextBox">

<Setter Property="Height" Value="30"/>

<Setter Property="VerticalAlignment" Value="Stretch"/>

<Setter Property="VerticalContentAlignment" Value="Center"/>

<Setter Property="Padding" Value="3 0"/>

<Setter Property="Background" Value="{StaticResource main}"/>

</Style>

<!--стиль для полей кнопок Button-->

<Style TargetType="Button">

<Setter Property="Width" Value="auto"/>

<Setter Property="Height" Value="auto"/>

<Setter Property="Background" Value="{StaticResource akcent}"/>

<Setter Property="Foreground" Value="{StaticResource main}"/>

<Setter Property="FontSize" Value="14"/>

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="Margin" Value="3"/>

<Setter Property="Padding" Value="5"/>

<Setter Property="Height" Value="30"/>

</Style>

<!--стиль для меток TextBlock -->

<Style TargetType="TextBlock" x:Key="base\_textblock">

<Setter Property="Height" Value="30"/>

<Setter Property="VerticalAlignment" Value="Stretch"/>

<Setter Property="HorizontalAlignment" Value="Stretch"/>

<Setter Property="Padding" Value="5"/>

<Setter Property="Background" Value="{StaticResource additional}"/>

</Style>

<!--стиль для меток TextBlock -->

<Style TargetType="TextBlock" x:Key="item\_textblock">

<Setter Property="TextAlignment" Value="Center"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="TextWrapping" Value="Wrap"/>

<Setter Property="FontSize" Value="14"/>

<Setter Property="Padding" Value="3"/>

</Style>

<!--стиль для меток ComboBoxItem -->

<Style TargetType="ComboBoxItem">

<Setter Property="Background" Value="{StaticResource main}"/>

<Setter Property="Height" Value="40"/>

</Style>

<!--стиль для меток ListViewItem -->

<Style TargetType="ListViewItem" x:Key="good\_item">

<Setter Property="Background" Value="{Binding GetColor}" />

<Setter Property="BorderBrush" Value="{StaticResource akcent}"/>

<Setter Property="BorderThickness" Value="1"/>

</Style>

</Application.Resources>

</Application>

<Window x:Class="WpfAssortmentCheck.Windows.LoginWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfAssortmentCheck.Windows"

mc:Ignorable="d"

Title="Вход" Style="{StaticResource base\_window}" Height="250"

Width="400" MinHeight="250" MinWidth="400"

WindowStartupLocation="CenterScreen"

Closing="WindowClosing"

ResizeMode="NoResize">

<!--NoResize - нужен, чтобы нельзя было изменить зааднные размеры окна-->

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition Height="40"/>

<RowDefinition Height="40"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Background="{StaticResource additional}" />

<Image Source="/WpfAssortmentCheck;component/Resources/square.png"

Margin="10"

HorizontalAlignment="Left" />

<TextBlock Grid.Row="0" Text="ИС Организация проверки ассортимента магазина"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Background="{StaticResource additional}"

FontSize="16" TextWrapping="Wrap" TextAlignment="Center"

FontWeight="Bold" Height="60" Margin="65,10,48,5" Width="281" />

<StackPanel Orientation="Horizontal" Grid.Row="1">

<TextBlock Style="{StaticResource item\_textblock}"

Grid.Row="1"

Text="Логин"

Margin="20 0"

VerticalAlignment="Center" Width="100"/>

<TextBox x:Name="TbLogin"

Grid.Column="1"

Width="200" Text="admin"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="2">

<TextBlock Style="{StaticResource item\_textblock}"

Grid.Row="2" Grid.Column="0"

Text="Пароль" Margin="20 0"

VerticalAlignment="Center" Width="100"/>

<PasswordBox x:Name="TbPass" Height="30" Width="200" Password="1" />

</StackPanel >

<StackPanel Orientation="Horizontal" Grid.Row="3" HorizontalAlignment="Center">

<Button Width="100" Height="30"

Content="OK"

x:Name="BtnOk"

Click="BtnOkClick"/>

<Button Width="100"

Height="30"

Content="Cancel"

x:Name="BtnCancel"

Click="BtnCancelClick"/>

</StackPanel>

</Grid>

</Window>

**LoginWindow.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using WpfAssortmentCheck.Models;

namespace WpfAssortmentCheck.Windows

{

/// <summary>

/// Логика взаимодействия для LoginWindow.xaml

/// </summary>

public partial class LoginWindow : Window

{

public LoginWindow()

{

InitializeComponent();

}

private void BtnOkClick(object sender, RoutedEventArgs e)

{

try

{ //загрузка всех пользователей из БД в список

List<user> users = AssortmentBdEntities.GetContext().users.ToList();

//попытка найти пользователя с указанным паролем и логином

//если такого пользователя не будет обнаружено то переменная u будет равна null

user u = users.FirstOrDefault(p => p.password == TbPass.Password && p.username == TbLogin.Text);

if (u != null)

{

// логин и пароль корректные запускаем главную форму приложения

MainWindow mainWindow = new MainWindow();

mainWindow.Owner = this;

this.Hide();

mainWindow.Show();

}

else

{

MessageBox.Show("Не верный логин или пароль");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

//код кнопки Cancel

private void BtnCancelClick(object sender, RoutedEventArgs e)

{

this.Close();

}

// попытка закрыть приложение

private void WindowClosing(object sender, System.ComponentModel.CancelEventArgs e)

{

// на экране отображается форма с двумя кнопками

MessageBoxResult x = MessageBox.Show("Вы действительно хотите закрыть приложение?",

"Выйти", MessageBoxButton.OKCancel, MessageBoxImage.Question);

if (x == MessageBoxResult.Cancel)

e.Cancel = true;

}

}

}

**MainWindow.xaml**

<Window x:Class="EnglishSchoolApp.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:EnglishSchoolApp"

mc:Ignorable="d"

Style="{DynamicResource base\_window}"

WindowStyle="None"

Title="Фирма Услуги" Icon="Resourses/wind.ico" WindowStartupLocation="CenterScreen" Closed="Window\_Closed"

>

<Grid>

<Frame x:Name="MainFrame" NavigationUIVisibility="Hidden" />

</Grid>

</Window>

**MainWindow.cs**

<Window x:Class="WpfAssortmentCheck.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfAssortmentCheck"

mc:Ignorable="d"

Title="{Binding ElementName=MainFrame, Path=Content.Title}"

Style="{StaticResource base\_window}"

Height="450" Width="800" Icon="/WpfAssortmentCheck;component/Resources/square.ico"

Closing="WindowClosing" WindowStartupLocation="CenterScreen"

Closed="WindowClosed" >

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

<Grid Background="{StaticResource additional}"/>

<Image Source="/WpfAssortmentCheck;component/Resources/square.png"

Margin="10"

HorizontalAlignment="Left"/>

<TextBlock Grid.Row="0" Text="ИС Организация проверки ассортимента магазина"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Background="{StaticResource additional}"

FontSize="20"

FontWeight="Bold"/>

<Frame x:Name="MainFrame" Grid.Row="1" BorderBrush="{StaticResource akcent}"

NavigationUIVisibility="Hidden"

BorderThickness="1" ContentRendered="MainFrameContentRendered"/>

<StackPanel Grid.Row="0" Orientation="Horizontal" HorizontalAlignment="Right" >

<Button x:Name="BtnEditGoods" Content="Услуги" Click="BtnEditGoodsClick"/>

<Button x:Name="BtnEditGroups" Content="Специализации" Click="BtnEditGroups\_Click"/>

<Button x:Name="BtnEditDev" Content="Поставщики" Click="BtnEditDev\_Click"/>

<Button x:Name="BtnBack" Content="Назад" Click="BtnBackClick"/>

</StackPanel>

<Grid Grid.Row="2" Background="{StaticResource additional}"/>

</Grid>

</Window>

MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfAssortmentCheck.Models;

using WpfAssortmentCheck.Pages;

namespace WpfAssortmentCheck

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

MainFrame.Navigate(new CatalogOfGoods());

Manager.MainFrame = MainFrame;

}

private void WindowClosed(object sender, EventArgs e)

{

// показать владельца формы

Owner.Show();

// или если надо, то можно закрыть приложение командой

// App.Current.Shutdown();

}

//событие попытки закрытия окна,

// если пользователь выберет Cancel, то форму не закроем

private void WindowClosing(object sender, System.ComponentModel.CancelEventArgs e)

{

MessageBoxResult x = MessageBox.Show("Вы действительно хотите выйти?",

"Выйти", MessageBoxButton.OKCancel, MessageBoxImage.Question);

if (x == MessageBoxResult.Cancel)

e.Cancel = true;

}

// Кнопка назад

private void BtnBackClick(object sender, RoutedEventArgs e)

{

Manager.MainFrame.GoBack();

}

// Кнопка навигации

private void BtnEditGoodsClick(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new GoodsPage());

}

// Событие отрисовки страницы

// Скрываем или показываем кнопку Назад

// Скрываем или показываем кнопки Для перехода к остальным страницам

private void MainFrameContentRendered(object sender, EventArgs e)

{

if (MainFrame.CanGoBack)

{

BtnBack.Visibility = Visibility.Visible;

BtnEditGoods.Visibility = Visibility.Collapsed;

BtnEditDev.Visibility = Visibility.Collapsed;

BtnEditGroups.Visibility = Visibility.Collapsed;

}

else

{

BtnBack.Visibility = Visibility.Collapsed;

BtnEditGoods.Visibility = Visibility.Visible;

BtnEditDev.Visibility = Visibility.Visible;

BtnEditGroups.Visibility = Visibility.Visible;

}

}

private void BtnEditDev\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new DeveloperPage());

}

private void BtnEditGroups\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new CategoryPage());

}

}

}

**CatalogOfGoods.xaml**

<Page x:Class="WpfAssortmentCheck.Pages.CatalogOfGoods"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:WpfAssortmentCheck.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Каталог"

IsVisibleChanged="PageIsVisibleChanged"

Style="{StaticResource base\_page}">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

<WrapPanel Grid.Row="0"

HorizontalAlignment="Stretch"

Background="{StaticResource additional}">

<StackPanel Orientation="Horizontal">

<TextBlock Text="Введите название для поиска:"

Style="{StaticResource base\_textblock}"/>

<TextBox Width="275" x:Name="TBoxSearch"

TextChanged="TBoxSearchTextChanged"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Выберите Группу товаров"

Style="{StaticResource base\_textblock}" />

<ComboBox Width="275" x:Name="ComboCategory"

SelectionChanged="ComboTypeSelectionChanged"

DisplayMemberPath="Name"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Сортировать по"

Style="{StaticResource base\_textblock}" />

<ComboBox Width="275"

x:Name="ComboSort"

SelectionChanged="ComboSortSelectionChanged" >

<ComboBoxItem Content="По возрастанию"/>

<ComboBoxItem Content="По убыванию"/>

</ComboBox>

</StackPanel>

</WrapPanel>

<ListView x:Name="LViewGoods" Grid.Row="1"

ScrollViewer.HorizontalScrollBarVisibility="Disabled"

ItemContainerStyle="{StaticResource good\_item}">

<ListView.ItemsPanel >

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal" HorizontalAlignment="Center"/>

</ItemsPanelTemplate>

</ListView.ItemsPanel>

<ListView.ItemTemplate>

<DataTemplate>

<Grid Margin="20" Width="300">

<Grid.RowDefinitions>

<RowDefinition Height="310" />

<RowDefinition Height="auto" />

<RowDefinition Height="20" />

</Grid.RowDefinitions>

<Image Width="240" Grid.Row="0" Stretch="Uniform"

HorizontalAlignment="Center" Margin="5"

Source="{Binding Path=GetPhoto}"/>

<StackPanel Grid.Row="1" Height="100">

<TextBlock Padding="3" Height="Auto"

Style="{StaticResource item\_textblock}" Text="{Binding GoodName}"/>

<TextBlock Text="{Binding Price, StringFormat='c',

ConverterCulture='ru-RU'}" Height="Auto"

Style="{StaticResource item\_textblock}"

VerticalAlignment="Center"

Foreground="{StaticResource akcent}"

FontWeight="Bold" FontSize="20"/>

<TextBlock Text="{Binding GetCount}" Height="Auto"

Style="{StaticResource item\_textblock}"

VerticalAlignment="Center"

Foreground="{StaticResource akcent}"

FontWeight="Bold" FontSize="10"/>

</StackPanel>

<TextBlock Text="{Binding GetStatus}" Height="Auto"

Style="{StaticResource item\_textblock}" Grid.Row="2"/>

</Grid>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<TextBlock Grid.Row="2" x:Name="TextBlockCount" />

</Grid>

</Page>

**CatalogOfGoods.xaml.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfAssortmentCheck.Models;

namespace WpfAssortmentCheck.Pages

{

/// <summary>

/// Логика взаимодействия для CatalogOfGoods.xaml

/// </summary>

public partial class CatalogOfGoods : Page

{

int \_itemcount = 0;

public CatalogOfGoods()

{

InitializeComponent();

// загрузка данных в combobox + добавление дополнительной строки

var categories = AssortmentBdEntities.GetContext().Categories.OrderBy(p => p.Name).ToList();

categories.Insert(0, new Category

{

Name = "Все типы"

}

);

ComboCategory.ItemsSource = categories;

ComboCategory.SelectedIndex = 0;

// загрузка данных в listview сортируем по названию

LViewGoods.ItemsSource = AssortmentBdEntities.GetContext().Goods.OrderBy(p => p.GoodName).ToList();

\_itemcount = LViewGoods.Items.Count;

// отображение количества записей

TextBlockCount.Text = $" Результат запроса: {\_itemcount} записей из {\_itemcount}";

}

private void PageIsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

//обновление данных после каждой активации окна

if (Visibility == Visibility.Visible)

{

AssortmentBdEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

LViewGoods.ItemsSource = AssortmentBdEntities.GetContext().Goods.OrderBy(p => p.GoodName).ToList();

}

}

// Поиск товаров, которые содержат данную поисковую строку

private void TBoxSearchTextChanged(object sender, TextChangedEventArgs e)

{

UpdateData();

}

// Поиск товаров конкретного производителя

private void ComboTypeSelectionChanged(object sender, SelectionChangedEventArgs e)

{

UpdateData();

}

/// <summary>

/// Метод для фильтрации и сортировки данных

/// </summary>

private void UpdateData()

{

// получаем текущие данные из бд

var currentGoods = AssortmentBdEntities.GetContext().Goods.OrderBy(p => p.GoodName).ToList();

// выбор только тех товаров, которые принадлежат данному производителю

if (ComboCategory.SelectedIndex > 0)

currentGoods = currentGoods.Where(p => p.CategoryId == (ComboCategory.SelectedItem as Category).CategoryId).ToList();

// выбор тех товаров, в названии которых есть поисковая строка

currentGoods = currentGoods.Where(p => p.GoodName.ToLower().Contains(TBoxSearch.Text.ToLower())).ToList();

// сортировка

if (ComboSort.SelectedIndex >= 0)

{

// сортировка по возрастанию цены

if (ComboSort.SelectedIndex == 0)

currentGoods = currentGoods.OrderBy(p => p.Price).ToList();

// сортировка по убыванию цены

if (ComboSort.SelectedIndex == 1)

currentGoods = currentGoods.OrderByDescending(p => p.Price).ToList();

}

// В качестве источника данных присваиваем список данных

LViewGoods.ItemsSource = currentGoods;

// отображение количества записей

TextBlockCount.Text = $" Результат запроса: {currentGoods.Count} записей из {\_itemcount}";

}

// сортировка товаров

private void ComboSortSelectionChanged(object sender, SelectionChangedEventArgs e)

{

UpdateData();

}

}

}

**GoodsPage.xaml**

<Page x:Class="WpfAssortmentCheck.Pages.GoodsPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:WpfAssortmentCheck.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Услуги" IsVisibleChanged="PageIsVisibleChanged" Style="{StaticResource base\_page}">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="40"/>

</Grid.RowDefinitions>

<DataGrid x:Name="DataGridGood" Grid.Row="0"

AutoGenerateColumns="False"

IsReadOnly="True"

RowHeight="30"

SelectionMode="Single"

LoadingRow="DataGridGoodLoadingRow">

<DataGrid.Columns>

<DataGridTemplateColumn Width="100" Header="Главное изображение">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Image Source="{Binding Path=GetPhoto}" />

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTextColumn Binding="{Binding GoodName}" Header="Наименование услуги" Width="10\*"/>

<DataGridTextColumn Binding="{Binding Price, StringFormat='c', ConverterCulture='ru-RU'}" Header="Стоимость" Width="\*"/>

<DataGridTextColumn Binding="{Binding Weight}" Header="Вес" Width="\*"/>

<DataGridTextColumn Binding="{Binding Length}" Header="Длина" Width="\*"/>

<DataGridTextColumn Binding="{Binding Width}" Header="Ширина" Width="\*"/>

<DataGridTextColumn Binding="{Binding Heigth}" Header="Высота" Width="\*"/>

<DataGridTextColumn Binding="{Binding GetStatus}" Header="Есть в продаже" Width="\*"/>

<DataGridTextColumn Binding="{Binding GetCount}" Header="Количество на складе" Width="\*"/>

<DataGridTemplateColumn Width="auto" Header="Изменить">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnEdit" Click="ButtonClick" Margin="0">

<Image Source="/WpfAssortmentCheck;component/Resources/edit.png" />

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="auto" Header="Продажи">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnSells" Click="BtnSells\_Click" Margin="0">

<Image Source="/WpfAssortmentCheck;component/Resources/zonemoney.png" />

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="auto" Header="Приёмка">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnAcceptance" Click="BtnAcceptance\_Click" Margin="0">

<Image Source="/WpfAssortmentCheck;component/Resources/arrow (2).png" />

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="auto" Header="Преподаватель">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnWriteOff" Click="BtnWriteOff\_Click" Margin="0">

<Image Source="/WpfAssortmentCheck;component/Resources/woodenboxminus.png" />

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="auto" Header="Услуги">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnReturns" Click="BtnReturns\_Click" Margin="0">

<Image Source="/WpfAssortmentCheck;component/Resources/arrow (4).png" />

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="auto" Header="Ценник">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="BtnTag" Click="BtnTag\_Click" Margin="0">

<Image Source="/WpfAssortmentCheck;component/Resources/tag.png" />

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel Orientation="Horizontal" Grid.Row="1" >

<Button x:Name="BtnExcel" Width="130" HorizontalAlignment="Left" Margin="3"

Content="Печать в Excel" Click="BtnExcel\_Click"/>

<Button x:Name="BtnLoadFromExcel" Width="auto" HorizontalAlignment="Left"

Content="Загрузить из Excel" Click="BtnLoadFromExcel\_Click"/>

<Button x:Name="BtnPrintAllPriceTag" Width="auto" HorizontalAlignment="Left"

Content="Печать всех ценников" Click="BtnPrintAllPriceTag\_Click"/>

</StackPanel>

<Button x:Name="BtnAdd" Grid.Row="1" Width="100" HorizontalAlignment="Right"

Content="Добавить" Click="BtnAddClick" Margin="0 0 110 0"/>

<Button x:Name="BtnDelete" Grid.Row="1" Width="100" HorizontalAlignment="Right" Margin="3"

Content="Удалить" Click="BtnDeleteClick"/>

</Grid>

</Page>

**GoodsPage.xaml.cs**

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.OleDb;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfAssortmentCheck.Models;

using WpfAssortmentCheck.Pages;

using Excel = Microsoft.Office.Interop.Excel;

namespace WpfAssortmentCheck.Pages

{

/// <summary>

/// Логика взаимодействия для GoodsPage.xaml

/// </summary>

public partial class GoodsPage : Page

{

List<Good> goods;

public GoodsPage()

{

InitializeComponent();

}

private void ButtonClick(object sender, RoutedEventArgs e)

{

// открытие редактирования услуги

// передача выбранного услуги в AddGoodPage

Manager.MainFrame.Navigate(new AddGoodPage((sender as Button).DataContext as Good));

}

private void PageIsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

//событие отображения данного Page

// обновляем данные каждый раз когда активируется этот Page

if (Visibility == Visibility.Visible)

{

DataGridGood.ItemsSource = null;

//загрузка обновленных данных

AssortmentBdEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

goods = AssortmentBdEntities.GetContext().Goods.OrderBy(p => p.GoodName).ToList();

DataGridGood.ItemsSource = goods;

}

}

private void BtnAddClick(object sender, RoutedEventArgs e)

{

// открытие AddGoodPage для добавления новой записи

Manager.MainFrame.Navigate(new AddGoodPage(null));

}

private void BtnDeleteClick(object sender, RoutedEventArgs e)

{

// удаление выбранного услуги из таблицы

//получаем все выделенные услуги

var selectedGoods = DataGridGood.SelectedItems.Cast<Good>().ToList();

// вывод сообщения с вопросом Удалить запись?

MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить {selectedGoods.Count()} записей???",

"Удаление", MessageBoxButton.OKCancel, MessageBoxImage.Question);

//если пользователь нажал ОК пытаемся удалить запись

if (messageBoxResult == MessageBoxResult.OK)

{

try

{

// берем из списка удаляемых товаров один элемент

Good x = selectedGoods[0];

// проверка, есть ли у услуги в таблице о продажах связанные записи

// если да, то выбрасывается исключение и удаление прерывается

if ((x.Acceptances.Count > 0) || (x.WriteOffs.Count>0) ||(x.Stores.Count>0) ||(x.Sells.Count > 0) || (x.GoodReturns.Count > 0))

throw new Exception("Ошибка удаления, есть связанные записи");

// удаляем услуги

AssortmentBdEntities.GetContext().Goods.Remove(x);

//сохраняем изменения

AssortmentBdEntities.GetContext().SaveChanges();

MessageBox.Show("Записи удалены");

goods.Clear();

goods = AssortmentBdEntities.GetContext().Goods.OrderBy(p => p.GoodName).ToList();

DataGridGood.ItemsSource = null;

DataGridGood.ItemsSource = goods;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString(), "Ошибка удаления", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void BtnSellClick(object sender, RoutedEventArgs e)

{

// открытие страницы о продажах SellGoodsPage

// передача в него выбранного услуги

//Manager.MainFrame.Navigate(new SellGoodsPage((sender as Button).DataContext as Good));

}

// отображение номеров строк в DataGrid

private void DataGridGoodLoadingRow(object sender, DataGridRowEventArgs e)

{

e.Row.Header = (e.Row.GetIndex() + 1).ToString();

}

private void PrintExcel()

{

string fileName = AppDomain.CurrentDomain.BaseDirectory + "\\" + "Goods" + ".xltx";

Excel.Application xlApp = new Excel.Application();

Excel.Worksheet xlSheet = new Excel.Worksheet();

try

{

//добавляем книгу

xlApp.Workbooks.Open(fileName, Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

//делаем временно неактивным документ

xlApp.Interactive = false;

xlApp.EnableEvents = false;

Excel.Range xlSheetRange;

//выбираем лист на котором будем работать (Лист 1)

xlSheet = (Excel.Worksheet)xlApp.Sheets[1];

//Название листа

xlSheet.Name = "Список товаров";

int row = 2;

int i = 0;

foreach (Good good in goods)

{

xlSheet.Cells[row, 1] = (i + 1).ToString();

string s;

// DateTime y = Convert.ToDateTime(dtOrders.Rows[i].Cells[1].Value);

xlSheet.Cells[row, 2] = good.GoodId.ToString();

s = "";

if (good.CategoryId != null) s = good.Category.Name.ToString();

xlSheet.Cells[row, 3] = s;

s = "";

xlSheet.Cells[row, 4] = good.GoodName.ToString();

xlSheet.Cells[row, 5] = good.Price.ToString();

s = "";

if (good.Weight != null) s = good.Weight.ToString();

xlSheet.Cells[row, 6] = s;

s = "";

if (good.Width != null) s = good.Width.ToString();

xlSheet.Cells[row, 7] = s;

s = "";

if (good.Length != null) s = good.Length.ToString();

xlSheet.Cells[row, 8] = s;

s = "";

if (good.Heigth != null) s = good.Heigth.ToString();

xlSheet.Cells[row, 9] = s;

if (good.Artikul != null) s = good.Artikul.ToString();

xlSheet.Cells[row, 10] = s;

xlSheet.Cells[row, 11] = good.GetStatus;

row++;

Excel.Range r = xlSheet.get\_Range("A" + row.ToString(), "K" + row.ToString());

r.Insert(Excel.XlInsertShiftDirection.xlShiftDown);

i++;

}

row--;

xlSheetRange = xlSheet.get\_Range("A2:K" + (row + 1).ToString(), Type.Missing);

xlSheetRange.Borders.LineStyle = true;

//xlSheet.Cells[row + 1, 9] = "=SUM(I2:I" + row.ToString() + ")";

//xlSheet.Cells[row + 1, 8] = "ИТОГО:";

row++;

//выбираем всю область данных\*/

xlSheetRange = xlSheet.UsedRange;

//выравниваем строки и колонки по их содержимому

xlSheetRange.Columns.AutoFit();

xlSheetRange.Rows.AutoFit();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

finally

{

//Показываем ексель

xlApp.Visible = true;

xlApp.Interactive = true;

xlApp.ScreenUpdating = true;

xlApp.UserControl = true;

}

}

private void BtnExcel\_Click(object sender, RoutedEventArgs e)

{

PrintExcel();

}

private void BtnLoadFromExcel\_Click(object sender, RoutedEventArgs e)

{

string filePath = string.Empty;

string fileExt = string.Empty;

OpenFileDialog file = new OpenFileDialog(); //open dialog to choose file

if (file.ShowDialog() == true) //if there is a file choosen by the user

{

filePath = file.FileName; //get the path of the file

fileExt = System.IO.Path.GetExtension(file.FileName); //get the file extension

MessageBox.Show("+");

if (fileExt.CompareTo(".xls") == 0 || fileExt.CompareTo(".xlsx") == 0)

{

try

{

DataTable dtExcel = new DataTable();

dtExcel = ReadExcel(filePath, fileExt); //read excel file

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

else

{

MessageBox.Show("Please choose .xls or .xlsx file only.", "Warning", MessageBoxButton.OK, MessageBoxImage.Error); //custom messageBox to show error

}

}

}

public DataTable ReadExcel(string fileName, string fileExt)

{

string conn = string.Empty;

DataTable dtexcel = new DataTable();

if (fileExt.CompareTo(".xls") == 0)

conn = @"provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + fileName + ";Extended Properties='Excel 8.0;HRD=Yes;IMEX=1';"; //for below excel 2007

else

conn = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + fileName + ";Extended Properties='Excel 12.0;HDR=NO';"; //for above excel 2007

using (OleDbConnection con = new OleDbConnection(conn))

{

try

{

OleDbDataAdapter oleAdpt = new OleDbDataAdapter("select \* from [Лист1$]", con); //here we read data from sheet1

oleAdpt.Fill(dtexcel); //fill excel data into dataTable

MessageBox.Show(dtexcel.Rows.Count.ToString());

List<Good> addgoods = new List<Good>();

for (int j = 1; j < dtexcel.Rows.Count; j++)

{

Good x = new Good();

x.CategoryId = Convert.ToInt32(dtexcel.Rows[j].ItemArray[1]);

x.GoodName = dtexcel.Rows[j].ItemArray[2].ToString();

x.MainPhoto = dtexcel.Rows[j].ItemArray[3].ToString();

x.Price = dtexcel.Rows[j].ItemArray[4] == DBNull.Value ? 0 : Convert.ToDouble(dtexcel.Rows[j].ItemArray[4]);

if (dtexcel.Rows[j].ItemArray[5] == DBNull.Value)

{

x.Weight = null;

}

else

{

x.Weight = Convert.ToDouble(dtexcel.Rows[j].ItemArray[5]);

}

if (dtexcel.Rows[j].ItemArray[6] == DBNull.Value)

{

x.Width = null;

}

else

{

x.Width = Convert.ToDouble(dtexcel.Rows[j].ItemArray[6]);

}

if (dtexcel.Rows[j].ItemArray[7] == DBNull.Value)

{

x.Length = null;

}

else

{

x.Length = Convert.ToDouble(dtexcel.Rows[j].ItemArray[7]);

}

if (dtexcel.Rows[j].ItemArray[8] == DBNull.Value)

{

x.Heigth = null;

}

else

{

x.Heigth = Convert.ToDouble(dtexcel.Rows[j].ItemArray[8]);

}

x.Artikul = Convert.ToString(dtexcel.Rows[j].ItemArray[9]);

x.Active = Convert.ToBoolean(Convert.ToInt32(dtexcel.Rows[j].ItemArray[10]));

addgoods.Add(x);

MessageBox.Show(x.GoodName.ToString() + x.CategoryId.ToString());

}

AssortmentBdEntities.GetContext().Goods.AddRange(addgoods);

//сохраняем изменения

AssortmentBdEntities.GetContext().SaveChanges();

MessageBox.Show("Записи загружены");

goods.Clear();

goods = AssortmentBdEntities.GetContext().Goods.OrderBy(p => p.GoodName).ToList();

DataGridGood.ItemsSource = null;

DataGridGood.ItemsSource = goods;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

return dtexcel;

}

private void BtnAcceptance\_Click(object sender, RoutedEventArgs e)

{// передача в него выбранного услуги

Manager.MainFrame.Navigate(new AcceptancesPage((sender as Button).DataContext as Good));

}

private void BtnWriteOff\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new WriteOffPage((sender as Button).DataContext as Good));

}

private void BtnSells\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new SellPage((sender as Button).DataContext as Good));

}

private void BtnReturns\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new ReturnGoodPage((sender as Button).DataContext as Good));

}

private void Image\_MouseDown(object sender, MouseButtonEventArgs e)

{

}

private void ImgEdit\_MouseDown(object sender, MouseButtonEventArgs e)

{

Manager.MainFrame.Navigate(new AddGoodPage((sender as Button).DataContext as Good));

}

void PrintPriceTag(int row, Excel.Worksheet xlSheet, Good good1)

{

xlSheet.Cells[row + 2, 1] = good1.GoodName.ToString();

string s = "";

if (good1.Artikul != null) s = good1.Artikul.ToString();

xlSheet.Cells[row + 3, 2] = s;

xlSheet.Cells[row + 5, 1] = good1.Price.ToString();

xlSheet.Cells[row + 6, 2] = DateTime.Today.ToShortDateString();

}

private void BtnTag\_Click(object sender, RoutedEventArgs e)

{

Good good1 = (sender as Button).DataContext as Good;

string fileName = AppDomain.CurrentDomain.BaseDirectory + "\\" + "pricetag" + ".xltx";

Excel.Application xlApp = new Excel.Application();

Excel.Worksheet xlSheet = new Excel.Worksheet();

try

{

//добавляем книгу

xlApp.Workbooks.Open(fileName, Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

//делаем временно неактивным документ

xlApp.Interactive = false;

xlApp.EnableEvents = false;

Excel.Range xlSheetRange;

//выбираем лист на котором будем работать (Лист 1)

xlSheet = (Excel.Worksheet)xlApp.Sheets[1];

//Название листа

xlSheet.Name = "Ценник";

PrintPriceTag(1, xlSheet, good1);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

finally

{

//Показываем ексель

xlApp.Visible = true;

xlApp.Interactive = true;

xlApp.ScreenUpdating = true;

xlApp.UserControl = true;

}

}

private void BtnPrintAllPriceTag\_Click(object sender, RoutedEventArgs e)

{

Good good1 = (sender as Button).DataContext as Good;

string fileName = AppDomain.CurrentDomain.BaseDirectory + "\\" + "pricetag" + ".xltx";

Excel.Application xlApp = new Excel.Application();

Excel.Worksheet xlSheet = new Excel.Worksheet();

try

{

//добавляем книгу

xlApp.Workbooks.Open(fileName, Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

//делаем временно неактивным документ

xlApp.Interactive = false;

xlApp.EnableEvents = false;

Excel.Range xlSheetRange;

//выбираем лист на котором будем работать (Лист 1)

xlSheet = (Excel.Worksheet)xlApp.Sheets[1];

//Название листа

xlSheet.Name = "Ценники";

int row = 1;

int i = 0;

//PrintPriceTag(1, xlSheet, good1);

foreach (Good good in goods)

{

PrintPriceTag(row, xlSheet, good);

Excel.Range r = xlSheet.get\_Range("A" + row.ToString(), "C" + (row + 6).ToString());

//r.Insert(Excel.XlInsertShiftDirection.xlShiftDown, Excel.XlPasteSpecialOperation.xlPasteSpecialOperationAdd);

row += 8;

Excel.Range r2 = xlSheet.get\_Range("A" + row.ToString(), "A" + row.ToString());

r.Copy(r2);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

finally

{

//Показываем ексель

xlApp.Visible = true;

xlApp.Interactive = true;

xlApp.ScreenUpdating = true;

xlApp.UserControl = true;

}

}

}

}

**AcceptancePage.xaml**

<Page x:Class="WpfAssortmentCheck.Pages.AcceptancesPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:WpfAssortmentCheck.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Приёмка" Style="{StaticResource base\_page}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="300"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="40"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal" Grid.ColumnSpan="2">

<TextBlock Text="Выберите товар" Style="{StaticResource base\_textblock}" />

<ComboBox HorizontalAlignment="Stretch" x:Name="ComboGoods"

SelectionChanged="ComboGoodsSelectionChanged"

SelectedValuePath="GoodId"

DisplayMemberPath="GoodName"/>

</StackPanel>

<Grid x:Name="GridGood" Width="300" Grid.Row="1" Grid.Column="0">

<Grid.RowDefinitions>

<RowDefinition Height="310" />

<RowDefinition Height="auto" />

<RowDefinition Height="20" />

</Grid.RowDefinitions>

<Image Width="240" Grid.Row="0" Stretch="Uniform"

HorizontalAlignment="Center"

Source="{Binding Path=GetPhoto}"/>

<StackPanel Grid.Row="1" Height="100">

<TextBlock Padding="3" Height="Auto"

Style="{StaticResource item\_textblock}">

<TextBlock.Text>

<MultiBinding StringFormat="{}{0}{1}">

<Binding Path="GoodName"/>

<Binding Path="GetCount"/>

</MultiBinding>

</TextBlock.Text>

</TextBlock>

<TextBlock x:Name="tbPrice" Text="{Binding Price, StringFormat='c',

ConverterCulture='ru-RU'}" Height="Auto"

Style="{StaticResource item\_textblock}"

VerticalAlignment="Center"

Foreground="{StaticResource akcent}"

FontWeight="Bold" FontSize="20"/>

</StackPanel>

<TextBlock Text="{Binding GetStatus}" Height="Auto"

Style="{StaticResource item\_textblock}" Grid.Row="2"/>

</Grid>

<DataGrid x:Name="DtData" AutoGenerateColumns="False" Grid.Row="1" Grid.Column="1" IsReadOnly="True"

RowHeight="30"

SelectionMode="Single " LoadingRow="DataGridGoodLoadingRow">

<DataGrid.Columns>

<DataGridTextColumn Header="Договор №" Binding="{Binding AcceptanceId}"/>

<DataGridTextColumn Header="Дата приёмки" Binding="{Binding Date, StringFormat='d', ConverterCulture='ru-RU'}"/>

<DataGridTextColumn Header="Количество" Binding="{Binding Count}" Width="\*"/>

<DataGridTextColumn Header="Стоимость" Binding="{Binding Price, StringFormat='c', ConverterCulture='ru-RU'}"/>

<DataGridTextColumn Header="Поставщик" Binding="{Binding Developer.Name}" Width="\*"/>

</DataGrid.Columns>

</DataGrid>

<StackPanel Grid.Row="2" Grid.Column="1" Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center">

<Button x:Name="btnAdd" Content="Добавить" Click="btnAdd\_Click"/>

<Button x:Name="btnChange" Content="Изменить" Click="btnChange\_Click" />

<Button x:Name="btnDelete" Content="Удалить" Click="btnDelete\_Click"/>

</StackPanel>

</Grid>

</Page>

**AcceptancePage.xaml.cs**

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WpfAssortmentCheck.Models;

using WpfAssortmentCheck.Windows;

namespace WpfAssortmentCheck.Pages

{

/// <summary>

/// Логика взаимодействия для AcceptancesPage.xaml

/// </summary>

public partial class AcceptancesPage : Page

{

public AcceptancesPage(Good good)

{

InitializeComponent();

LoadData(good);

}

private void DataGridGoodLoadingRow(object sender, DataGridRowEventArgs e)

{

e.Row.Header = (e.Row.GetIndex() + 1).ToString();

}

// загрузка данных в DataGrid и ComboBox

void LoadData(Good good)

{

DtData.ItemsSource = AssortmentBdEntities.GetContext().Acceptances.Where(p => p.GoodId == good.GoodId).OrderBy(p => p.Date).ToList();

ComboGoods.ItemsSource = AssortmentBdEntities.GetContext().Goods.OrderBy(p => p.GoodName).ToList(); ;

ComboGoods.SelectedIndex = 0;

ComboGoods.SelectedValue = good.GoodId;

GridGood.DataContext = good;

}

// фильтрация продаж по товару

private void ComboGoodsSelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (ComboGoods.SelectedIndex >= 0)

{

int goodId = Convert.ToInt32(ComboGoods.SelectedValue);

var x = AssortmentBdEntities.GetContext().Acceptances.Where(p => p.GoodId == goodId).OrderBy(p => p.Date).ToList();

DtData.ItemsSource = x;

GridGood.DataContext = ComboGoods.SelectedItem;

}

}

private void btnAdd\_Click(object sender, RoutedEventArgs e)

{

try

{

Good g = ComboGoods.SelectedItem as Good;

AcceptanceWindow window = new AcceptanceWindow(new Acceptance(), g);

if (window.ShowDialog() == true)

{

AssortmentBdEntities.GetContext().Acceptances.Add(window.currentItem);

AssortmentBdEntities.GetContext().SaveChanges();

Store s = AssortmentBdEntities.GetContext().Stores.FirstOrDefault(p => p.GoodId == g.GoodId);

if (s is null)

{

Store store = new Store();

store.GoodId = g.GoodId;

store.Count = window.currentItem.Count;

AssortmentBdEntities.GetContext().Stores.Add(store);

AssortmentBdEntities.GetContext().SaveChanges();

}

else

{

s.Count += window.currentItem.Count;

AssortmentBdEntities.GetContext().Entry(s).State = EntityState.Modified;

AssortmentBdEntities.GetContext().SaveChanges();

}

LoadData(g);

MessageBox.Show("Запись добавлена", "Внимание", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch

{

MessageBox.Show("Ошибка");

}

}

private void btnChange\_Click(object sender, RoutedEventArgs e)

{

try

{

Good g = ComboGoods.SelectedItem as Good;

// если ни одного объекта не выделено, выходим

if (DtData.SelectedItem == null) return;

// получаем выделенный объект

Acceptance selected = DtData.SelectedItem as Acceptance;

double k = selected.Count;

AcceptanceWindow window = new AcceptanceWindow(

new Acceptance

{

AcceptanceId = selected.AcceptanceId,

GoodId = selected.GoodId,

Price = selected.Price,

Count = selected.Count,

Date = selected.Date,

DeveloperId= selected.DeveloperId

}

, g);

if (window.ShowDialog() == true)

{

selected = AssortmentBdEntities.GetContext().Acceptances.Find(window.currentItem.AcceptanceId);

// получаем измененный объект

if (selected != null)

{

selected.DeveloperId = window.currentItem.DeveloperId;

selected.GoodId = window.currentItem.GoodId;

selected.Price = window.currentItem.Price;

selected.Count = window.currentItem.Count;

selected.Date = window.currentItem.Date;

selected.DeveloperId = window.currentItem.DeveloperId;

AssortmentBdEntities.GetContext().Entry(selected).State = EntityState.Modified;

AssortmentBdEntities.GetContext().SaveChanges();

double d = k - selected.Count;

Store s = AssortmentBdEntities.GetContext().Stores.FirstOrDefault(p => p.GoodId == g.GoodId);

if (s != null)

{

s.Count -= d;

AssortmentBdEntities.GetContext().Entry(s).State = EntityState.Modified;

AssortmentBdEntities.GetContext().SaveChanges();

}

//g = AssortmentBdEntities.GetContext().Goods.Find(window.currentItem.GoodId);

//Good good = GridGood.DataContext as Good;

//good.Price = selected.Price;

//GridGood.DataContext = good;

LoadData(g);

MessageBox.Show("Запись изменена", "Внимание", MessageBoxButton.OK, MessageBoxImage.Information);

//ComboGoods.SelectedIndex = -1;

}

}

}

catch

{

MessageBox.Show("Ошибка");

}

}

private void btnDelete\_Click(object sender, RoutedEventArgs e)

{

try

{

Good g = ComboGoods.SelectedItem as Good;

// если ни одного объекта не выделено, выходим

if (DtData.SelectedItem == null) return;

// получаем выделенный объект

MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить запись? ", "Удаление", MessageBoxButton.OKCancel,

MessageBoxImage.Question);

if (messageBoxResult == MessageBoxResult.OK)

{

Acceptance deletedItem = DtData.SelectedItem as Acceptance;

Store s = AssortmentBdEntities.GetContext().Stores.FirstOrDefault(p => p.GoodId == g.GoodId);

if (s != null)

{

s.Count -= deletedItem.Count;

AssortmentBdEntities.GetContext().Entry(s).State = EntityState.Modified;

AssortmentBdEntities.GetContext().SaveChanges();

}

AssortmentBdEntities.GetContext().Acceptances.Remove(deletedItem);

AssortmentBdEntities.GetContext().SaveChanges();

LoadData(g);

MessageBox.Show("Запись удалена", "Внимание", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WpfEnglishSchoolApp.Models

{

public partial class Service

{

/// <summary>

/// Возвращает абсолютный путь к изображению

/// </summary>

public string GetPhoto

{

get

{

if (MainPhoto is null) return null;

return Directory.GetCurrentDirectory() + @"\Images\" + MainPhoto.Trim();

}

}

/// <summary>

/// Задает цвет фона товара

/// </summary>

public string GetColor

{

get

{

if (Discount > 0)

return "#98FB98";

else

return "#FFF";

}

}

public string GetVisibility

{

get

{

if (Discount == 0)

return "Collapsed";

else

return "Visible";

}

}

public string TotalPrice

{

get

{

double p = Price - Price \* Discount / 100;

return $"{p:F2} рублей за {TimeLength} минут";

}

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WpfEnglishSchoolApp.Models

{

public partial class Good

{

/// <summary>

/// Возвращает абсолютный путь к изображению

/// </summary>

public string GetPhoto

{

get

{

if (MainPhoto is null)

return null;

return Directory.GetCurrentDirectory() + @"\Images\" + MainPhoto.Trim();

}

}

/// <summary>

/// Задает цвет фона услуги

/// </summary>

public string GetColor

{

get

{

if (Stores.Count > 0)

return "#fff";

else

return "#8C8181";

}

}

/// <summary>

/// Текстовое представление активности услуги

/// </summary>

public string GetStatus

{

get

{

if (Stores.Count > 0)

return $"в наличии {Stores.Count} шт.";

else

return "нет в наличии";

}

}

/// <summary>

/// Возвращает количество дополнительных товаров

/// </summary>

public string GetCount

{

get

{

if (Complects.Count > 0)

return $"({Complects.Count})";

else

return "";

}

}

public int GetCountStore

{

get

{

return Stores.Count;

}

}

public string GetInfo

{

get

{

return $"{GoodName} - {Price.ToString("c")} рублей.";

} } }}

**Приложение Б**

«АИС «Школа английского языка “ABC”

(подсистема работы с услугами)»

Спецификация

460.ЗМК.00005-01 01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Обозначение** | | |  | | --- | | **Наименование** | | |  | | --- | | **Примечание** | |
|  | |  | | --- | | Документация | |  |
| 460.ЗМК.00005-01 12 01 | «АИС «Школа английского языка “ABC”(подсистема работы с услугами)» |  |
|  | Текст программы |  |
| 460.ЗМК.00005-01 13 01 | «АИС «Школа английского языка “ABC”(подсистема работы с услугами)» |  |
|  | Описание программы |  |
| 460.ЗМК.00005-01 34 01 | «АИС «Школа английского языка “ABC”(подсистема работы с услугами)» |  |
|  | Руководство пользователя |  |
| 460.ЗМК.00005-01 81 01 | «АИС «Школа английского языка “ABC”(подсистема работы с услугами)» |  |
|  | Пояснительная записка |  |

**Приложение B**

**Диск с программным средством**