**Оглавление**

[Аннотация 4](#_Toc26196310)

[Введение 5](#_Toc26196311)

[1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ 6](#_Toc26196312)

[1.1 Основное функциональное назначение программного средства 6](#_Toc26196313)

[1.2 Полное наименование программного средства 6](#_Toc26196314)

[1.3 Условное обозначение программного средства 6](#_Toc26196315)

[1.4 Разработчики программного средства 6](#_Toc26196316)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 7](#_Toc26196317)

[2.1 Основание для разработки 7](#_Toc26196318)

[2.2 Назначение разработки 7](#_Toc26196319)

[2.3 Требование к программному средству 7](#_Toc26196320)

[2.4 Требования к программной документации 8](#_Toc26196321)

[2.5 Требования к эргономике и технической эстетике 8](#_Toc26196322)

[2.6 Стадии и этапы разработки 9](#_Toc26196323)

[2.7 Порядок контроля и приемки 10](#_Toc26196324)

[3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ 11](#_Toc26196325)

[3.1 Декомпозиция поставленной задачи 11](#_Toc26196326)

[3.2 Общая архитектура программного средства 12](#_Toc26196327)

[3.3 Разработка алгоритма решения задачи 15](#_Toc26196328)

[3.4 Реализация функционального назначения программного средства 16](#_Toc26196329)

[3.5 Структурная организация данных 16](#_Toc26196330)

[3.6 Разработка интерфейса ПС 20](#_Toc26196331)

[3.7 Описание структуры выходной информации 25](#_Toc26196332)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 26](#_Toc26196333)

[4.1 Назначение программного средства 26](#_Toc26196334)

[4.2 Условия выполнения программного средства 26](#_Toc26196335)

[4.3 Эксплуатация программного средства 26](#_Toc26196336)

[4.4 Сообщения пользователю 32](#_Toc26196337)

[5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА 34](#_Toc26196338)

[5.1 Обоснование необходимого количества тестов 34](#_Toc26196339)

[5.2 Описание тестовых пакетов 35](#_Toc26196340)

[6. ЗАКЛЮЧЕНИЕ 42](#_Toc26196341)

[6.1 Оценка качества программного средства с помощью метрик 42](#_Toc26196342)

[7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45](#_Toc26196343)

[Приложение А листинг программного модуля 47](#_Toc26196344)

[Приложение B диск с исполняемым кодом программного модуля 65](#_Toc26196345)

**Аннотация**

На курсовой проект студента Белова А.С. на тему: Разработка интегрированного программного модуля «Охотхозяйство ООО Охотничьи традиции». Курсовой проект представлен пояснительной запиской на 65 листов машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В главе 1 содержится общие сведения о программном средстве Охотхозяйство ООО Охотничьи традиции. 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 – интегрированного программного модуля «Охотхозяйство ООО Охотничьи традиции», 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 курсовом проекте была поставлена задача разработки интегрированного программного модуля «Охотхозяйство ООО Охотничьи традиции», который позволяет автоматизировать учет мероприятий.

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 моего курсового проекта.

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.

**1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ**

**1.1 Основное функциональное назначение программного средства**

Программа предназначена для автоматизации учета мероприятий. 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**

Разработка интегрированного программного модуля «Охотхозяйство ООО Охотничьи традиции».

**1.3 Условное обозначение программного средства**

«Hunt.01»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 205 Белов А.С.

**2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**2.1 Основание для разработки**

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

**2.2 Назначение разработки**

Программа «Hunt.01» работает под управление операционной системы Windows 7, Windows 8, Windows 10. Программа предназначена для автоматизации деятельности администратора по учету мероприятий охотхозяйства . Программа позволяет вводить, изменять и удалять данные мероприятиях.

**2.3 Требование к программному средству**

«Hunt.01» 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 «Hunt.01» является кроссплатформенным продуктом, что позволяет не ограничиваться в работе с определенной версией ОС.

«Hunt.01» 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.

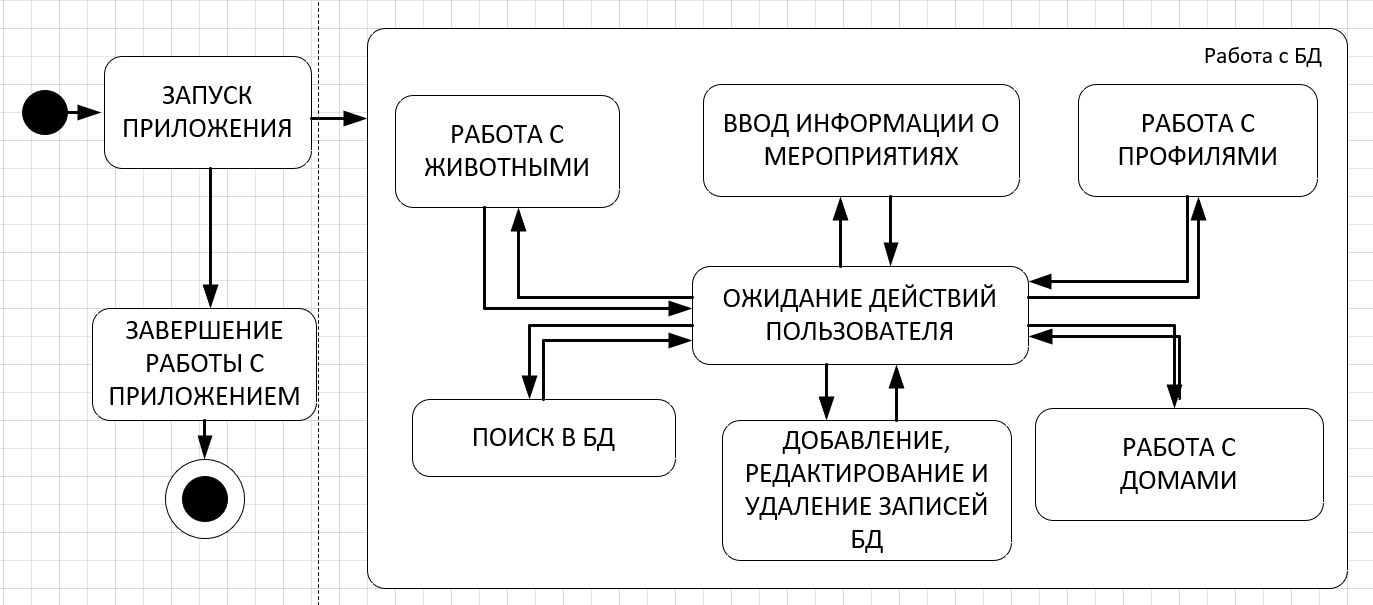


Рис.1 Декомпозиция поставленной задачи

**3.2 Общая архитектура программного средства**

Объектом моделирования является Афиша.

Рассматриваемые процессы: Добавление и удаление информации о мероприятиях.

Объекты моделирования представлены на диаграмме классов, рис.2.

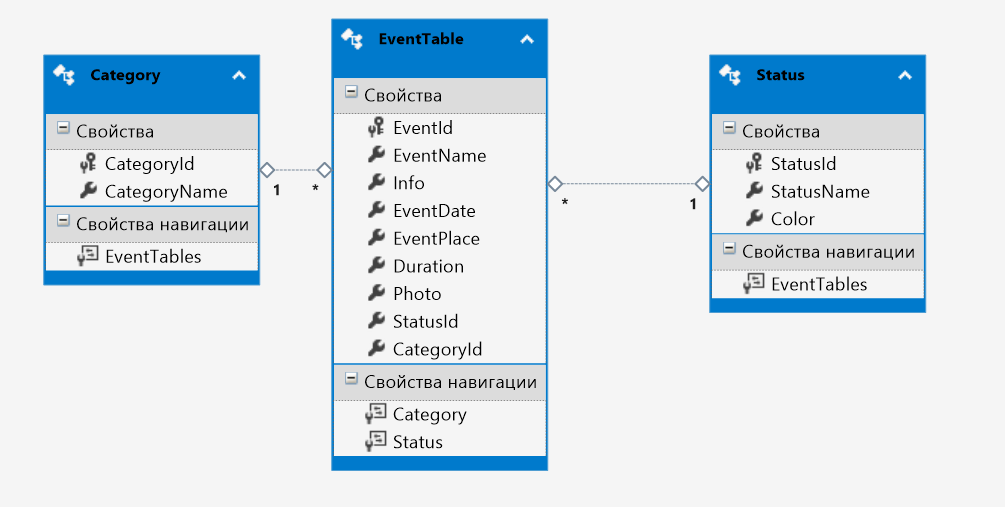


Рис.2 Диаграмма классов

По полученной декомпозиции задач была спроектирована следующая архитектура программного средства.

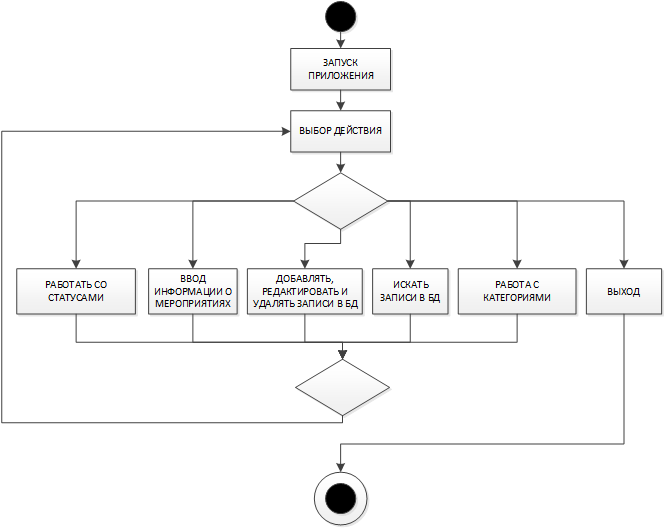


Рис.3 Схема событийно-ориентированной архитектуры

В «Afisha1.01» 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 «Afisha1.01» имеет следующий набор входных данных, такие как: мероприятия и информация о них.

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

Выходными данными являются: отображение списка мероприятий в виде плиточного интерфейса.

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 «БД Афиша» 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 «Мероприятие». Реляционная модель автоматизированной системы соответствует всем 12 правилам Кодда. (рис.6)

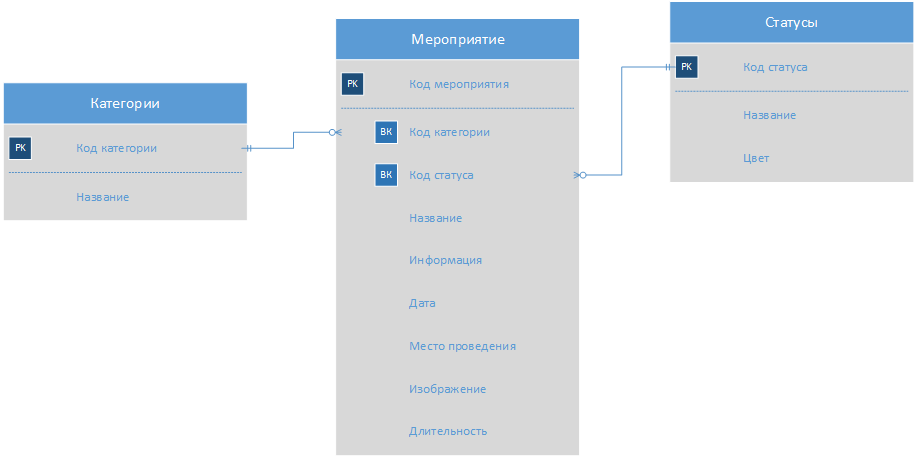


Рис.6 Реляционная модель

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\_категории (таблица Категории), Код\_мероприятия (Таблица Мероприятия), Код\_статуса (Таблица Статусы). 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 “EventTable” – “EventId”. Связь таблиц “Category” и “EventTable” по полю “CategoryId” один ко многим (1:М). Таблицы “Service” и “EventTable” связаны по полю “StatusId” типом связи 1:М. Все связи между таблицами базы данных типа один ко многим.



Рис.7 Схема базы данных

Таблица “Status”(Статусы)

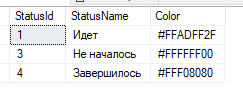


Рис.8 Таблица Status

Таблица “Category”(Категории)

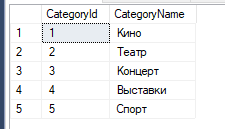


Рис.9 Таблица Category

Таблица “EventTable”(Мероприятия)

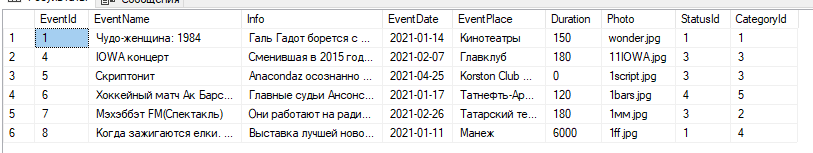


Рис.10 Таблица EventTable

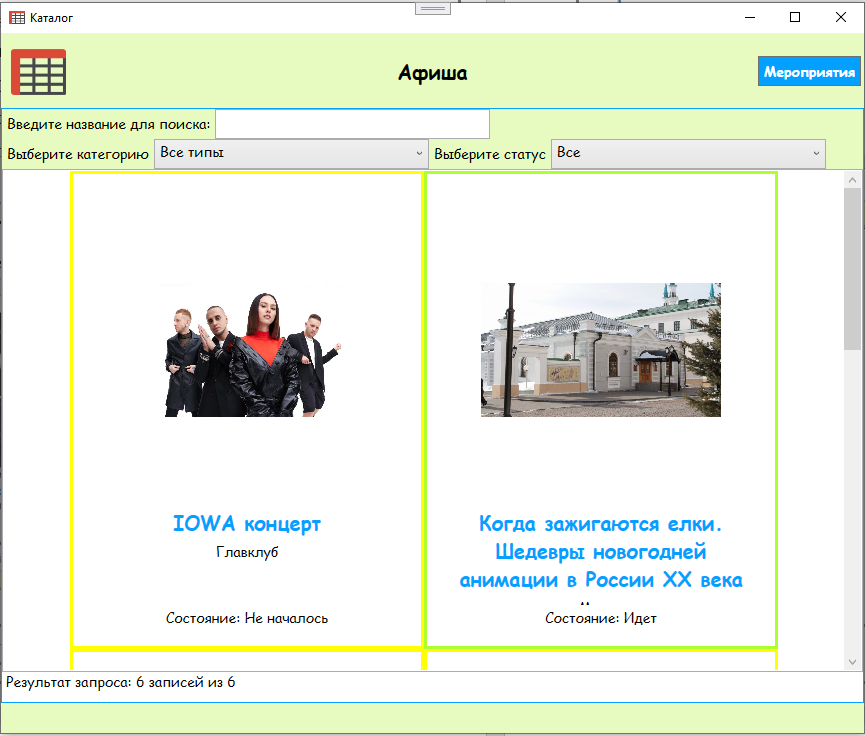
**3.6 Разработка интерфейса ПС**

В приложении не предусмотрена авторизация, так как оно однопользовательское. Для запуска приложения достаточно щелкнуть дважды левой кнопкой мыши по ярлыку.

После старта приложения пользователь приступит к работе с главной формой программы, на которой и будут отображаться все события в базе данных. «Afisha1.01» 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 Каталог**. Основная форма приложения, в которой в виде плитки отображаются экспонатфы музея. Компоненты формы: ListView – для отображения мероприятий, Combobox – выпадающие списки, TextBox и Button. (Рис.11)



TextBox

Button

ComboBox

ListView

Рис.11 Каталог

**Страница Мероприятия**. Компоненты Страницы: DataGrid – для отображения списка мероприятий и кнопки Button для перехода на дополнительные страницы. (Рис.12)



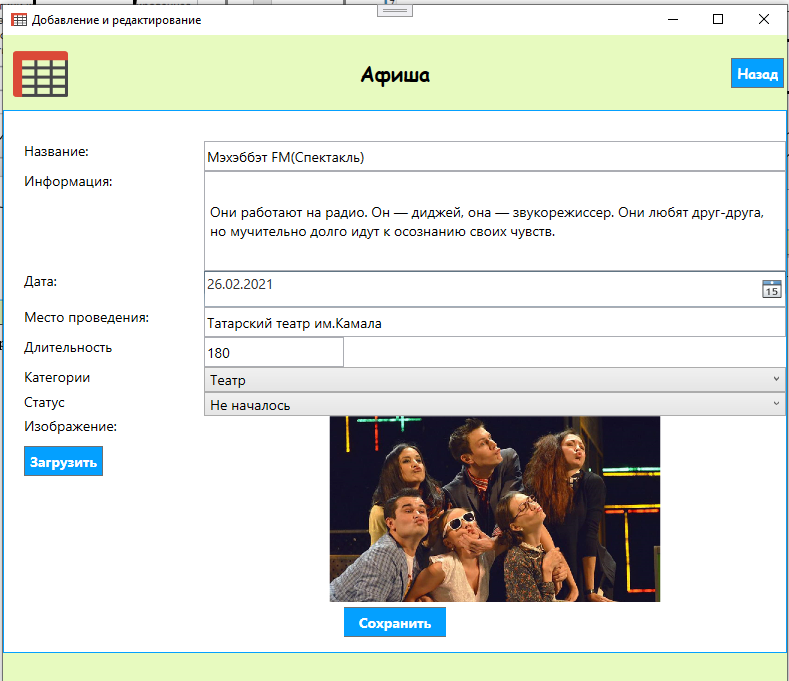
Button

Button

DataGrid

Рис.12 Афиша

**Страница Добавление и редактирование**. Интерфейс данной формы состоит из компонентов: TextBox – поля ввода, ComboBox – выпадающие списки, Image – дляотображения изображения и Button. Также есть возможность открытия дополнительных окон. (Рис.13)



TextBox

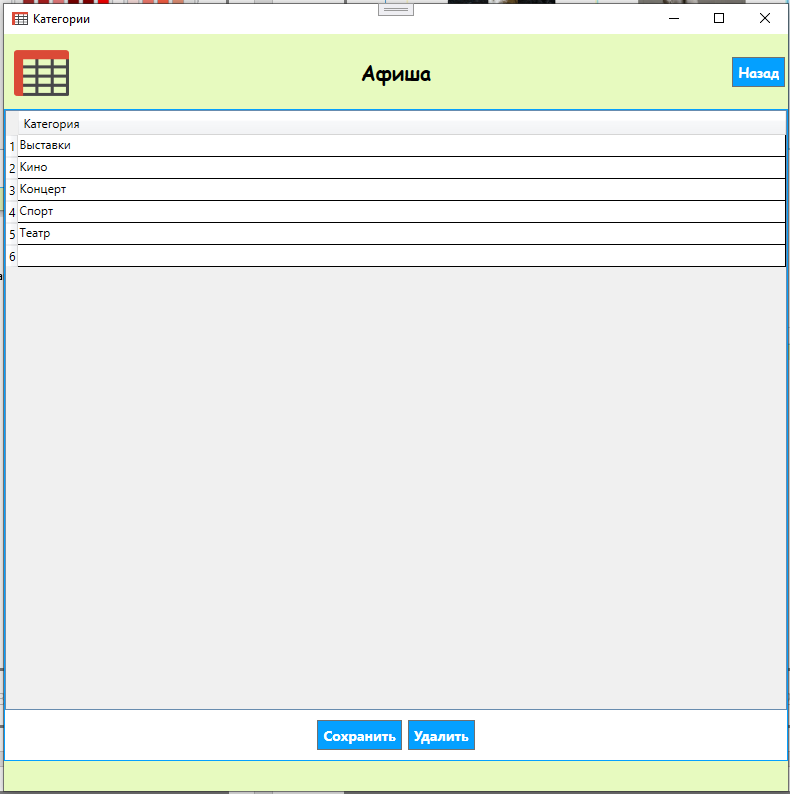
Button

ComboBox

Image

Рис.13 Страница Добавление и редактирование

**Страница Категории**. Интерфейс данной формы состоит из компонентов: DataGrid и кнопок.(Рис.14)



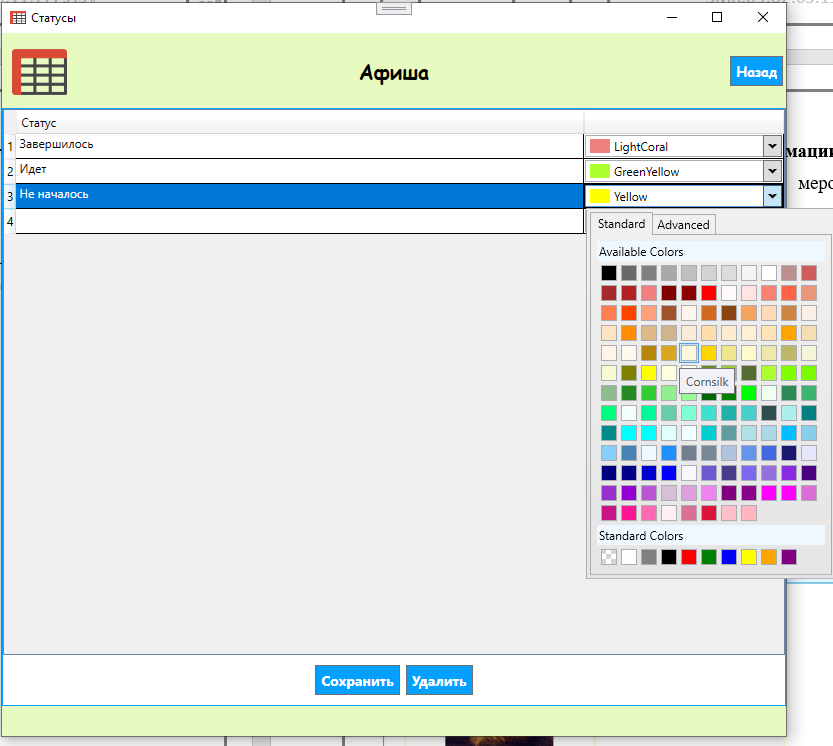
Кнопка Назад

Кнопки Сохранить и Удалить

DataGrid

Рис.14 Страница Категории

**Страница Статусы**. Интерфейс данной формы состоит из компонентов: DataGrid и кнопок.(Рис.15)



Кнопка Назад

Кнопки Сохранить и Удалить

DataGrid

Рис.15 Страница Статусы

**3.7 Описание структуры выходной информации**

Выходными данными являются списки мероприятий отфильтрованные по разным критериям.

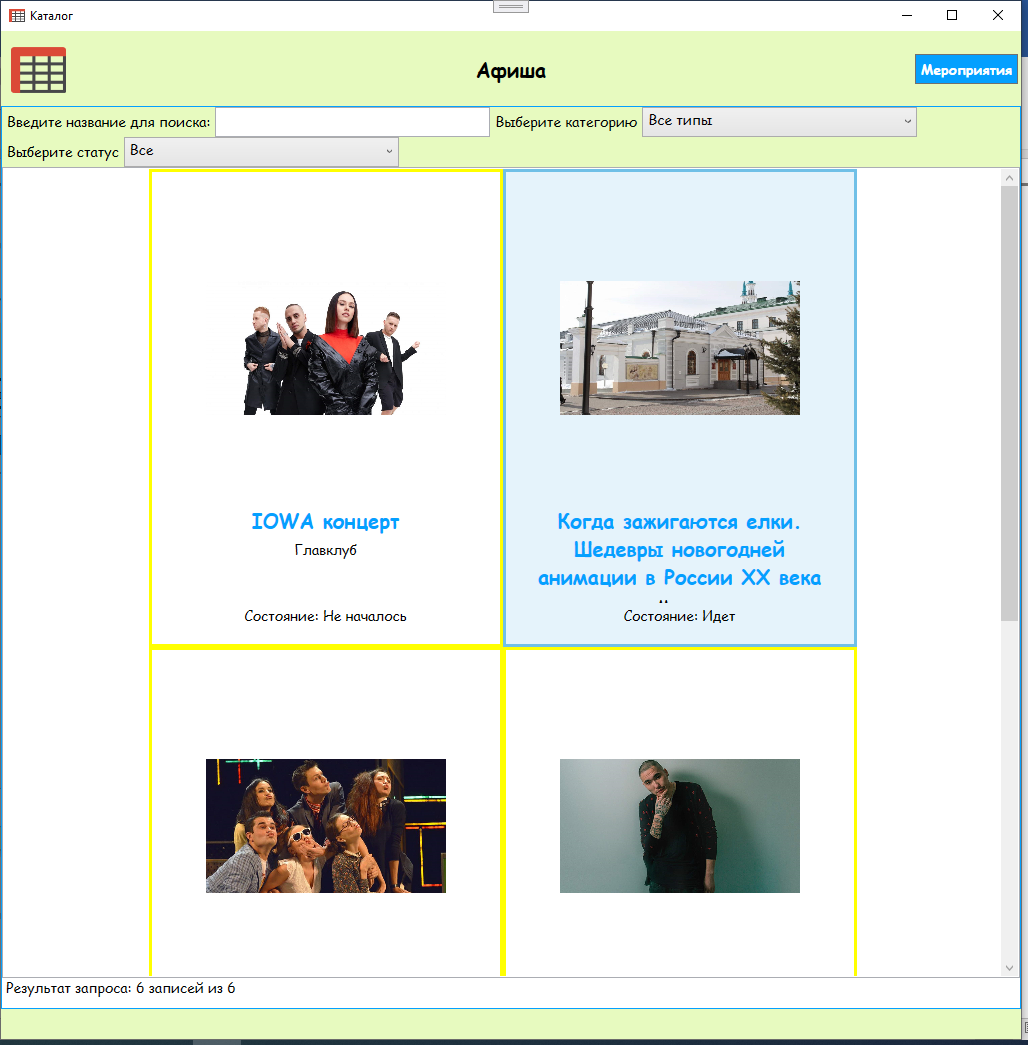


Рис.16 Списки мероприятий

**4. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**4.1 Назначение программного средства**

Назначением программного средства является автоматизация учета мероприятий.

**4.2 Условия выполнения программного средства**

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

-Компьютер с установленной на нём «Afisha1.01».

Подготовительные действия:

-На иконке «Afisha1.01» рабочего стола произвести двойной щелчок левой кнопкой мыши.

**4.3 Эксплуатация программного средства**

Этот раздел содержит обзор приложений «Afisha1.01» и ее основных параметров.

***Активация продукта***

Активировать копию программы Afisha1.01 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 старта приложения откроется главное окно приложения – Каталог мероприятий в плиточном интерфейсе. На форме есть кнопка для перехода на страницу мероприятий. Есть поля для поиска мероприятия по назвванию и выпадающие списки для фильтрации. (Рис.17)

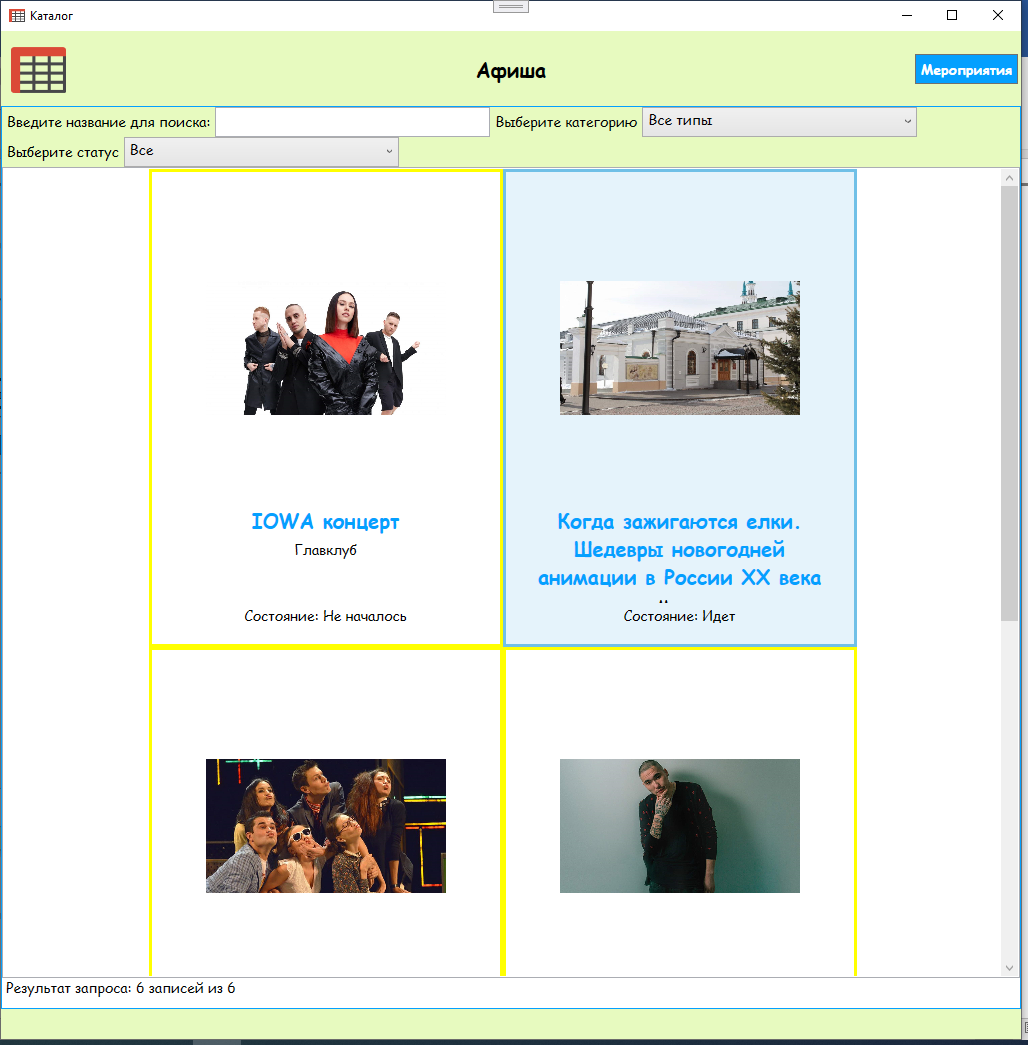


Рис.17 Окно Афиша

Кликнув по кнопке «Выход» программа потребует подтверждение. (Рис.18)



Рис.18 Сообщение программы

Ниже описаны пункты меню управления и информации.

«Страница Мероприятия»: эта форма предназначена для отображения списка мероприятий, а также для перехода на дополнительные страницы. (Рис.19)



Рис.19 Страница Экспонаты

«Страница Добавление и и редактирования»: эта форма предназначена для редактирования и добавление данных в таблице Мероприятия. (Рис.22)

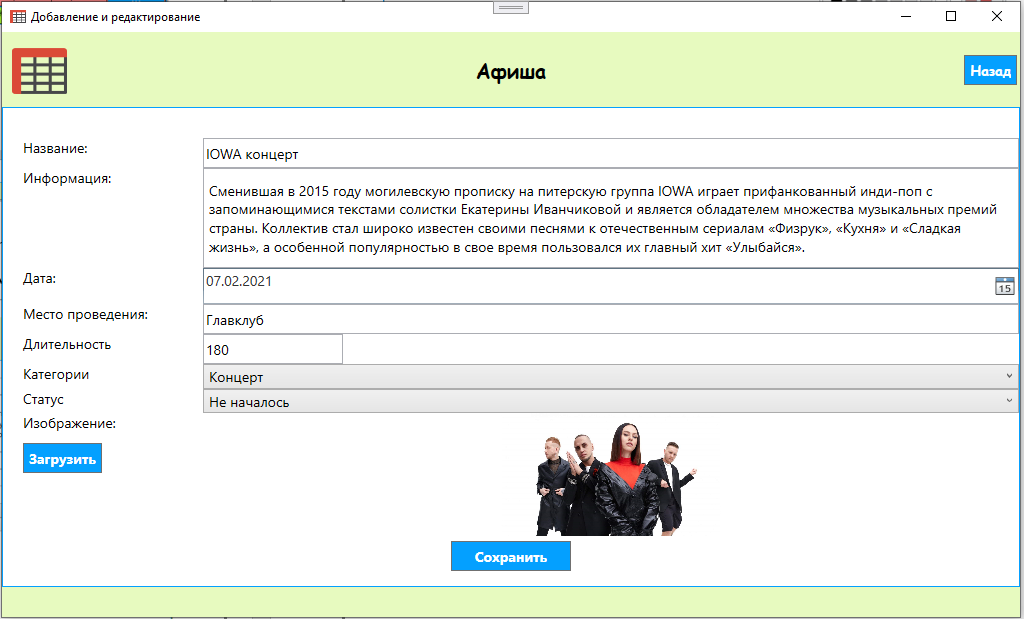


Рис.20 Страница Добавление и редактирование

«Статусы»: эта страница предназначена для ввода, удаления и редактирования данных в таблице Статусы. (Рис.21)

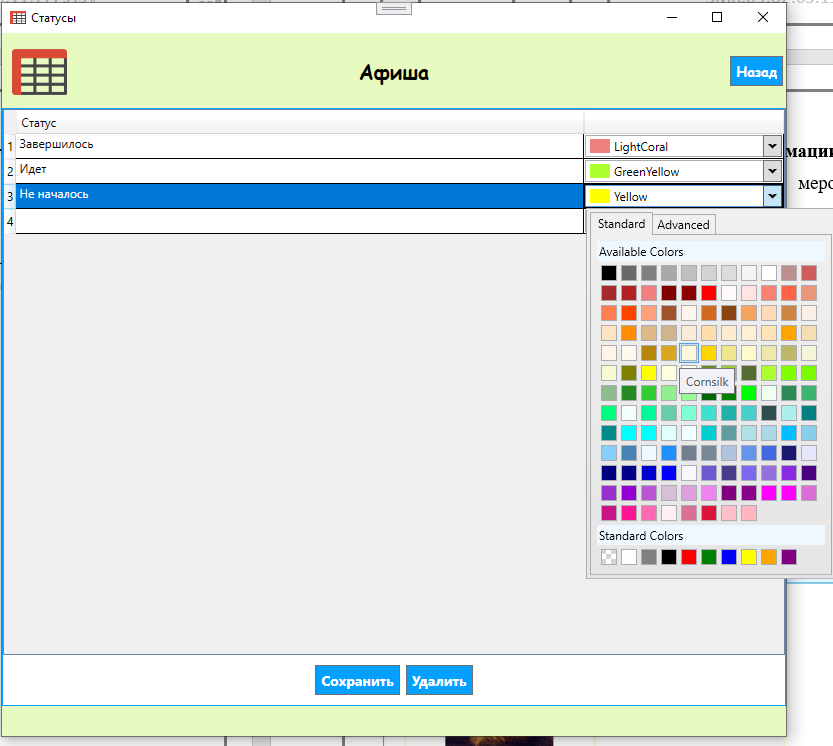


Рис.21 Страница Статусы

«Категории»: эта страница предназначена для ввода, удаления и редактирования данных в таблице Категории. (Рис.22)

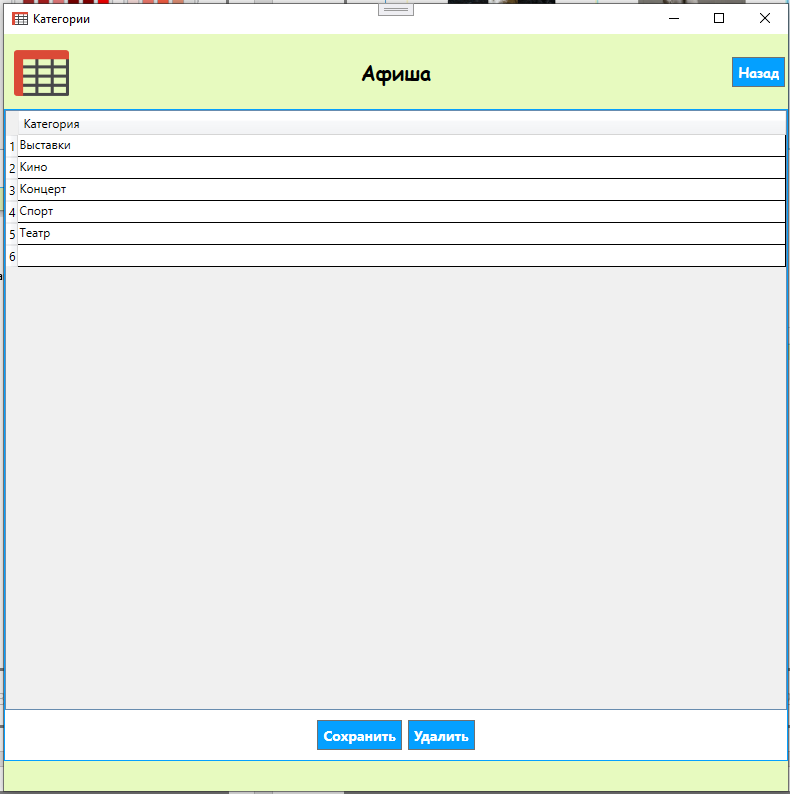


Рис.22 Страница Категории

Работа с Afisha1.01

После запуска программы Вы перейдете на главную форму программы. В формах «Мероприятия», «Категории», «Статусы», «Добавление и удаление» 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 Afisha1.01 кликните на соответствующей кнопке «Выход».

Основные кнопки Afisha1.01

Кнопки управления «Добавить запись», «Удалить запись», «Сохранить» расположены на каждой конкретной форме в базе данных.

Действия этих кнопок приведены в таблице 1:

Таблица-1 Действие кнопок

|  |  |
| --- | --- |
| Добавить запись | Прежде чем добавить запись в таблицу необходимо нажать на данную кнопку. Поля формы будут очищены, пользователю требуется ввести необходимые данные. Если пользователь нажмет кнопку «Сохранить данные», то введенная информация добавится в БД. Если нажать кнопку «Отменить», то введенная информация пользователем удалится. |
| Удалить запись/отменить  , , | Кнопка «Удалить запись» позволяет отменить действия при вводе, либо удаляет конкретную запись. |
| Сохранить данные/обновить запись  , , | Кнопка «Сохранить данные» добавляет запись введенную пользователем в БД, либо обновляет существующую. |
| Редактировать | Открывает форму в режиме редактирования информации |
|  | «Поиск» позволяет найти нужную информацию. |

**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.23)



Рис.23 Сообщение Подтверждение удаления

После добавления новой записи, пользователю выводится сообщение. (Рис.24)



Рис.24 Сообщение Запись добавлена

После обновления записи, пользователю выводится сообщение. (Рис.25)



Рис.25 Сообщение Запись обновлена

При попытке удалить связанные записи, программа выводит пользователю сообщение о невозможности действия. (Рис.26)



Рис.26 Сообщение Ошибка удаления

**5. РАЗРАБОТКА ТЕСТОВОГО НАБОРА**

**5.1 Обоснование необходимого количества тестов**

Для разработанного алгоритма был построен граф.

****

Рис.27 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 Afisha1.01 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 Описание тестовых пакетов**

Таблица 2 – Тестирование корректного добавления информации в Статусы

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

Таблица 3 – Тестирование некорректного добавления информации в таблицу Статусы

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

Таблица 4 – Тестирование корректного редактирования информации в таблице Статусы

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Статусы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Статусы | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Статус = Завершилось  Цвет = Красный | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Статусы  открыта | | Отображается форма Статусы |  |

Таблица 5 – Тестирование некорректного редактирования информации в таблице Статусы

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

Таблица 6 – Тестирование корректного добавления информации в таблицу Мероприятие

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Мероприятие | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Мероприятие | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = IOWA концерт  Информация = Концерт  Дата проведения = 7.02.2021  Место проведения = Главклуб  Длительность 180 минут  Категория = концерт  Статус = Не началось  Изображение = загружено | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Мероприятие открыта | | Отображается форма Мероприятие |  |

Таблица 7 – Тестирование некорректного добавления информации в таблицу Мероприятие

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Мероприятие | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Мероприятие | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = , Информация = ,  Дата проведения = , Место проведения = , Длительность 180 минут, Категория =, Статус = Не началось, Изображение = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Мероприятие  открыта | | Отображается форма Мероприятие |  |

Таблица 8 – Тестирование корректного редактирования информации в таблице Мероприятие

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Мероприятие | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Мероприятие | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название = IOWA концерт  Информация = Концерт  Дата проведения = 7.02.2021  Место проведения = Главклуб  Длительность 180 минут  Категория = концерт  Статус = Не началось  Изображение = загружено | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Мероприятие открыта | | Отображается форма Мероприятие |  |

Таблица 9 – Тестирование некорректного редактирования информации в таблице Мероприятие

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Мероприятие | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Мероприятие | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название = , Информация = ,  Дата проведения = , Место проведения = , Длительность 180 минут, Категория =, Статус = Не началось, Изображение = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Мероприятие открыта | | Отображается форма Мероприятие |  |

Таблица 10 – Тестирование корректного добавления информации в Категории

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

Таблица 11 – Тестирование некорректного добавления информации в таблицу Категории

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

Таблица 12 – Тестирование корректного редактирования информации в таблице Категории

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Категории | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Категории | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Категория = Концерт | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Категории открыта | | Отображается форма Категории |  |

Таблица 13 – Тестирование некорректного редактирования информации в таблице Категории

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

**6. ЗАКЛЮЧЕНИЕ**

**6.1 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 7 Ultimate 32бит.

Процессор: Intel Pentium G2030 3.0 GHz.

ОЗУ: 2 Gb

Таблица 14 – 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 15 – 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 16 – Сложность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип объекта | Количество | Вес | | | Итого |
|  |  | Простой | Средний | Сложный |  |
| Экран | 9 | x1 | x2 | **x3** | =27 |
| Отчет | 2 | x2 | **x5** | x8 | =10 |

Таблица 17 – Сложность экрана

|  |  |  |  |
| --- | --- | --- | --- |
| Экраны | Количество Категорииских (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 18 – Сложность отчета

|  |  |  |  |
| --- | --- | --- | --- |
| Отчеты | Количество Категорииских (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 |

Производительность = 4 тысяч строк кода/2 месяца= 2 тысяч строк кода/месяц

Качество = 2 ошибки /4 тысяч строк кода=0,5 ошибок/тысяч строк кода

**7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Г. Шилдт C#: Учебный курс. - Спб.: Питер, 2016. - 512с

2. Либерти, Д. Программирование на Си Шарп. - М.: Символ-плюс, 2016. - 684 с.

3. Ватсон К. Си Шарп/К.Ватсон.-М.:Лори, 2017.-862 с

4. Балена Ф., Димауро Д. Современная практика программирования на Visual Basic и Visual C#.-М.:Русская редакция,2015,604с.

5. Енин А., Енин Н. Локальная СУБД своими руками. Учимся на примерах.-М.:СОЛОН-ПРЕСС,2015,464с.,СД

6. Лабор В.В. Си Шарп: СоЗдание приложений для Windows/-Мн.: Харвест,2016-384с.

7. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2009-512с.

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#. 2016 г.

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

14. Никита Культин. Основы программирования в MicrosoftVisualC# 2017. 2017 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, 2017.

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

16. Йен Гриффитс. Программирование на С#. 2016 г.

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

**Приложение А**

Разработка интегрированного программного модуля «Афиша»

Текст кода программного средства

460.ЗМК.14-01 12 01

**Текст кода программного средства**

**Словарь ресурсов**

<Application x:Class="AfishaApp.App"

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

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

xmlns:local="clr-namespace:AfishaApp"

StartupUri="Windows/MainWindow.xaml">

<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>

<!--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="{Binding GetColor}"/>

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

</Style>

<Style TargetType="DataGridRow" x:Key="grid\_item">

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

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

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

</Style>

</Application.Resources>

</Application>

**Класс Manager**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Controls;

namespace MapOfSomeApp.Models

{

public class Manager

{

public static Frame MainFrame { get; set; }

}

}

**Класс** EventTable**.cs**

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан по шаблону.

//

// Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.

// Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.

// </auto-generated>

//------------------------------------------------------------------------------

namespace AfishaApp.Models

{

using System;

using System.Collections.Generic;

using System.IO;

public partial class EventTable

{

public int EventId { get; set; }

public string EventName { get; set; }

public string Info { get; set; }

public System.DateTime EventDate { get; set; }

public string EventPlace { get; set; }

public int Duration { get; set; }

public string Photo { get; set; }

public int StatusId { get; set; }

public int CategoryId { get; set; }

public string GetPhoto

{

get

{

if (Photo is null)

return null;

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

}

}

/// <summary>

/// Задает цвет фона товара

/// </summary>

public string GetColor

{

get

{

return Status.Color;

}

}

/// <summary>

/// Текстовое представление активности товара

/// </summary>

public string GetStatus

{

get

{

return $"Состояние: {Status.StatusName}";

}

}

public virtual Category Category { get; set; }

public virtual Status Status { get; set; }

}

}**Форма Главная**

**Код интерфейса**

<Window x:Class="AfishaApp.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:AfishaApp"

mc:Ignorable="d"

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

Style="{StaticResource base\_window}"

Height="450" Width="800"

Closing="WindowClosing"

Closed="WindowClosed" WindowStartupLocation="CenterScreen" Icon="/AfishaApp;component/Resources/spreadsheet.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

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

<Image Source="/AfishaApp;component/Resources/spreadsheet.png"

Margin="10"

HorizontalAlignment="Left"/>

<TextBlock Grid.Row="0" Text="Афиша"

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="Мероприятия" Click="BtnEditGoodsClick"/>

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

</StackPanel>

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

</Grid>

</Window>

**Код программы**

using AfishaApp.Models;

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 AfishaApp.Pages;

namespace AfishaApp

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

MainFrame.Navigate(new Сatalog());

Manager.MainFrame = MainFrame;

}

private void WindowClosed(object sender, EventArgs e)

{

// показать владельца формы

this.Close();

//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 EventsPage());

}

// 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)

{

// показываем кнопку назад

BtnBack.Visibility = Visibility.Visible;

// скрываем кнопку товары

BtnEditGoods.Visibility = Visibility.Collapsed;

}

else

{

// скрываем кнопку назад

BtnBack.Visibility = Visibility.Collapsed;

// показываем кнопку товары

BtnEditGoods.Visibility = Visibility.Visible;

}

}

}

}

**Страница Каталог**

**Код интерфейса**

<Page x:Class="AfishaApp.Pages.Сatalog"

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:AfishaApp.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 категорию"

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

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

SelectionChanged="ComboTypeSelectionChanged"

DisplayMemberPath="CategoryName"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Выберите статус"

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

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

SelectionChanged="ComboTypeSelectionChanged"

DisplayMemberPath="StatusName"/>

</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}"

VerticalAlignment="Center"

Foreground="{StaticResource akcent}"

FontWeight="Bold" FontSize="20"

Text="{Binding EventName}"/>

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

Style="{StaticResource item\_textblock}"

/>

</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>

**Код программный**

using AfishaApp.Models;

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;

namespace AfishaApp.Pages

{

/// <summary>

/// Логика взаимодействия для Сatalog.xaml

/// </summary>

public partial class Сatalog : Page

{

int \_itemcount = 0;

public Сatalog()

{

InitializeComponent();

// загрузка данных в combobox + добавление дополнительной строки

var categories = AfishaBDEntities.GetContext().Categories.OrderBy(p => p.CategoryName).ToList();

categories.Insert(0, new Category

{

CategoryName = "Все типы"

}

);

ComboCategory.ItemsSource = categories;

ComboCategory.SelectedIndex = 0;

var statuses = AfishaBDEntities.GetContext().Status.OrderBy(p => p.StatusName).ToList();

statuses.Insert(0, new Status

{

StatusName = "Все"

}

);

ComboStatus.ItemsSource = statuses;

ComboStatus.SelectedIndex = 0;

// загрузка данных в listview сортируем по названию

LViewGoods.ItemsSource = AfishaBDEntities.GetContext().EventTables.OrderBy(p => p.EventName).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)

{

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

LViewGoods.ItemsSource = AfishaBDEntities.GetContext().EventTables.OrderBy(p => p.EventName).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 = AfishaBDEntities.GetContext().EventTables.OrderBy(p => p.EventName).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();

if (ComboStatus.SelectedIndex > 0)

currentGoods = currentGoods.Where(p => p.StatusId == (ComboStatus.SelectedItem as Status).StatusId).ToList();

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

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

}

}

}

**Страница Мероприятие**

**Файл интерфейса**

<Page x:Class="AfishaApp.Pages.EventsPage"

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:AfishaApp.Pages"

mc:Ignorable="d"

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

Title="Мероприятия" 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"

ItemContainerStyle="{StaticResource grid\_item}"

LoadingRow="DataGridGoodLoadingRow">

<DataGrid.Columns>

<DataGridTemplateColumn Width="100" Header="Изображение">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

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

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTextColumn Binding="{Binding EventName}" Header="Название" Width="2\*"/>

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

<DataGridTextColumn Binding="{Binding EventDate, StringFormat='d'}" Header="Дата проведения" Width="\*"/>

<DataGridTextColumn Binding="{Binding EventPlace}" Header="Место проведения" Width="\*"/>

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

<DataGridTextColumn Binding="{Binding Category.CategoryName}" Header="Категория" Width="\*"/>

<DataGridTextColumn Binding="{Binding Status.StatusName}" Header="Статус" Width="\*"/>

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

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

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

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<Button x:Name="BtnAdd" Grid.Row="1" Width="100" HorizontalAlignment="Right" Margin="3 0 120 0"

Content="Добавить" Click="BtnAddClick"/>

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

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

<Button x:Name="BtnEditStatus" Grid.Row="1" Width="100" HorizontalAlignment="Left" Content="Статусы" Click="BtnEditStatusClick"/>

<Button x:Name="BtnEditCategory" Grid.Row="1" Width="100" HorizontalAlignment="Left" Margin="120 3 0 0" Content="Категории" Click="BtnEditCategoryClick"/>

</Grid>

</Page>

**Файл программного кода**

using AfishaApp.Models;

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 AfishaApp.Pages;

namespace AfishaApp.Pages

{

/// <summary>

/// Логика взаимодействия для EventsPage.xaml

/// </summary>

public partial class EventsPage : Page

{

public EventsPage()

{

InitializeComponent();

}

private void ButtonClick(object sender, RoutedEventArgs e)

{

// открытие редактирования товара

// передача выбранного товара в AddGoodPage

Manager.MainFrame.Navigate(new AddEventPage((sender as Button).DataContext as EventTable));

}

private void PageIsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

//событие отображения данного Page

// обновляем данные каждый раз когда активируется этот Page

if (Visibility == Visibility.Visible)

{

//загрузка обновленных данных

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

List<EventTable> goods = AfishaBDEntities.GetContext().EventTables.OrderBy(p => p.EventName).ToList();

DataGridGood.ItemsSource = null;

DataGridGood.ItemsSource = goods;

}

}

private void BtnAddClick(object sender, RoutedEventArgs e)

{

// открытие AddGoodPage для добавления новой записи

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

}

private void BtnDeleteClick(object sender, RoutedEventArgs e)

{

// удаление выбранного товара из таблицы

//получаем все выделенные товары

var selectedGoods = DataGridGood.SelectedItems.Cast<EventTable>().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

EventTable x = selectedGoods[0];

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

// если да, то выбрасывается исключение и удаление прерывается

// удаляем товара

AfishaBDEntities.GetContext().EventTables.Remove(x);

//сохраняем изменения

AfishaBDEntities.GetContext().SaveChanges();

MessageBox.Show("Записи удалены");

List<EventTable> goods = AfishaBDEntities.GetContext().EventTables.OrderBy(p => p.EventName).ToList();

DataGridGood.ItemsSource = null;

DataGridGood.ItemsSource = goods;

}

catch (Exception ex)

{

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

}

}

}

// отображение номеров строк в DataGrid

private void DataGridGoodLoadingRow(object sender, DataGridRowEventArgs e)

{

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

}

private void BtnEditStatusClick(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new StatusPage());

}

private void BtnEditCategoryClick(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new CategoryPage());

}

}

}

**Страница Добавления и редактирования Мероприятие**

**Файл интерфейса**

<Page x:Class="AfishaApp.Pages.AddEventPage"

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:AfishaApp.Pages"

mc:Ignorable="d"

Background="White"

d:DesignHeight="600" d:DesignWidth="800" FontSize="14"

Title="Добавление и редактирование" >

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="200"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Rectangle Grid.Row="0" Grid.ColumnSpan="2" />

<!--public int EventId { get; set; }

public string EventName { get; set; }

public string Info { get; set; }

public System.DateTime EventDate { get; set; }

public string EventPlace { get; set; }

public int Duration { get; set; }

public string Photo { get; set; }

public int StatusId { get; set; }

public int CategoryId { get; set; }-->

<TextBlock x:Name="TextBlockExhibitId" Grid.Row="0" Grid.Column="0" Text="ID:" Margin="20 0"/>

<TextBox x:Name="TextBoxExhibitId" Text="{Binding EventId}" Grid.Row="0" Grid.Column="1" Padding="0,2" IsReadOnly="True"/>

<TextBlock Grid.Row="1" Grid.Column="0" Text="Название:" Margin="20 0"/>

<TextBox x:Name="TextBoxExhibitName" Text="{Binding EventName}" Grid.Row="1" Grid.Column="1" Padding="0,2"/>

<TextBlock Grid.Row="2" Grid.Column="0" Text="Информация:" Margin="20 0"/>

<TextBox Grid.Row="2" Grid.Column="1" x:Name="TextBoxInfo"

Text="{Binding Info}" TextWrapping="Wrap" Height="100" />

<TextBlock Grid.Row="3" Grid.Column="0" Text="Дата:" Margin="20 0"/>

<DatePicker x:Name="TextBoxPhone" SelectedDate="{Binding EventDate}"

Grid.Row="3" Grid.Column="1" Padding="0,2"/>

<TextBlock Grid.Row="4" Grid.Column="0" Text="Место проведения:" Margin="20 0"/>

<TextBox x:Name="TextBoxAddress" Text="{Binding EventPlace}"

Grid.Row="4" Grid.Column="1" Padding="0,2"/>

<TextBlock Grid.Row="5" Grid.Column="0" Text="Длительность" Margin="20 0"/>

<TextBox x:Name="TextBoxCount" Text="{Binding Duration}" Width="140" HorizontalAlignment="Left"

Grid.Row="5" Grid.Column="1" Padding="0,2"/>

<TextBlock Grid.Row="6" Grid.Column="0" Text="Категории" Margin="20 0"/>

<ComboBox x:Name="ComboCategory" SelectedItem="{Binding Category}"

Grid.Row="6" Grid.Column="1" DisplayMemberPath="CategoryName"

SelectedValuePath="CategoryId" />

<TextBlock Grid.Row="7" Grid.Column="0" Text="Статус" Margin="20 0"/>

<ComboBox x:Name="ComboStatus" SelectedItem="{Binding Status}"

Grid.Row="7" Grid.Column="1" DisplayMemberPath="StatusName"

SelectedValuePath="StatusId" />

<TextBlock Grid.Row="8" Grid.Column="0" Text="Изображение:"

Margin="20 0" Height="30" VerticalAlignment="Top"/>

<Button x:Name="BtnSave" Grid.Row="9" Content="Сохранить"

HorizontalAlignment="Center" Margin="140,5,340,0" VerticalAlignment="Top" Width="120"

Click="BtnSaveClick" Grid.Column="1"/>

<Image x:Name="ImagePhoto" Source="{Binding GetPhoto}" Grid.Row="8" Grid.Column="1"

VerticalAlignment="Stretch" HorizontalAlignment="Stretch"/>

<Button x:Name="BtnLoad" Grid.Row="8" Grid.Column="0" Margin="20,30,0,0"

HorizontalAlignment="Left" VerticalAlignment="Top"

Click="BtnLoadClick" Content="Загрузить"/>

</Grid>

</Page>

**Файл программного кода**

using AfishaApp.Models;

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 AfishaApp.Pages;

using System.IO;

using Microsoft.Win32;

namespace AfishaApp.Pages

{

/// <summary>

/// Логика взаимодействия для AddEventPage.xaml

/// </summary>

public partial class AddEventPage : Page

{

//текущий товар

private EventTable \_currentExhibit = new EventTable();

// путь к файлу

private string \_filePath = null;

// название текущей главной фотографии

private string \_photoName = null;

// текущая папка приложения

private static string \_currentDirectory = Directory.GetCurrentDirectory() + @"\Images\";

// передача в AddExhibitPage товара

public AddEventPage(EventTable selectedExhibit)

{

InitializeComponent();

// если передано null, то мы добавляем новый товар

if (selectedExhibit != null)

{

\_currentExhibit = selectedExhibit;

TextBoxExhibitId.Visibility = Visibility.Hidden;

TextBlockExhibitId.Visibility = Visibility.Hidden;

int x = selectedExhibit.EventId;

List<EventTable> goods = new List<EventTable>();

\_filePath = \_currentDirectory + \_currentExhibit.Photo;

}

DataContext = \_currentExhibit;

\_photoName = \_currentExhibit.Photo;

//загрузка производителей

ComboCategory.ItemsSource = AfishaBDEntities.GetContext().Categories.ToList();

ComboStatus.ItemsSource = AfishaBDEntities.GetContext().Status.ToList();

}

// проверка полей

private StringBuilder CheckFields()

{

StringBuilder s = new StringBuilder();

if (string.IsNullOrWhiteSpace(\_currentExhibit.EventName))

s.AppendLine("Поле название пустое");

if (\_currentExhibit.Category == null)

s.AppendLine("Выберите производителя");

if (string.IsNullOrWhiteSpace(\_photoName))

s.AppendLine("фото не выбрано пустое");

return s;

}

// сохранение

private void BtnSaveClick(object sender, RoutedEventArgs e)

{

StringBuilder \_error = CheckFields();

// если ошибки есть, то выводим ошибки в MessageBox

// и прерываем выполнение

if (\_error.Length > 0)

{

MessageBox.Show(\_error.ToString());

return;

}

// проверка полей прошла успешно

if (\_currentExhibit.EventId == 0)

{

// добавление нового товара

// формируем новое название файла картинки,

// так как в папке может быть файл с тем же именем

string photo = ChangePhotoName();

// путь куда нужно скопировать файл

string dest = \_currentDirectory + photo;

File.Copy(\_filePath, dest);

\_currentExhibit.Photo = photo;

// добавляем товар в БД

AfishaBDEntities.GetContext().EventTables.Add(\_currentExhibit);

}

try

{

if (\_filePath != null)

{

string photo = ChangePhotoName();

string dest = \_currentDirectory + photo;

File.Copy(\_filePath, dest);

\_currentExhibit.Photo = photo;

}

// Сохраняем изменения в БД

AfishaBDEntities.GetContext().SaveChanges();

MessageBox.Show("Запись Изменена");

// Возвращаемся на предыдущую форму

Manager.MainFrame.GoBack();

}

catch (Exception ex)

{

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

}

}

// загрузка фото

private void BtnLoadClick(object sender, RoutedEventArgs e)

{

try

{

//Диалог открытия файла

OpenFileDialog op = new OpenFileDialog();

op.Title = "Select a picture";

op.Filter = "JPEG Files (\*.jpeg)|\*.jpeg|PNG Files (\*.png)|\*.png|JPG Files (\*.jpg)|\*.jpg|GIF Files (\*.gif)|\*.gif";

// диалог вернет true, если файл был открыт

if (op.ShowDialog() == true)

{

// проверка размера файла

// по условию файл дожен быть не более 2Мб.

FileInfo fileInfo = new FileInfo(op.FileName);

if (fileInfo.Length > (1024 \* 1024 \* 2))

{

// размер файла меньше 2Мб. Поэтому выбрасывается новое исключение

throw new Exception("Размер файла должен быть меньше 2Мб");

}

ImagePhoto.Source = new BitmapImage(new Uri(op.FileName));

\_photoName = op.SafeFileName;

\_filePath = op.FileName;

}

}

catch (Exception ex)

{

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

\_filePath = null;

}

}

//подбор имени файла

string ChangePhotoName()

{

string x = \_currentDirectory + \_photoName;

string photoname = \_photoName;

int i = 0;

if (File.Exists(x))

{

while (File.Exists(x))

{

i++;

x = \_currentDirectory + i.ToString() + photoname;

}

photoname = i.ToString() + photoname;

}

return photoname;

}

}

}

**Страница Категории**

**Файл интерфейса**

<Page x:Class="AfishaApp.Pages.CategoryPage"

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:AfishaApp.Pages"

mc:Ignorable="d"

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

Title="Категории">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<DataGrid Grid.Row="0" x:Name="DtStatus" HorizontalAlignment="Stretch" VerticalAlignment="Stretch"

LoadingRow="DataGridGoodLoadingRow"

AutoGenerateColumns="False" SelectionMode="Single" AlternationCount="1">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding CategoryID}" Header = "ID" Visibility="Hidden" />

<DataGridTextColumn Binding="{Binding CategoryName}" Header = "Категория" Width="3\*">

<DataGridTextColumn.ElementStyle>

<Style>

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

<Setter Property="TextBlock.TextAlignment" Value="Justify"/>

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

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

<Button x:Name="btnSave" Content="Сохранить" Click="BtnSave\_Click" />

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

</StackPanel>

</Grid>

</Page>

**Файл программного кода**

using AfishaApp.Models;

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;

namespace AfishaApp.Pages

{

/// <summary>

/// Логика взаимодействия для CategoryPage.xaml

/// </summary>

public partial class CategoryPage : Page

{

public CategoryPage()

{

InitializeComponent();

LoadData();

}

void LoadData()

{

AfishaBDEntities.GetContext().Categories.Load();

DtStatus.ItemsSource = AfishaBDEntities.GetContext().Categories.Local.ToBindingList();

}

void SaveData()

{

try

{

AfishaBDEntities.GetContext().SaveChanges();

MessageBox.Show("Сохранено");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoadData();

}

}

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

{

SaveData();

}

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

{

var selectedService = DtStatus.SelectedItems.Cast<Category>().ToList();

MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить { selectedService.Count()} записей ??? ", "Удаление", MessageBoxButton.OKCancel,

MessageBoxImage.Question);

if (messageBoxResult == MessageBoxResult.OK)

{

try

{

AfishaBDEntities.GetContext().Categories.RemoveRange(selectedService);

AfishaBDEntities.GetContext().SaveChanges();

MessageBox.Show("Записи удалены");

}

catch (Exception ex)

{

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

}

finally

{

LoadData();

}

}

}

// отображение номеров строк в DataGrid

private void DataGridGoodLoadingRow(object sender, DataGridRowEventArgs e)

{

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

}

}

}

**Страница Статусы**

**Файл интерфейса**

<Page x:Class="AfishaApp.Pages.StatusPage"

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:AfishaApp.Pages"

mc:Ignorable="d"

xmlns:xctk1="http://schemas.xceed.com/wpf/xaml/toolkit"

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

Title="Статусы">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<DataGrid Grid.Row="0" x:Name="DtStatus" HorizontalAlignment="Stretch" VerticalAlignment="Stretch"

LoadingRow="DataGridGoodLoadingRow"

AutoGenerateColumns="False" SelectionMode="Single" AlternationCount="1">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding StatusID}" Header = "ID" Visibility="Hidden" />

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

<DataGridTemplateColumn Width="200">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<xctk1:ColorPicker x:Name="colorStatus" SelectedColor="{Binding Color, UpdateSourceTrigger=PropertyChanged}"

DisplayColorAndName="True" AvailableColorsSortingMode="HueSaturationBrightness" UsingAlphaChannel="False"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

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

<Button x:Name="btnSave" Content="Сохранить" Click="BtnSave\_Click" />

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

</StackPanel>

</Grid>

</Page>

**Файл программного кода**

using AfishaApp.Models;

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;

namespace AfishaApp.Pages

{

/// <summary>

/// Логика взаимодействия для StatusPage.xaml

/// </summary>

public partial class StatusPage : Page

{

public StatusPage()

{

InitializeComponent();

LoadData();

}

void LoadData()

{

AfishaBDEntities.GetContext().Status.Load();

DtStatus.ItemsSource = AfishaBDEntities.GetContext().Status.Local.ToBindingList();

}

void SaveData()

{

try

{

AfishaBDEntities.GetContext().SaveChanges();

MessageBox.Show("Сохранено");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoadData();

}

}

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

{

SaveData();

}

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

{

var selectedService = DtStatus.SelectedItems.Cast<Status>().ToList();

MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить { selectedService.Count()} записей ??? ", "Удаление", MessageBoxButton.OKCancel,

MessageBoxImage.Question);

if (messageBoxResult == MessageBoxResult.OK)

{

try

{

AfishaBDEntities.GetContext().Status.RemoveRange(selectedService);

AfishaBDEntities.GetContext().SaveChanges();

MessageBox.Show("Записи удалены");

}

catch (Exception ex)

{

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

}

finally

{

LoadData();

}

}

}

// отображение номеров строк в DataGrid

private void DataGridGoodLoadingRow(object sender, DataGridRowEventArgs e)

{

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

}

}

}

**Файл программного кода**

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 MuseumApp.Models;

using System.Data.Entity;

namespace MuseumApp.Pages

{

/// <summary>

/// Логика взаимодействия для CategoryPage.xaml

/// </summary>

public partial class CategoryPage : Page

{

public CategoryPage()

{

InitializeComponent();

LoadData();

}

void LoadData()

{

MuseumDBEntities.GetContext().Categories.Load();

DtStatus.ItemsSource = MuseumDBEntities.GetContext().Categories.Local.ToBindingList();

}

void SaveData()

{

try

{

MuseumDBEntities.GetContext().SaveChanges();

MessageBox.Show("Сохранено");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoadData();

}

}

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

{

SaveData();

}

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

{

var selectedService = DtStatus.SelectedItems.Cast<Category>().ToList();

MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить { selectedService.Count()} записей ??? ", "Удаление", MessageBoxButton.OKCancel,

MessageBoxImage.Question);

if (messageBoxResult == MessageBoxResult.OK)

{

try

{

MuseumDBEntities.GetContext().Categories.RemoveRange(selectedService);

MuseumDBEntities.GetContext().SaveChanges();

MessageBox.Show("Записи удалены");

}

catch (Exception ex)

{

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

}

finally

{

LoadData();

}

}

}

// отображение номеров строк в DataGrid

private void DataGridGoodLoadingRow(object sender, DataGridRowEventArgs e)

{

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

}

}

}}

**Приложение B**

**Диск с исполняемым кодом программного модуля**