**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 16](#_Toc8739507)

[3.6 Разработка интерфейса ПС 20](#_Toc8739508)

[3.7 Описание структуры выходной информации 25](#_Toc8739509)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 26](#_Toc8739510)

[4.1 Назначение программного средства 26](#_Toc8739511)

[4.2 Условия выполнения программного средства 26](#_Toc8739512)

[4.3 Эксплуатация программного средства 26](#_Toc8739513)

[4.4 Сообщения пользователю 32](#_Toc8739514)

[5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА 34](#_Toc8739515)

[5.1 Определение верхней границы количества тестов 34](#_Toc8739516)

[5.2 Разработка тестовых вариантов 35](#_Toc8739517)

[5.3 Составление отчета о тестировании 42](#_Toc8739518)

[ЗАКЛЮЧЕНИЕ 44](#_Toc8739519)

[Оценка качества программного средства с помощью метрик 44](#_Toc8739520)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc8739521)

[Приложение А Текст кода программного средства 49](#_Toc8739522)

[Приложение Б Спецификация 79](#_Toc8739523)

[Приложение B Диск с программным средством 81](#_Toc8739524)

**Аннотация**

На дипломную работу студента Угарова Д.А. на тему: База данных «Фитнес-клуб S-Fitnes». Дипломная работа представлена пояснительной запиской на 81 лист машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В главе 1 содержится общие сведения о программном средстве Фитнес-клуб S-Fitnes. 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 – Автоматизированная информационная система «Фитнес-клуб S-Fitnes», 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 базы данных «Фитнес-клуб S-Fitnes», который позволяет автоматизировать деятельность фитнес-клуба по учёту продаж абонементов и учёту посещаемости.

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.

**1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ**

**1.1 Основное функциональное назначение программного средства**

Программа предназначена автоматизации деятельности фитнес-клуба S-Fitnes. 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 обработки данных Автоматизированная информационная система «Фитнес-клуб S-Fitnes».

**1.3 Условное обозначение программного средства**

«S-Fitnes 1.01»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 205 Угаров Д.А.

**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 «S-Fitnes 1.01» работает под управление операционной системы Windows 7, Windows 8, Windows 10. Программа предназначена для автоматизации деятельности фитнес-клуба по учёту абонементов и посещаемости клиентами. Программа позволяет вводить, изменять и удалять данные о клиентах, абонементах.

**2.3 Требование к программному средству**

«S-Fitnes 1.01» 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 «S-Fitnes 1.01» является кроссплатформенным продуктом, что позволяет не ограничиваться в работе с определенной версией ОС.

«S-Fitnes 1.01» написана на объектно-ориентированном языке программировании C# (си-шарп) в интегрированной среде разработки Visual Studio 2015. 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.

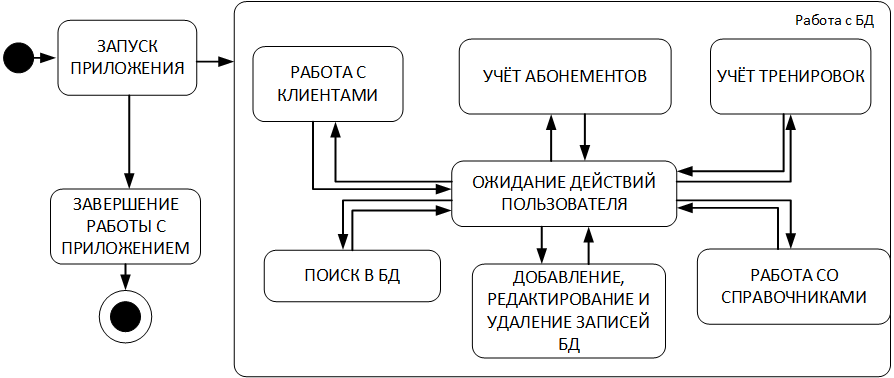


Рис.1 Декомпозиция поставленной задачи

**3.2 Общая архитектура программного средства**

Объектом моделирования является Фитнес-клуб S-Fitnes.

Рассматриваемые процессы: учёт абонементов и тренировок.

Объекты моделирования представлены на диаграмме классов, рис.2.



Рис.2 Диаграмма классов

По полученной декомпозиции задач была спроектирована следующая архитектура программного средства.

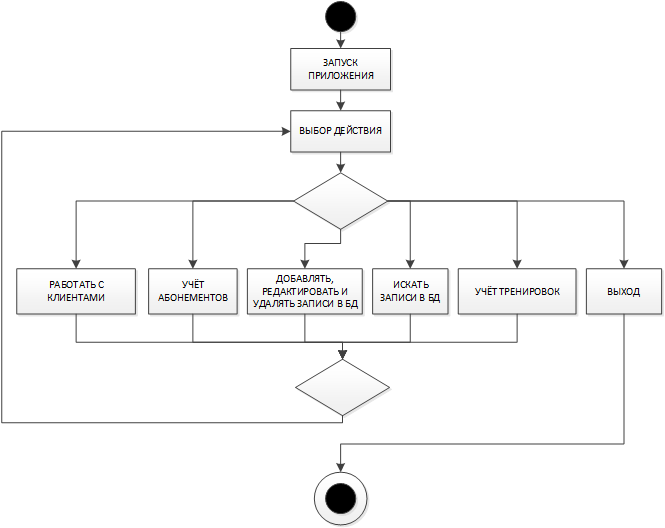


Рис.3 Схема событийно-ориентированной архитектуры

В «S-Fitnes 1.01» 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 «S-Fitnes 1.01» имеет следующий набор входных данных, такие как: клиенты, тренера, абонементы, тренировки.

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

Выходными данными являются: учёт продаж абонементов, учет тренировок.

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 Фитнес-клуб S-Fitnes» 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 «Тренировка». Реляционная модель автоматизированной системы соответствует всем 12 правилам Кодда. (рис.6)

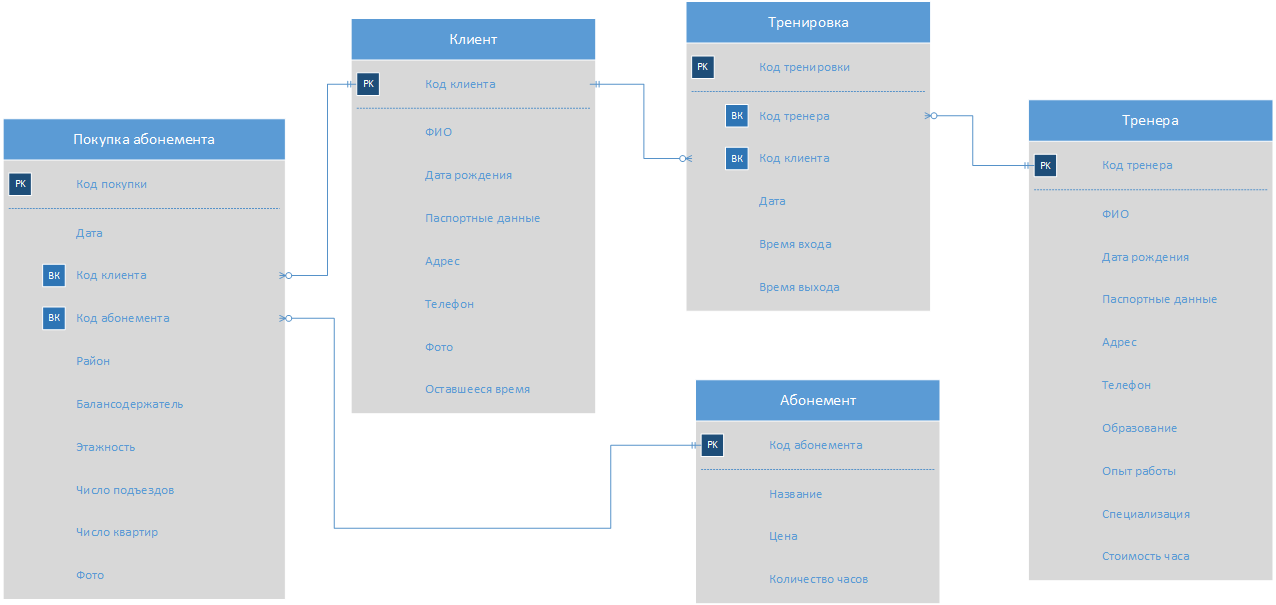


Рис.6 Реляционная модель

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\_покупки (таблица Покупка абонемента), Код\_абонемента (Таблица Абонементы), Код\_тренера (Таблица Тренеры), Код\_клиента (таблица Клиенты), Код\_тренировки (таблица Тренировки). 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 “tVisit” – “ID\_visit”. Связь таблиц “tVisit” и “tWorker” по полю “ID\_worker” один ко многим (1:М). Таблицы “tVisit” и “tCustomer” связаны по полю “ID\_customer” типом связи 1:М. Таблицы “tBuy” и “tCustomer” связаны по полю “ID\_customer” типом связи 1:М. Таблицы “tBuy” и “tAbonement” связаны по полю “ID\_abonement” типом связи 1:М. Все связи между таблицами базы данных типа один ко многим.

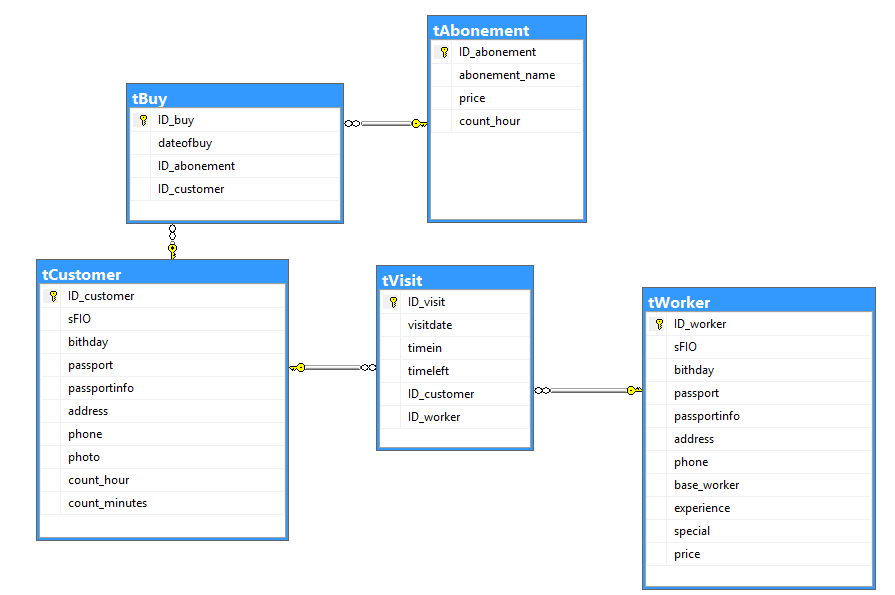


Рис.7 Схема базы данных

Таблица “tAbonement”(Абонементы)

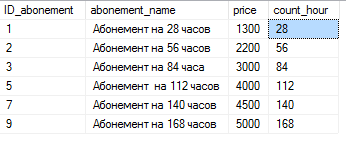


Рис.8 Таблица tAbonement

Таблица “tWorker”(Тренеры)

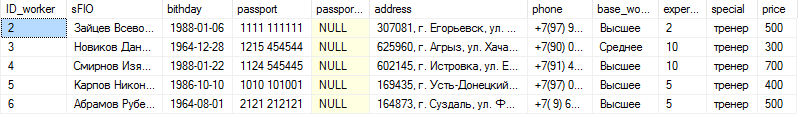


Рис.9 Таблица tWorker

Таблица “tBuy”(Покупка абонементов)

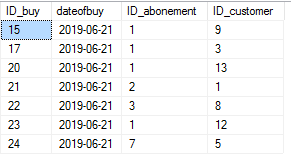


Рис.10 Таблица tBuy

Таблица “tVisit”(Тренировки)

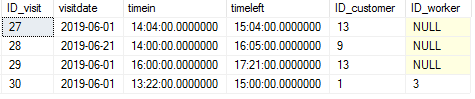


Рис.11 Таблица tVisit

Таблица “tCustomer”(Клиенты)

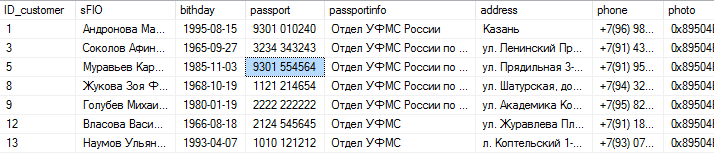


Рис.12 Таблица tCustomer

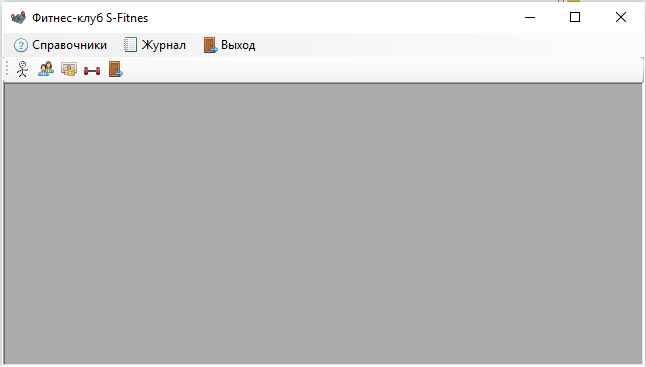
**3.6 Разработка интерфейса ПС**

Приложение однопользовательское, поэтому не предусмотрена авторизация. Для запуска приложения достаточно щелкнуть дважды левой кнопкой мыши по ярлыку.

После запуска программы пользователь приступит к работе с главной формой программы, на которой и будут отображаться все события в базе данных. «S-Fitnes 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 Фитнес-клуб S-Fitnes**. Основная форма приложения. Компоненты формы: menuStrip и toolStrip– для перехода к другим формам приложениия. (Рис.13)



Выход

Меню

форма Покупка абонемента

форма Клиенты

форма

Тренера

форма Тренировки

Рис.13 Фитнес-клуб S-Fitnes

**Форма Абонемент**. Интерфейс данной формы состоит из компонентов: DataGridView, BindingNavigator и ряда кнопок для удаления, сохранения и обновления данных. Также есть возможность поиска Абонемента. (Рис.14)

Кнопки навигации по записям

Удалить, обновить, сохранить

Поиск

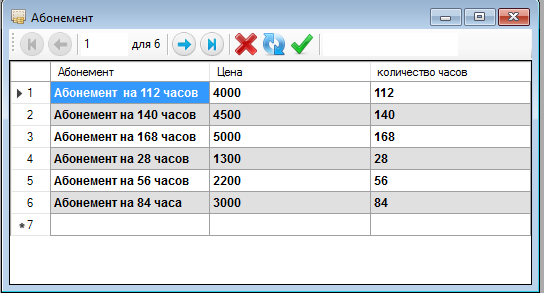


Рис.14 Окно Абонементы

**Форма Тренера**. Интерфейс данной формы состоит из компонентов: DataGridView, BindingNavigator и ряда кнопок.(Рис.15)

Кнопки навигации по записям

Удалить, обновить, сохранить

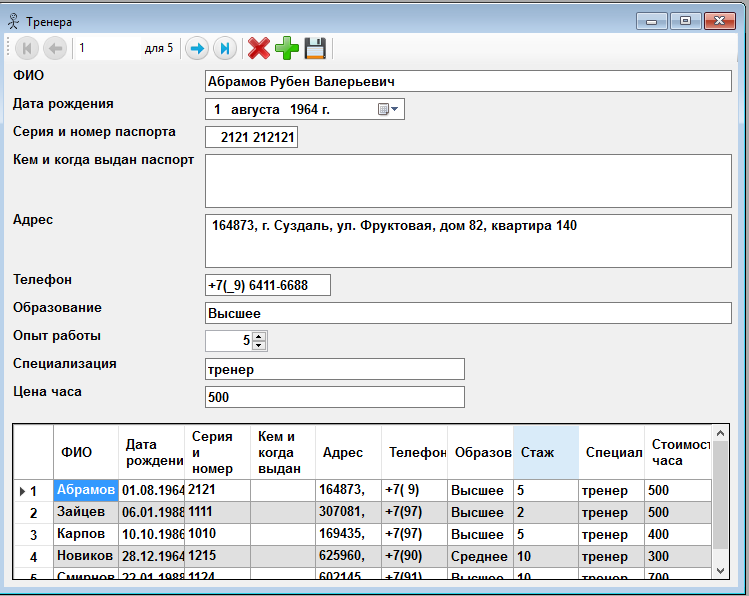


Рис.15 Окно Тренера

**Форма Клиенты**. Интерфейс данной формы состоит из компонентов: DataGridView, BindingNavigator, PictureBox и ряда кнопок.(Рис.16)

Кнопки навигации по записям

Удалить, обновить, сохранить

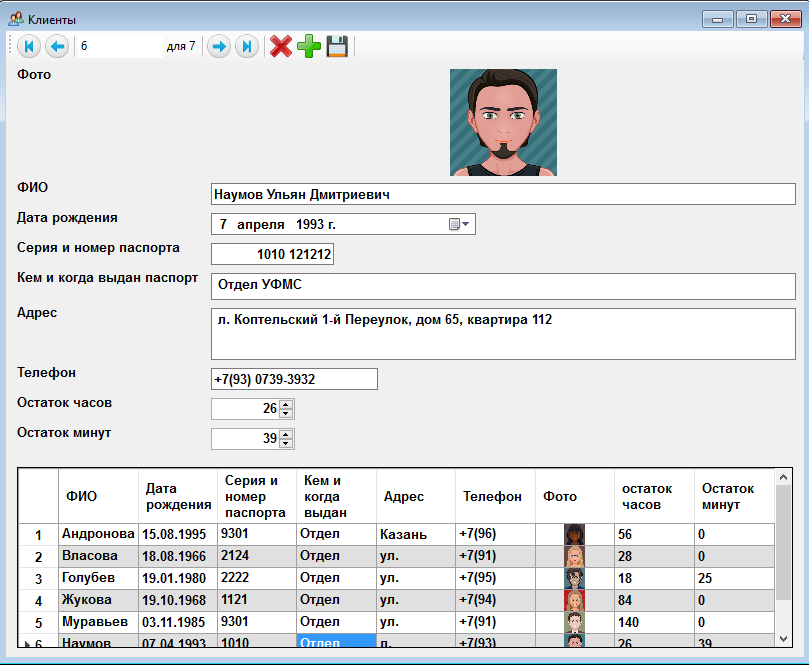


Рис.16 Окно Клиенты

**Форма Покупка абонемента**. Интерфейс данной формы состоит из компонентов: DataGridView, BindingNavigator, Textbox, Combobox, checkbox и label и ряда кнопок для удаления и вызова дополнительной формы. (Рис.17)

Кнопки навигации по записям

Удалить

Список продаж

вывод в Excel

Открыть доп. форму

поиск и фильтрация

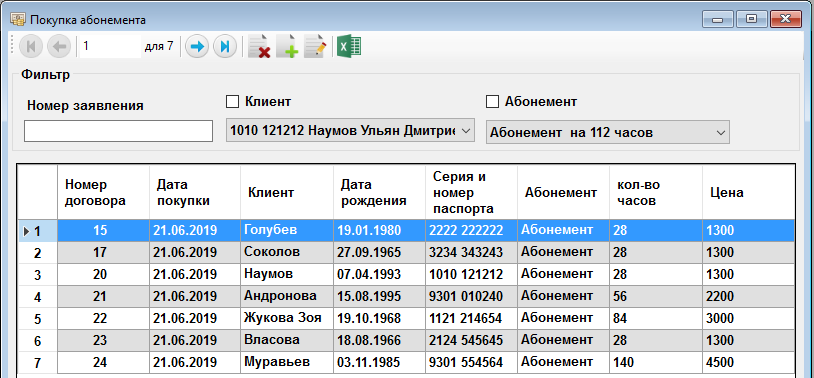
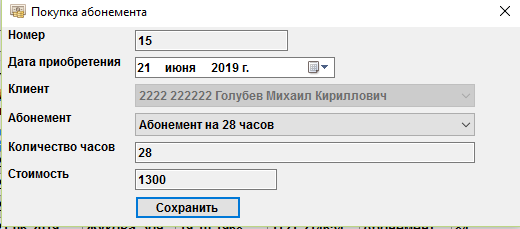


Рис.17 Окно Покупка абонемента

**Форма Покупка абонемента.** Интерфейс данной формы состоит из компонентов: label – отображают статический текст, textbox – отображают записи из таблицы Покупка абонемента, combobox – выводит список, PictureBox – изображение, а также кнопка для сохранения информации (Рис.18)



Сохранить

ComboBox

фото здания

Textbox номер

Рис.18 Окно Покупка абонемента

**Форма Тренировки**. Интерфейс данной формы состоит из компонентов: DataGridView, BindingNavigator и label и ряда кнопок для удаления и вызова дополнительной формы. (Рис.19)

Кнопки навигации по записям

Удалить

Список квартир

вывод в Excel

Открыть доп. форму

поиск по клиенту

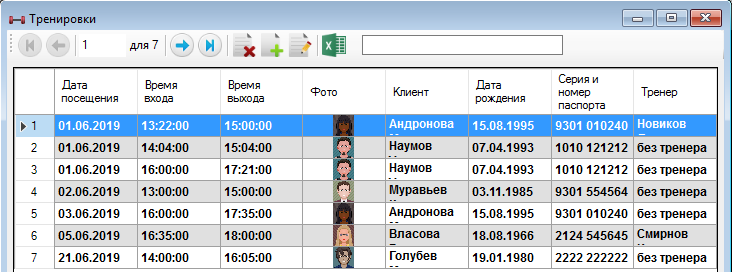
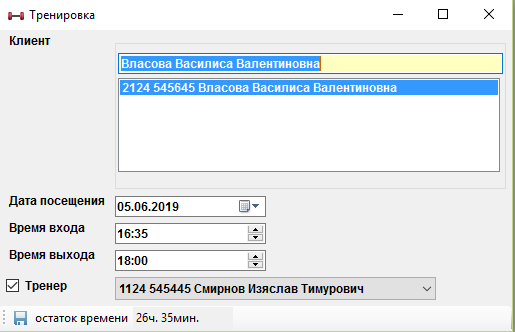


Рис.19 Окно Тренировки

**Форма Тренировка.** Интерфейс данной формы состоит из компонентов: label – отображают статический текст, textbox – отображают записи из таблицы Тренировки, combobox – выводит список, а также кнопки (Рис.20)



Textbox Клиент

datetimepicker – дата и время посещения

listbox список клиентов

Сохранить

Combobox – список тренеров

Рис.20 Окно Тренировка

**3.7 Описание структуры выходной информации**

Выходными данными являются списки тренировок и продаж абонементов.

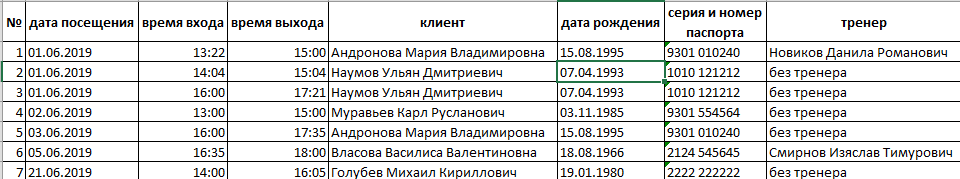


Рис.21 Список всех тренировок

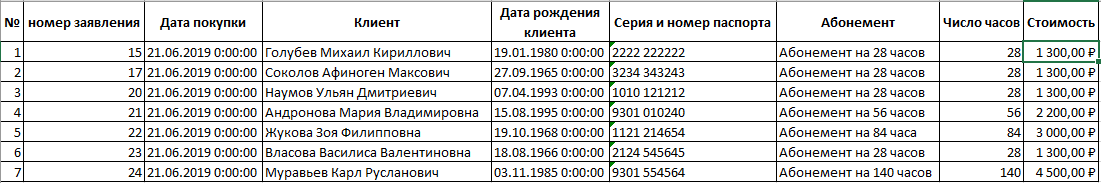


Рис.22 Список всех проаддных абонементов

**4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**4.1 Назначение программного средства**

Назначением программного средства является автоматизация учета продажи абонементов и учёт тренировок.

**4.2 Условия выполнения программного средства**

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

-Компьютер с установленной на нём «S-Fitnes 1.01».

Подготовительные действия:

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

**4.3 Эксплуатация программного средства**

Этот раздел содержит обзор приложений «S-Fitnes 1.01» и ее основных параметров.

***Активация продукта***

Активировать копию программы S-Fitnes 1.01 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

В главном окне приложения, расположены 4 кнопки с переходом на основные формы приложения и кнопка выхода. Дополнительные формы открываются через соответствующие пункты меню. Пункты меню также дублируют функционал кнопок. Пользователь выбирает нужную кнопку и открывается интересующая его форма. (Рис.23)

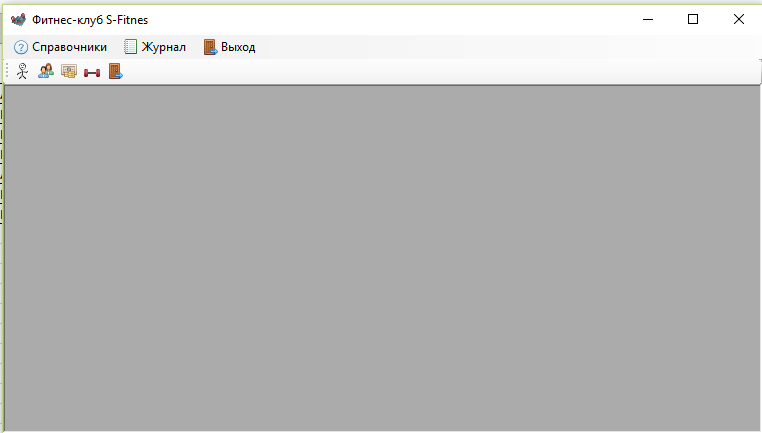


Рис.23 Окно Фитнес-клуб S-Fitnes

Кликнув по кнопке «Выход» программа потребует подтверждение. (Рис.24)

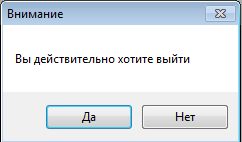


Рис.24 Сообщение программы

Ниже описаны пункты меню управления и информации.

«Абонемент»: эта форма предназначена для ввода, удаления, редактирования и поиска данных в таблице Абонементы. (Рис.25)

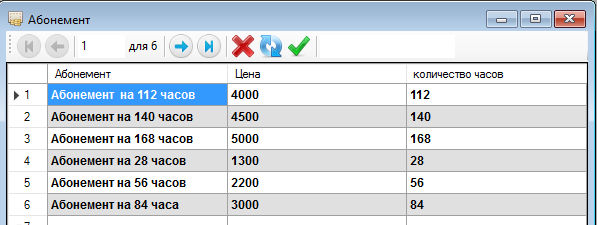


Рис.25 Окно Абонементы

«Клиенты»: эта форма предназначена для ввода, удаления, редактирования и поиска данных в таблице Клиенты. (Рис.26)

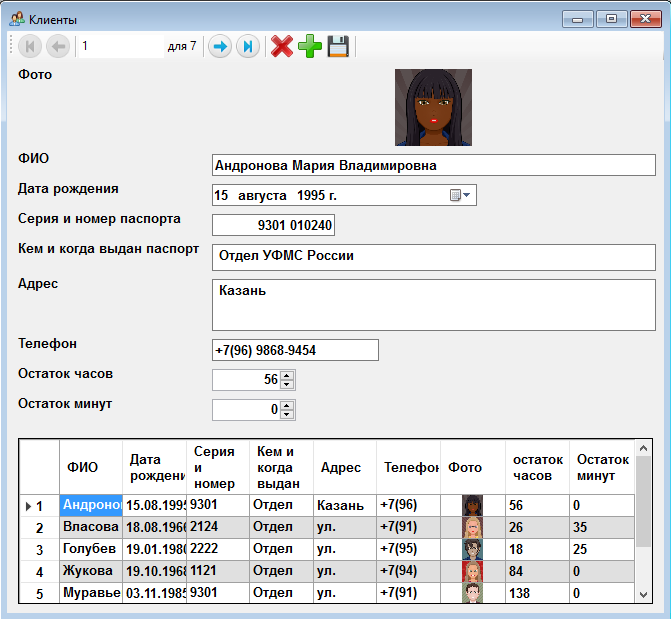


Рис.26 Окно Клиенты

«Тренера»: эта форма предназначена для ввода, удаления, редактирования и поиска данных в таблице Тренера. (Рис.27)

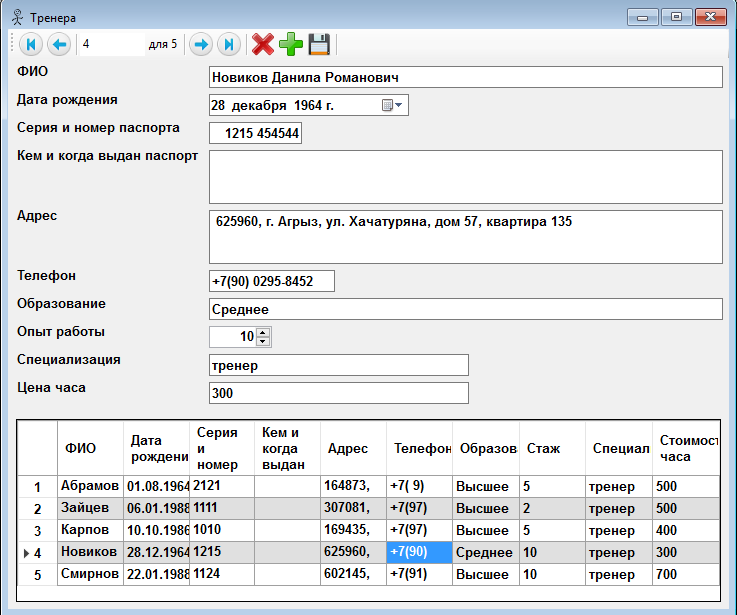


Рис.27 Окно Тренера

«Покупка абонемента»: одна из основных форм приложения. Эта форма позволяет вывести список отфильтрованных данных в Excel. Кроме этого также реализована возможность удаления записи. Для того чтобы добавить новое запись нужно нажать кнопку Добавить. Для того чтобы отредактировать запись нужно либо нажать на кнопку редактировать, предварительно выбрав нужную запись, либо дважды щелкнуть по выбранной записи.(Рис.28)

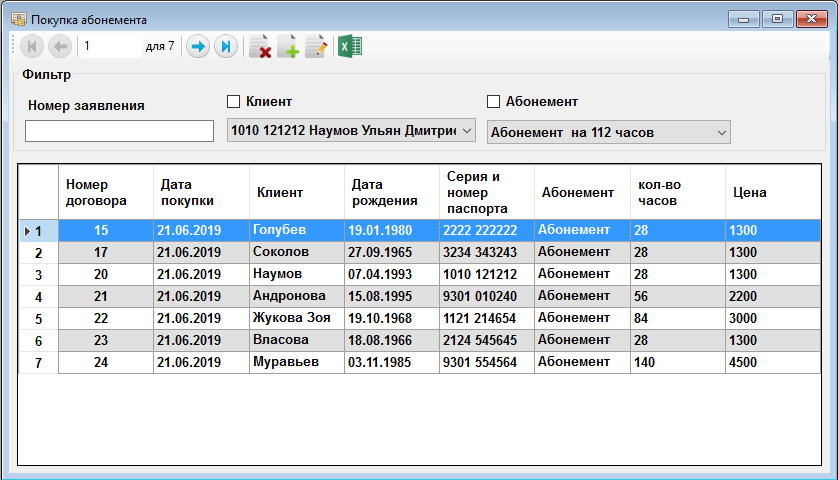


Рис.28 Окно Покупка абонемента

«Покупка абонемента»: форма предназначена для формирования новой записи или редактирования существующей. (Рис.29)

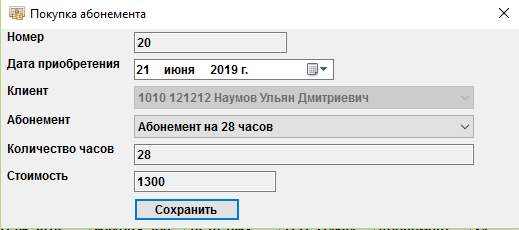


Рис.29 Окно Покупка абонемента

«Тренировки»: форма позволяет вывести список тренировок в Excel. Кроме этого также реализована возможность удаления. Для того чтобы добавить новую запись нужно нажать кнопку Добавить. Для того чтобы отредактировать запись нужно либо нажать на кнопку Редактировать, предварительно выбрав нужную запись, либо дважды щелкнуть по выбранной записи.(Рис.30)

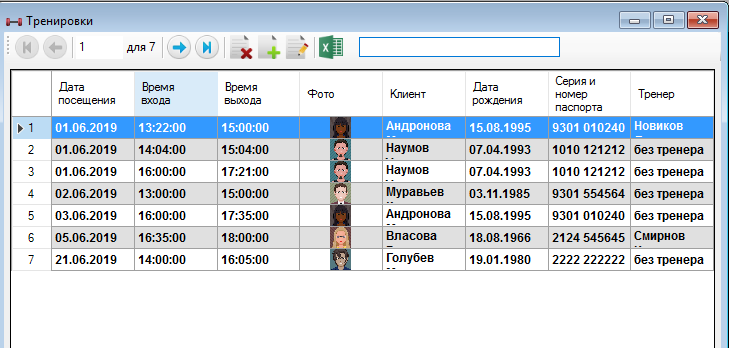


Рис.30 Окно Тренировки

«Тренировка»: форма предназначена для формирования новой записи или редактирования существующей. (Рис.31)

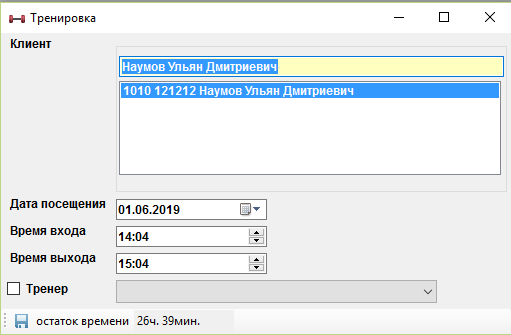


Рис.31 Окно Тренировка

Работа с S-Fitnes 1.01

После запуска программы Вы перейдете на главную форму программы. В формах «Абонементы», «Клиенты», «Тренера», «Покупка абонемента», «Тренировки» 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 S-Fitnes 1.01 кликните на соответствующей кнопке «Выход».

Основные кнопки S-Fitnes 1.01

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.32)



Рис.32 Сообщение Подтверждение удаления

После добавления новой записи, пользователю выводится сообщение. (Рис.33)



Рис.33 Сообщение Запись добавлена

После обновления записи, пользователю выводится сообщение. (Рис.34)



Рис.34 Сообщение Запись обновлена

При попытке удалить связанные записи, программа выводит пользователю сообщение о невозможности действия. (Рис.35)



Рис.35 Сообщение Ошибка удаления

**5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА**

**5.1 Определение верхней границы количества тестов**

Для разработанного алгоритма был построен граф.

****

Рис.36 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 S-Fitnes 1.01 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 Разработка тестовых вариантов**

Таблица 2 – Тестирование корректного добавления информации в Абонементы

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Абонементы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Абонементы | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Абонементы = на 112 часов  Цена = 4000 рублей  Количество часов = 112 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Абонементы открыта | | Отображается форма Абонементы |  |

Таблица 3 – Тестирование некорректного добавления информации в таблицу Абонементы

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Абонементы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Абонементы | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Абонементы =  Цена = 4000 рублей  Количество часов = 112 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, так как поле пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Абонементы открыта | | Отображается форма Абонементы |  |

Таблица 4 – Тестирование корректного редактирования информации в таблице Абонементы

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Абонементы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Абонементы | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Абонементы = на 112 часов  Цена = 4000 рублей  Количество часов = 112 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Абонементы  открыта | | Отображается форма Абонементы |  |

Таблица 5 – Тестирование некорректного редактирования информации в таблице Абонементы

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Абонементы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Абонементы | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Абонементы =  Цена = 4000 рублей  Количество часов = 112 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Абонементы  открыта | | Отображается форма Абонементы |  |

Таблица 6 – Тестирование корректного добавления информации в таблицу Клиент

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Клиент | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Клиент | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Фото= загружено  ФИО = Андронова Мария  Дата рождения = 15.08.2019  Паспорт = 1245 454645  Адрес = Казань, телефон = 8964574575, Остаток часов =56, Остаток минут = 35 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Клиент  открыта | | Отображается форма Клиент |  |

Таблица 7 – Тестирование некорректного добавления информации в таблицу Клиент

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Клиент | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Клиент | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Фото= загружено  ФИО =  Дата рождения = 15.08.2019  Паспорт = 1245 454645  Адрес = Казань, телефон = 8964574575, Остаток часов =56, Остаток минут = 35 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Клиент  открыта | | Отображается форма Клиент |  |

Таблица 8 – Тестирование корректного редактирования информации в таблице Клиент

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Клиент | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Клиент | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Фото= загружено  ФИО = Андронова Мария  Дата рождения = 15.08.2019  Паспорт = 1245 454645  Адрес = Казань, телефон = 8964574575, Остаток часов =56, Остаток минут = 35 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Клиент  открыта | | Отображается форма Клиент |  |

Таблица 9 – Тестирование некорректного редактирования информации в таблице Клиент

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Клиент | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Клиент | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Фото= загружено  ФИО =  Дата рождения = 15.08.2019  Паспорт = 1245 454645  Адрес = Казань, телефон = 8964574575, Остаток часов =56, Остаток минут = 35 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Клиент  открыта | | Отображается форма Клиент |  |

Таблица 10 – Тестирование корректного добавления информации в таблицу Тренер

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Тренер | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Тренер | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  ФИО = Абрамов Рубен  Дата рождения = 15.08.2019  Паспорт = 1245 454645  Адрес = Казань,  телефон = 8964574575,  Образование =Высшее, Стоимость занятия = 500 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Тренер  открыта | | Отображается форма Тренер |  |

Таблица 11 – Тестирование некорректного добавления информации в таблицу Тренер

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Тренер | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Тренер | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  ФИО =  Дата рождения = 15.08.2019  Паспорт = 1245 454645  Адрес = Казань,  телефон = 8964574575,  Образование =Высшее, Стоимость занятия = 500 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Тренер  открыта | | Отображается форма Тренер |  |

Таблица 12 – Тестирование корректного редактирования информации в таблице Тренер

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Тренер | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Тренер | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  ФИО = Абрамов Рубен  Дата рождения = 15.08.2019  Паспорт = 1245 454645  Адрес = Казань,  телефон = 8964574575,  Образование =Высшее, Стоимость занятия = 500 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Тренер  открыта | | Отображается форма Тренер |  |

Таблица 13 – Тестирование некорректного редактирования информации в таблице Тренер

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Тренер | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Тренер | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  ФИО =  Дата рождения = 15.08.2019  Паспорт = 1245 454645  Адрес = Казань,  телефон = 8964574575,  Образование =Высшее, Стоимость занятия = 500 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поля пустые | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Тренер  открыта | | Отображается форма Тренер |  |

**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 |
| Главная (Фитнес-клуб S-Fitnes) | 4 | 4 | | 0 |
| Абонементы | 4 | 4 | | 0 |
| Тренера | 4 | 4 | | 0 |
| Клиенты | 4 | 4 | | 0 |
| Покупка абонементов | 4 | 4 | | 0 |
| Тренировки | 4 | 4 | | 0 |
| ИТОГО | 24 | 24 | 0 | |

По результатам тестирования были выявлены и устранены следующие проблемы:

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

**Ошибка**



Рис.37 Ошибка теста №8

**Причина**

В поле количество часов было введено вещественное число. В программном коде при сохранении предусматривалось, что число будет целочисленным.

**Принятые меры**

Ошибка устранена. Программный код был адаптирован под вещественные данные.

При выполнении теста таблица 6. Тестирование формы «Клиент», Некорректный запрос

**Ошибка**

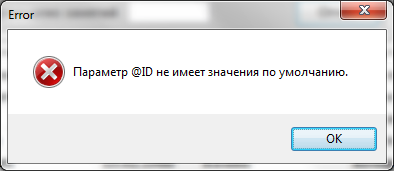


Рис. 38 Ошибка теста №6

**Причина**

Ввод данных поиска символами. Посылался запрос в таблицу Клиент на поиск клиента с = «!М». Но такого клиента нет.

**Принятые меры**

Код изменен. В поле поиска можно вводить любые символы и поиск корректно работает.

**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 7 Ultimate 32бит.

Процессор: Intel Pentium G2030 3.0 GHz.

ОЗУ: 2 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 | 9 | x1 | x2 | **x3** | =27 |
| Отчет | 2 | x2 | **x5** | x8 | =10 |

Таблица 18 – Сложность экрана

|  |  |  |  |
| --- | --- | --- | --- |
| Экраны | Количество Тренерских (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 Тренерских (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

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Г. Шилдт C#: Учебный курс. - Спб.: Питер, 2021. - 512с

2. Либерти, Д. Программирование на Си Шарп. - М.: Символ-плюс, 2022. - 684 с.

3. Ватсон К. Си Шарп/К.Ватсон.-М.:Лори, 2021.-862 с

4. Балена Ф., Димауро Д. Современная практика программирования на Visual Basic и Visual C#.-М.:Русская редакция, 2022,604с.

5. Енин А., Енин Н. Локальная СУБД своими руками. Учимся на примерах.-М.:СОЛОН-ПРЕСС, 2022,464с.,СД

6. Лабор В.В. Си Шарп: Создание приложений для Windows/-Мн.: Харвест, 2022-384с.

7. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2022-512с.

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#. 2022 г.

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

14. Никита Культин. Основы программирования в MicrosoftVisualC# 2022г.

[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, 2022.

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

16. Йен Гриффитс. Программирование на С#. 2022 г.

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

**Приложение А**

Автоматизированная информационная система «Фитнес-клуб S-Fitnes»

Текст кода программного средства

460.ЗМК.00027-01 12 01

**Текст кода программного средства**

**Форма Главная**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kvartirs

{

public partial class MainForm : Form

{

AbonementForm material\_form;

AllBuildingForm all\_building\_form;

AllFlats all\_flats;

private CustomerForm customerForm;

TrainerForm trenerForm;

public MainForm()

{

InitializeComponent();

}

private void mnuExit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void MainForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)

{

try

{

DialogResult result = MessageBox.Show("Вы действительно хотите выйти из приложения", "Внимание",

MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.No)

{

e.Cancel = true;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void mnuMaterial\_Click(object sender, EventArgs e)

{

if (material\_form == null || material\_form.IsDisposed)

{

material\_form = new AbonementForm();

material\_form.MdiParent = this;

material\_form.Show();

}

else

{

material\_form.Activate();

}

}

private void mnuBuilding\_Click(object sender, EventArgs e)

{

if (all\_building\_form == null || all\_building\_form.IsDisposed)

{

all\_building\_form = new AllBuildingForm();

all\_building\_form.MdiParent = this;

all\_building\_form.Show();

}

else

{

all\_building\_form.Activate();

}

}

private void mnuFlats\_Click(object sender, EventArgs e)

{

if (all\_flats == null || all\_flats.IsDisposed)

{

all\_flats = new AllFlats();

all\_flats.MdiParent = this;

all\_flats.Show();

}

else

{

all\_flats.Activate();

}

}

private void mnuCustomer\_Click(object sender, EventArgs e)

{

if (customerForm == null || customerForm.IsDisposed)

{

customerForm = new CustomerForm();

customerForm.MdiParent = this;

customerForm.Show();

}

else

{

customerForm.Activate();

}

}

private void mnuTrener\_Click(object sender, EventArgs e)

{

if (trenerForm == null || trenerForm.IsDisposed)

{

trenerForm = new TrainerForm();

trenerForm.MdiParent = this;

trenerForm.Show();

}

else

{

trenerForm.Activate();

} } }

}

**Форма Абонементы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

namespace Kvartirs

{

public partial class AbonementForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

DataSet ds;

SqlDataAdapter dataAdapter;

BindingSource bs1;

public AbonementForm()

{

InitializeComponent();

LoadDataFromTable();

}

private void saveToolStripButton\_Click(object sender, EventArgs e)

{

try

{

SqlCommandBuilder CmbSAve = new SqlCommandBuilder(dataAdapter);

dataAdapter.Update(ds);

saveToolStripButton.Enabled = false;

}

catch (Exception ex)

{

if (ex.HResult == -2146232060)

MessageBox.Show("Ошибка удаления, есть связанные записи");

else MessageBox.Show(ex.Message);

LoadDataFromTable();

}

}

void LoadDataFromTable()

{

try

{

ds = new DataSet();

dataAdapter = new SqlDataAdapter();

dataAdapter.SelectCommand = new SqlCommand("Select \* from tAbonement Order by abonement\_name", connection);

//dt1 = new DataTable();

ds.Clear();

dataAdapter.Fill(ds);

ds.AcceptChanges();

bs1 = new BindingSource();

bs1.DataSource = ds.Tables[0];

bindingNavigator1.BindingSource = bs1;

dgvTypeTO.DataSource = bs1;

dgvTypeTO.Columns[0].Visible = false;

dgvTypeTO.Columns[1].HeaderText = "Абонемент";

dgvTypeTO.Columns[2].HeaderText = "Цена";

dgvTypeTO.Columns[3].HeaderText = "количество часов";

dgvTypeTO.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void deleteStripButton2\_Click(object sender, EventArgs e)

{

try

{

int i = dgvTypeTO.CurrentRow.Index;

dgvTypeTO.Rows.RemoveAt(i);

saveToolStripButton.Enabled = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

LoadDataFromTable();

}

private void dgvTypeTO\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

saveToolStripButton.Enabled = true;

int i = dgvTypeTO.CurrentRow.Index;

}

private void dgvTypeTO\_UserDeletedRow(object sender, DataGridViewRowEventArgs e)

{

saveToolStripButton.Enabled = true;

}

private void dgvTypeTO\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

saveToolStripButton.Enabled = true;

}

private void toolStripTextBox1\_TextChanged(object sender, EventArgs e)

{

bs1.Filter = "abonement\_name LIKE '%" + toolStripTextBox1.Text + "%'";

}

private void dgvTypeTO\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{

// right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat);

} }}

**Форма Покупка абонементов**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

using System.Drawing.Imaging;

using System.IO;

using System.Windows.Forms.VisualStyles;

using Excel = Microsoft.Office.Interop.Excel;

namespace Kvartirs

{

public partial class AllBuildingForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

DataTable dtBuilding, dataTableMaterial, dataTableBase;

DataSet dsBuilding;

SqlDataAdapter dataAdapterBuilding;

BindingSource bsBuilding, bsMaterial, bsBase;

bool add\_items;

public AllBuildingForm()

{

InitializeComponent();

LoadDataFromTable();

loadComboAbonement();

loadComboCustomer();

}

void loadComboCustomer()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter dataAdapterMaterial = new SqlDataAdapter("SELECT ID\_customer, passport +' '+ sfio as sfio from tCustomer Order by sfio", connection);

dataTableMaterial = new DataTable();

dataAdapterMaterial.Fill(dataTableMaterial);

bsMaterial = new BindingSource();

bsMaterial.DataSource = dataTableMaterial;

cmbCustomer.DataSource = dataTableMaterial;

cmbCustomer.ValueMember = "ID\_customer";

cmbCustomer.DisplayMember = "sfio";

}

}

void loadComboAbonement()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter dataAdapterBase = new SqlDataAdapter("SELECT ID\_abonement, abonement\_name FROM tAbonement order by abonement\_name", connection);

dataTableBase = new DataTable();

dataAdapterBase.Fill(dataTableBase);

bsBase = new BindingSource();

bsBase.DataSource = dataTableBase;

cmbAbonement.DataSource = dataTableBase;

cmbAbonement.ValueMember = "ID\_abonement";

cmbAbonement.DisplayMember = "abonement\_name";

}

}

void LoadDataFromTable()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

dtBuilding = new DataTable();

dataAdapterBuilding = new SqlDataAdapter();

dataAdapterBuilding.SelectCommand = new SqlCommand(" SELECT rtrim(str(tBuy.ID\_buy)) as ID\_buy, tBuy.dateofbuy, "+

"tBuy.ID\_customer, tCustomer.sFIO, tCustomer.bithday, "+

" tCustomer.passport, tBuy.ID\_abonement, tAbonement.abonement\_name,"+

" tAbonement.count\_hour, tAbonement.price FROM tAbonement"+

" join(tbuy join tCustomer on tBuy.ID\_customer = tCustomer.ID\_customer)"+

" on tAbonement.ID\_abonement = tBuy.ID\_abonement", connection);

dataAdapterBuilding.Fill(dtBuilding);

bsBuilding = new BindingSource();

bsBuilding.DataSource = dtBuilding;

bindingNavigator1.BindingSource = bsBuilding;

dgvTypeTO.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dgvTypeTO.DataSource = bsBuilding;

dgvTypeTO.Columns[0].HeaderText = "Номер договора";

dgvTypeTO.Columns[1].HeaderText = "Дата покупки";

dgvTypeTO.Columns[2].Visible = false;

dgvTypeTO.Columns[3].HeaderText = "Клиент";

dgvTypeTO.Columns[4].HeaderText = "Дата рождения";

dgvTypeTO.Columns[5].HeaderText = "Серия и номер паспорта";

dgvTypeTO.Columns[6].Visible = false;

dgvTypeTO.Columns[7].HeaderText = "Абонемент";

dgvTypeTO.Columns[8].HeaderText = "кол-во часов";

dgvTypeTO.Columns[9].HeaderText = "Цена";

add\_items = false;

if (bsBuilding.Count <= 0) tsbDelete.Enabled = false;

else tsbDelete.Enabled = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

private void tsbAdd\_Click(object sender, EventArgs e)

{

BuildingForm building\_form = new BuildingForm();

building\_form.ShowDialog();

LoadDataFromTable();

}

void ShowBuildingToChange()

{

if ((bsBuilding.Count > 0) && (dgvTypeTO.SelectedRows.Count > 0))

{

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsBuilding.Current).Row["ID\_buy"]);

BuildingForm building\_form = new BuildingForm(ID\_SS);

building\_form.ShowDialog();

LoadDataFromTable();

bsBuilding.Position = bsBuilding.Find("ID\_buy", (ID\_SS));

}

}

private void tsbChange\_Click(object sender, EventArgs e)

{

ShowBuildingToChange();

}

private void dgvTypeTO\_DoubleClick(object sender, EventArgs e)

{

ShowBuildingToChange();

}

private void chbMaterial\_CheckedChanged(object sender, EventArgs e)

{

FilterData();

}

private void chbBase\_CheckedChanged(object sender, EventArgs e)

{

FilterData();

}

private void cmbMaterial\_SelectedIndexChanged(object sender, EventArgs e)

{

FilterData();

}

private void cmbBase\_SelectedIndexChanged(object sender, EventArgs e)

{

FilterData();

}

private void tsbDelete\_Click(object sender, EventArgs e)

{

if (bsBuilding.Count > 0)

{

int i = bsBuilding.Position;

string ID\_SS = Convert.ToString(((DataRowView)this.bsBuilding.Current).Row["ID\_buy"]);

int hours = Convert.ToInt32(((DataRowView)this.bsBuilding.Current).Row["count\_hour"]);

int id\_c = Convert.ToInt32(((DataRowView)this.bsBuilding.Current).Row["ID\_customer"]);

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

DialogResult result = MessageBox.Show("Вы действительно хотите удалить запись", "Внимание", MessageBoxButtons.YesNo);

if (result == DialogResult.No)

{

LoadDataFromTable();

return;

}

if (result == DialogResult.Yes)

{

SqlCommand commandDelete = new SqlCommand("Delete From tBuy where ID\_buy = @ID\_buy", connection);

commandDelete.Parameters.AddWithValue("@ID\_buy", ID\_SS);

commandDelete.ExecuteNonQuery();

UpdateHours(hours, id\_c);

}

}

catch (SqlException exception)

{

if (exception.HResult == -2146232060)

MessageBox.Show("Ошибка удаления, есть связанные записи");

else MessageBox.Show(exception.Message);

}

finally

{

LoadDataFromTable();

}

}

}

}

void UpdateHours(int x, int y)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand commandUpdate = new SqlCommand("UPDATE tCustomer SET" +

" count\_hour=count\_hour-@count\_hour " +

" WHERE ID\_customer= @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@count\_hour", x);

commandUpdate.Parameters.AddWithValue("@IDSS", y);

commandUpdate.ExecuteNonQuery();

// MessageBox.Show("Запись обновлена");

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

}

private void tsbExcel\_Click(object sender, EventArgs e)

{

PrintExcel();

}

private void PrintExcel()

{

string fileName = System.Windows.Forms.Application.StartupPath + "\\" + "Prodagi" + ".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 = "Продажи";

int row = 2;

int i = 0;

if (dgvTypeTO.RowCount > 0)

{

for (i = 0; i < dgvTypeTO.RowCount; i++)

{

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

xlSheet.Cells[row, 2] = dgvTypeTO.Rows[i].Cells[0].Value.ToString();

xlSheet.Cells[row, 3] = dgvTypeTO.Rows[i].Cells[1].Value.ToString();

xlSheet.Cells[row, 4] = dgvTypeTO.Rows[i].Cells[3].Value.ToString();

xlSheet.Cells[row, 5] = dgvTypeTO.Rows[i].Cells[4].Value.ToString();

xlSheet.Cells[row, 6] = dgvTypeTO.Rows[i].Cells[5].Value.ToString();

xlSheet.Cells[row, 7] = dgvTypeTO.Rows[i].Cells[7].Value.ToString();

xlSheet.Cells[row, 8] = dgvTypeTO.Rows[i].Cells[8].Value.ToString();

xlSheet.Cells[row, 9] = dgvTypeTO.Rows[i].Cells[9].Value.ToString();

row++;

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

r.Insert(Excel.XlInsertShiftDirection.xlShiftDown);

}

}

row--;

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

xlSheetRange.Borders.LineStyle = true;

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 tbKadastr\_TextChanged(object sender, EventArgs e)

{

FilterData();

}

void FilterData()

{

bsBuilding.RemoveFilter();

var queries = new List<string>();

if (chbMaterial.Checked)

{

queries.Add(string.Format("[ID\_customer]={0}", cmbCustomer.SelectedValue));

}

if (chbBase.Checked)

{

queries.Add(string.Format("[ID\_abonement]={0}", cmbAbonement.SelectedValue));

}

if (tbKadastr.Text != "")

{

queries.Add(string.Format("[ID\_buy] LIKE '%{0}%'", tbKadastr.Text));

}

if (queries.Count >= 1)

{

var queryFilter = String.Join(" AND ", queries);

bsBuilding.Filter = queryFilter;

}

}

private void dgvTypeTO\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{

// right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat);

}

}

}

**Форма Покупка Абонемента**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

using System.Drawing.Imaging;

using System.IO;

using System.Windows.Forms.VisualStyles;

using static Kvartirs.models;

namespace Kvartirs

{

public partial class BuildingForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

DataTable dtBuilding, dataTableMaterial, dataTableBase, dtGoods;

DataSet dsBuilding;

SqlDataAdapter dataAdapterBuilding;

BindingSource bsBuilding, bsMaterial, bsBase;

List <Abonement> ab = new List<Abonement>();

bool add\_items;

int hours = 0;

Buy build = null;

public int ID\_kadastr { get; set; }

private void BuildingForm\_Load(object sender, EventArgs e)

{

loadComboBase();

loadComboMaterial();

LoadDataFromTable();

}

public BuildingForm()

{

InitializeComponent();

add\_items = false;

}

public BuildingForm(int ID)

{

InitializeComponent();

ID\_kadastr = ID;

add\_items = true;

cmbCustomer.Enabled = false;

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if ((cmbAbonement.SelectedIndex == -1)

|| (cmbCustomer.SelectedIndex == -1))

{

MessageBox.Show("Ключевые поля пустые");

return;

}

if (add\_items == true)

{

UpdateData();

}

else

{

SaveData();

}

}

void LoadDataFromTable()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

SqlCommand SelectCommand = new SqlCommand(" SELECT ID\_buy, tBuy.dateofbuy, "+

"tBuy.ID\_customer, tCustomer.sFIO, " +

" tBuy.ID\_abonement, tAbonement.abonement\_name," +

" tAbonement.count\_hour, tAbonement.price FROM tAbonement" +

" join(tbuy join tCustomer on tBuy.ID\_customer = tCustomer.ID\_customer)" +

" on tAbonement.ID\_abonement = tBuy.ID\_abonement WHERE ID\_buy ='" + ID\_kadastr + "'", connection);

SqlDataReader reader = SelectCommand.ExecuteReader();

if (reader.HasRows)

{

reader.Read();

build = new Buy();

build.ID\_buy = Convert.ToInt32(reader.GetValue(0));

build.dateofbuy = Convert.ToDateTime(reader.GetValue(1));

build.ID\_customer = Convert.ToInt32(reader.GetValue(2));

build.customer\_name = Convert.ToString(reader.GetValue(3));

build.ID\_abonement= Convert.ToInt32(reader.GetValue(4));

build.abonement\_name = Convert.ToString(reader.GetValue(5));

build.count\_hour = Convert.ToInt32(reader.GetValue(6));

build.price = Convert.ToDouble(reader.GetValue(7));

tbIDbuy.Text = build.ID\_buy.ToString();

tbCount.Text = build.count\_hour.ToString();

tbPrice.Text = build.price.ToString();

hours = build.count\_hour;

cmbCustomer.SelectedValue = build.ID\_customer;

cmbAbonement.SelectedValue = build.ID\_abonement;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

if ((cmbAbonement.SelectedIndex == -1)

|| (cmbCustomer.SelectedIndex == -1))

{

MessageBox.Show("Ключевые поля пустые");

return;

}

if (add\_items == true)

{

int x = Convert.ToInt32(tbCount.Text)- hours;

MessageBox.Show(x.ToString());

UpdateHours(x);

UpdateData();

}

else

{

SaveData();

UpdateHours(Convert.ToInt32(tbCount.Text));

}

}

private void cmbAbonement\_SelectedIndexChanged(object sender, EventArgs e)

{

foreach (Abonement x in ab)

{

if (x.ID\_abonement == Convert.ToInt32(cmbAbonement.SelectedValue))

{

tbCount.Text = x.count\_hour.ToString();

tbPrice.Text = x.price.ToString();

}

}

//if (bsBase.Count > 0)

//{

// int i = bsBase.Find("ID\_abonement", Convert.ToString(cmbAbonement.SelectedValue));

// tbCount.Text = Convert.ToString(((DataRowView)this.bsBase[i]).Row["count\_hour"]);

//}

}

private void label4\_Click(object sender, EventArgs e)

{

}

void UpdateHours(int x)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand commandUpdate = new SqlCommand("UPDATE tCustomer SET" +

" count\_hour=count\_hour+@count\_hour " +

" WHERE ID\_customer= @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@count\_hour", x);

commandUpdate.Parameters.AddWithValue("@IDSS", Convert.ToInt32(cmbCustomer.SelectedValue));

commandUpdate.ExecuteNonQuery();

// MessageBox.Show("Запись обновлена");

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

}

void loadComboMaterial()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter dataAdapterMaterial = new SqlDataAdapter("SELECT ID\_customer, passport +' '+ sfio as sfio from tCustomer Order by sfio", connection);

dataTableMaterial = new DataTable();

dataAdapterMaterial.Fill(dataTableMaterial);

bsMaterial = new BindingSource();

bsMaterial.DataSource = dataTableMaterial;

cmbCustomer.DataSource = dataTableMaterial;

cmbCustomer.ValueMember = "ID\_customer";

cmbCustomer.DisplayMember = "sfio";

}

}

void loadComboBase()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter dataAdapterBase = new SqlDataAdapter("SELECT ID\_abonement, abonement\_name, price, count\_hour FROM tAbonement order by abonement\_name", connection);

dataTableBase = new DataTable();

dataAdapterBase.Fill(dataTableBase);

bsBase = new BindingSource();

bsBase.DataSource = dataTableBase;

cmbAbonement.DataSource = dataTableBase;

cmbAbonement.ValueMember = "ID\_abonement";

cmbAbonement.DisplayMember = "abonement\_name";

ab = new List<Abonement>();

for (int i = 0; i < bsBase.Count; i++)

{

Abonement x = new Abonement();

x.abonement\_name = Convert.ToString(((DataRowView)this.bsBase[i]).Row["abonement\_name"]);

x.ID\_abonement = Convert.ToInt32(((DataRowView)this.bsBase[i]).Row["ID\_abonement"]);

x.price = Convert.ToDouble(((DataRowView)this.bsBase[i]).Row["price"]);

x.count\_hour = Convert.ToInt32(((DataRowView)this.bsBase[i]).Row["count\_hour"]);

ab.Add(x);

}

}

}

void SaveData()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand commandInsert = new SqlCommand("INSERT INTO [tBuy] VALUES(" +

"@dateofbuy," +

"@ID\_abonement," +

"@ID\_customer" +

") ; SELECT SCOPE\_IDENTITY()", connection);

commandInsert.Parameters.AddWithValue("@dateofbuy", dtpDateofbuy.Value);

commandInsert.Parameters.AddWithValue("@ID\_abonement", Convert.ToInt32(cmbAbonement.SelectedValue));

commandInsert.Parameters.AddWithValue("@ID\_customer", Convert.ToInt32(cmbCustomer.SelectedValue));

ID\_kadastr = Convert.ToInt32(commandInsert.ExecuteScalar());

MessageBox.Show("Запись добавлена");

add\_items = true;

cmbCustomer.Enabled = false;

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

LoadDataFromTable();

}

}

void UpdateData()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand commandUpdate = new SqlCommand("UPDATE tBuy SET" +

" dateofbuy=@dateofbuy," +

"ID\_abonement=@ID\_abonement" +

// "ID\_customer=@ID\_customer " +

" WHERE ID\_buy= @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@dateofbuy", dtpDateofbuy.Value);

commandUpdate.Parameters.AddWithValue("@ID\_abonement", Convert.ToInt32(cmbAbonement.SelectedValue));

// commandUpdate.Parameters.AddWithValue("@ID\_customer", Convert.ToInt32(cmbCustomer.SelectedValue));

commandUpdate.Parameters.AddWithValue("@IDSS", build.ID\_buy);

commandUpdate.ExecuteNonQuery();

MessageBox.Show("Запись обновлена");

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

LoadDataFromTable();

} } }}

**Форма Тренер**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

namespace Kvartirs

{

public partial class TrainerForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

BindingSource bsPeople, bsEduc;

BindingSource bs2, bsPosts;

DataTable dt, dtChilds, dtPosts, dtEduc, dtBrigada;

private MaskedTextBox mtbData;

DataTable dtJob;

DataSet dsJob;

private BindingSource bsJob, bsBrigada;

SqlDataAdapter dataAdapter, dataAdapterPost;

private bool Jobless = false;

string IDSERIA = "";

string IDNOMER = "";

bool add\_items, b1, b2;

public TrainerForm()

{

InitializeComponent();

LoadDataFromTable();

}

void LoadDataFromTable()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

ClearItems();

connection.Open();

SqlDataAdapter sda = new SqlDataAdapter("SELECT \* FROM tWorker Order by sFIO, passport ", connection);

dt = new DataTable();

sda.Fill(dt);

bsPeople = new BindingSource();

//= dataTable;

bsPeople.DataSource = dt;

bindingNavigator1.BindingSource = bsPeople;

// Очистка

// textBox1

if (bsPeople.Count > 0) toolStripButton2.Enabled = true;

else toolStripButton2.Enabled = false;

tbSurName.DataBindings.Clear();

tbSurName.DataBindings.Add(new Binding("Text", bsPeople, "sFIO"));

dtpBithday.DataBindings.Clear();

dtpBithday.DataBindings.Add(new Binding("Value", bsPeople, "Bithday"));

mtbPassport.DataBindings.Clear();

mtbPassport.DataBindings.Add(new Binding("Text", bsPeople, "Passport"));

tbPassportinfo.DataBindings.Clear();

tbPassportinfo.DataBindings.Add(new Binding("Text", bsPeople, "Passportinfo"));

tbAdress.DataBindings.Clear();

tbAdress.DataBindings.Add(new Binding("Text", bsPeople, "Address"));

tbPhone.DataBindings.Clear();

tbPhone.DataBindings.Add(new Binding("Text", bsPeople, "Phone"));

tbBase\_worker.DataBindings.Clear();

tbBase\_worker.DataBindings.Add(new Binding("Text", bsPeople, "base\_worker"));

nudExperience.DataBindings.Clear();

nudExperience.DataBindings.Add(new Binding("Text", bsPeople, "experience"));

tbSpecial.DataBindings.Clear();

tbSpecial.DataBindings.Add(new Binding("Text", bsPeople, "special"));

tbPrice.DataBindings.Clear();

tbPrice.DataBindings.Add(new Binding("Text", bsPeople, "price"));

dgvTypeTO.DataSource = bsPeople;

dgvTypeTO.Columns[0].Visible = false;

dgvTypeTO.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dgvTypeTO.Columns[1].HeaderText = "ФИО";

dgvTypeTO.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgvTypeTO.Columns[2].HeaderText = "Дата рождения";

dgvTypeTO.Columns[3].HeaderText = "Серия и номер паспорта";

dgvTypeTO.Columns[4].HeaderText = "Кем и когда выдан";

dgvTypeTO.Columns[5].HeaderText = "Адрес";

dgvTypeTO.Columns[6].HeaderText = "Телефон";

dgvTypeTO.Columns[7].HeaderText = "Образование";

dgvTypeTO.Columns[8].HeaderText = "Стаж";

dgvTypeTO.Columns[9].HeaderText = "Специализация";

dgvTypeTO.Columns[10].HeaderText = "Стоимость часа";

add\_items = false;

}

}

private void tbPassport\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar >= '0' && e.KeyChar <= '9')

{

return;

}

if (Char.IsControl(e.KeyChar)) return;

e.Handled = true;

}

private void tbSurName\_KeyPress(object sender, KeyPressEventArgs e)

{

if (Char.IsLetter(e.KeyChar)) return;

if (Char.IsSeparator(e.KeyChar)) return;

if (Char.IsControl(e.KeyChar)) return;

e.Handled = true;

}

private void tbPhone\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar >= '0' && e.KeyChar <= '9') return;

if (Char.IsControl(e.KeyChar)) return;

e.Handled = true;

}

private void tbPrice\_KeyPress(object sender, KeyPressEventArgs e)

{

System.Windows.Forms.TextBox textBox = (System.Windows.Forms.TextBox)sender;

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

return;

if (e.KeyChar == '.')

e.KeyChar = ',';

if (e.KeyChar == ',')

{

// в поле редактирования не может

// быть больше одной запятой и запятая

// не может быть первым символом

if ((textBox.Text.IndexOf(',') != -1) ||

(textBox.Text.Length == 0))

{

e.Handled = true;

}

return;

}

if (Char.IsControl(e.KeyChar))

{

// <Enter>, <Backspace>, <Esc>

return;

}

// остальные символы запрещены

e.Handled = true;

}

private void ClearItems()

{

tbSurName.Text = "";

dtpBithday.Value = DateTime.Now;

mtbPassport.Text = "";

tbAdress.Text = "";

tbPhone.Text = "";

tbPassportinfo.Text = "";

tbSpecial.Text = "";

tbBase\_worker.Text = "";

tbPrice.Text = "0";

}

private void AddStripButton\_Click(object sender, EventArgs e)

{

ClearItems();

add\_items = true;

saveToolStripButton.Enabled = false;

}

private void toolStripButton2\_Click(object sender, EventArgs e)

{

if (add\_items)

{

add\_items = false;

saveToolStripButton.Enabled = true;

LoadDataFromTable();

}

else if (bsPeople.Count > 0)

{

int i = bsPeople.Position;

string ID\_SS = ((DataRowView)this.bsPeople.Current).Row["ID\_worker"].ToString();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

DialogResult result = MessageBox.Show("Вы действительно хотите удалить запись", "Внимание",

MessageBoxButtons.YesNo);

if (result == DialogResult.No)

{

LoadDataFromTable();

return;

}

if (result == DialogResult.Yes)

{

SqlCommand commandDelete = new SqlCommand("Delete From tWorker where ID\_worker = @ID\_worker", connection);

commandDelete.Parameters.AddWithValue("@ID\_worker", ID\_SS);

commandDelete.ExecuteNonQuery();

}

}

catch (SqlException exception)

{

if ((uint)exception.ErrorCode == 0x80004005)

MessageBox.Show("Удаление прервано, есть связанные записи", "Error", MessageBoxButtons.OK,

MessageBoxIcon.Error);

else

MessageBox.Show(exception.ToString());

}

finally

{

LoadDataFromTable();

}

}

}

}

void InsertData()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

SqlCommand commandInsert = new SqlCommand("INSERT INTO [tWorker] VALUES(" +

"@sFIO," +

"@bithday," +

"@passport," +

"@passportinfo," +

"@address," +

"@phone," +

"@base\_worker," +

"@experience," +

"@special," +

"@price" +

") ", connection);

commandInsert.Parameters.AddWithValue("@sFIO", tbSurName.Text);

commandInsert.Parameters.AddWithValue("@bithday", dtpBithday.Value.Date);

commandInsert.Parameters.AddWithValue("@passport", mtbPassport.Text);

commandInsert.Parameters.AddWithValue("@passportinfo", tbPassportinfo.Text);

commandInsert.Parameters.AddWithValue("@address", tbAdress.Text);

commandInsert.Parameters.AddWithValue("@phone", tbPhone.Text);

commandInsert.Parameters.AddWithValue("@base\_worker", tbBase\_worker.Text);

commandInsert.Parameters.AddWithValue("@experience", Convert.ToInt32(nudExperience.Value));

commandInsert.Parameters.AddWithValue("@special", tbSpecial.Text);

commandInsert.Parameters.AddWithValue("@price", tbPrice.Text);

commandInsert.ExecuteNonQuery();

MessageBox.Show("Запись добавлена");

add\_items = false;

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

connection.Close();

LoadDataFromTable();

}

}

}

void UpdateData()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string x = "";

connection.Open();

try

{

string ID\_SS = ((DataRowView)this.bsPeople.Current).Row["ID\_worker"].ToString();

x = ID\_SS;

SqlCommand commandUpdate = new SqlCommand("UPDATE tWorker SET" +

" sFIO=@sFIO," +

"bithday=@bithday," +

"passport=@passport," +

"passportinfo=@passportinfo," +

"address=@address," +

"phone=@phone," +

"base\_worker=@base\_worker," +

"experience=@experience," +

"special=@special," +

"price=@price " +

" WHERE ID\_Worker= @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@sFIO", tbSurName.Text);

commandUpdate.Parameters.AddWithValue("@bithday", dtpBithday.Value.Date);

commandUpdate.Parameters.AddWithValue("@passport", mtbPassport.Text);

commandUpdate.Parameters.AddWithValue("@passportinfo", tbPassportinfo.Text);

commandUpdate.Parameters.AddWithValue("@address", tbAdress.Text);

commandUpdate.Parameters.AddWithValue("@phone", tbPhone.Text);

commandUpdate.Parameters.AddWithValue("@base\_worker", tbBase\_worker.Text);

commandUpdate.Parameters.AddWithValue("@experience", Convert.ToInt32(nudExperience.Value));

commandUpdate.Parameters.AddWithValue("@special", tbSpecial.Text);

commandUpdate.Parameters.AddWithValue("@price", tbPrice.Text);

commandUpdate.Parameters.AddWithValue("@IDSS", ID\_SS);

commandUpdate.ExecuteNonQuery();

MessageBox.Show("Запись обновлена");

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

connection.Close();

LoadDataFromTable();

bsPeople.Position = bsPeople.Find("ID\_worker", x);

}

}

}

private void saveToolStripButton\_Click(object sender, EventArgs e)

{

if ((mtbPassport.Text == "") || (tbSurName.Text == "") || (tbPrice.Text == ""))

{

MessageBox.Show("Ключевые поля пустые", "error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (add\_items)

{

InsertData();

// UpdateData();

}

else

{

UpdateData();

}

}

private void tbPassport\_TextChanged(object sender, EventArgs e)

{

if (mtbPassport.Text == "" || tbSurName.Text == "") saveToolStripButton.Enabled = false;

else saveToolStripButton.Enabled = true;

}

private void dgvTypeTO\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{

// right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat);

}

private void dgGrid\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{

// right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat); } }}

**Форма Тренировки**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

using System.Drawing.Imaging;

using System.IO;

using System.Windows.Forms.VisualStyles;

using Excel = Microsoft.Office.Interop.Excel;

namespace Kvartirs

{

public partial class AllFlats : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

DataTable dtFlat, dataTableMaterial, dataTableBase;

DataSet dsFlat;

SqlDataAdapter dataAdapterFlat;

BindingSource bsFlat, bsMaterial, bsBase;

private void tsbChange\_Click(object sender, EventArgs e)

{

ShowBuildingToChange();

}

void ShowBuildingToChange()

{

if ((bsFlat.Count > 0) && (dgvTypeTO.SelectedRows.Count > 0))

{

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsFlat.Current).Row["ID\_visit"]);

FlatForm flat\_form = new FlatForm(ID\_SS);

flat\_form.ShowDialog();

LoadDataFromTable();

bsFlat.Position = bsFlat.Find("ID\_visit", (ID\_SS));

}

}

private void dgvTypeTO\_DoubleClick(object sender, EventArgs e)

{

ShowBuildingToChange();

}

private void tbKadastr\_TextChanged(object sender, EventArgs e)

{

bsFlat.RemoveFilter();

bsFlat.Filter = string.Format("[sFIO] LIKE '%{0}%'", tbKadastr.Text);

}

int GetMinutes()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsFlat.Current).Row["ID\_customer"]);

SqlCommand commandSelect = new SqlCommand("SELECT (count\_hour\*60+ count\_minutes) as minutes " +

"FROM tCustomer " +

" WHERE ID\_customer= @IDSS", connection);

commandSelect.Parameters.AddWithValue("@IDSS", ID\_SS);

int x = Convert.ToInt32(commandSelect.ExecuteScalar());

// MessageBox.Show(x.ToString());

return x;

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

return 0;

}

void UpdateHours(int x)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsFlat.Current).Row["ID\_customer"]);

SqlCommand commandUpdate = new SqlCommand("UPDATE tCustomer SET" +

" count\_hour=@count\_hour, count\_minutes=@count\_minutes " +

" WHERE ID\_customer= @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@count\_hour", x / 60);

commandUpdate.Parameters.AddWithValue("@count\_minutes", x % 60);

commandUpdate.Parameters.AddWithValue("@IDSS", Convert.ToInt32(ID\_SS));

commandUpdate.ExecuteNonQuery();

// MessageBox.Show("Запись обновлена");

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

}

private void tsbDelete\_Click(object sender, EventArgs e)

{

if (bsFlat.Count > 0)

{

int i = bsFlat.Position;

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsFlat.Current).Row["ID\_visit"]);

TimeSpan x = TimeSpan.Parse((((DataRowView)this.bsFlat.Current).Row["timein"]).ToString());

TimeSpan y = TimeSpan.Parse((((DataRowView)this.bsFlat.Current).Row["timeleft"]).ToString());

// DateTime y = Convert.ToDateTime(((DataRowView)this.bsFlat.Current).Row["timeleft"]);

int z = Convert.ToInt32(y.Subtract(x).TotalMinutes);

//MessageBox.Show(z.ToString());

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

DialogResult result = MessageBox.Show("Вы действительно хотите удалить запись", "Внимание", MessageBoxButtons.YesNo);

if (result == DialogResult.No)

{

LoadDataFromTable();

return;

}

if (result == DialogResult.Yes)

{

SqlCommand commandDelete = new SqlCommand("Delete From tVisit where ID\_visit = @ID\_visit", connection);

commandDelete.Parameters.AddWithValue("@ID\_visit", ID\_SS);

commandDelete.ExecuteNonQuery();

UpdateHours(GetMinutes()+z);

}

}

catch (SqlException exception)

{

if (exception.HResult == -2146232060)

MessageBox.Show("Ошибка удаления, есть связанные записи");

else MessageBox.Show(exception.Message);

}

finally

{

LoadDataFromTable();

}

}

}

}

private void tsbAdd\_Click(object sender, EventArgs e)

{

FlatForm flat\_form = new FlatForm();

flat\_form.ShowDialog();

LoadDataFromTable();

}

private void tsbExcel\_Click(object sender, EventArgs e)

{

PrintExcel();

}

private void PrintExcel()

{

string fileName = System.Windows.Forms.Application.StartupPath + "\\" + "Visit" + ".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 = "Квартиры";

int row = 2;

int i = 0;

if (dgvTypeTO.RowCount > 0)

{

for (i = 0; i < dgvTypeTO.RowCount; i++)

{

string x = "";

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

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

xlSheet.Cells[row, 2] = y.Date.ToShortDateString();

xlSheet.Cells[row, 3] = dgvTypeTO.Rows[i].Cells[2].Value.ToString();

xlSheet.Cells[row, 4] = dgvTypeTO.Rows[i].Cells[3].Value.ToString();

xlSheet.Cells[row, 5] = dgvTypeTO.Rows[i].Cells[6].Value.ToString();

y = Convert.ToDateTime(dgvTypeTO.Rows[i].Cells[7].Value);

xlSheet.Cells[row, 6] = y.Date.ToShortDateString();

xlSheet.Cells[row, 7] = dgvTypeTO.Rows[i].Cells[8].Value.ToString();

xlSheet.Cells[row, 8] = dgvTypeTO.Rows[i].Cells[10].Value.ToString();

row++;

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

r.Insert(Excel.XlInsertShiftDirection.xlShiftDown);

}

}

row--;

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

xlSheetRange.Borders.LineStyle = true;

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;

}

}

bool add\_items;

public AllFlats()

{

InitializeComponent();

LoadDataFromTable();

}

void LoadDataFromTable()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

dtFlat = new DataTable();

dataAdapterFlat = new SqlDataAdapter();

dataAdapterFlat.SelectCommand = new SqlCommand(" SELECT tVisit.ID\_visit, tVisit.visitdate,"+

" tVisit.timein, tVisit.timeleft, tCustomer.Photo, tVisit.ID\_customer, "+

" tCustomer.sFIO, tCustomer.bithday, tCustomer.passport,"+

" ISNULL(tVisit.ID\_worker, 0) ID\_worker, ISNULL(tWorker.sFIO, 'без тренера') Worker" +

" FROM tWorker RIGHT join(tVisit join tCustomer on tVisit.ID\_customer = tCustomer.ID\_customer) " +

" on tVisit.ID\_worker = tWorker.ID\_worker Order by tVisit.visitdate, tVisit.timein, tVisit.timeleft", connection);

dataAdapterFlat.Fill(dtFlat);

bsFlat = new BindingSource();

bsFlat.DataSource = dtFlat;

bindingNavigator1.BindingSource = bsFlat;

dgvTypeTO.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dgvTypeTO.DataSource = bsFlat;

dgvTypeTO.Columns[0].Visible = false;

dgvTypeTO.Columns[1].HeaderText = "Дата посещения";

dgvTypeTO.Columns[2].HeaderText = "Время входа";

dgvTypeTO.Columns[3].HeaderText = "Время выхода";

dgvTypeTO.Columns[4].HeaderText = "Фото";

dgvTypeTO.Columns[5].Visible = false;

dgvTypeTO.Columns[6].HeaderText ="Клиент";

dgvTypeTO.Columns[7].HeaderText = "Дата рождения";

dgvTypeTO.Columns[8].HeaderText = "Серия и номер паспорта";

dgvTypeTO.Columns[10].HeaderText = "Тренер";

dgvTypeTO.Columns[9].Visible = false;

((DataGridViewImageColumn)dgvTypeTO.Columns[4]).ImageLayout = DataGridViewImageCellLayout.Zoom;

add\_items = false;

if (bsFlat.Count <= 0) tsbDelete.Enabled = false;

else tsbDelete.Enabled = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

private void dgvTypeTO\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{

// right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat); } }

}

**Форма Тренировка**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

using System.Drawing.Imaging;

using System.IO;

using System.Windows.Forms.VisualStyles;

using static Kvartirs.models;

namespace Kvartirs

{

public partial class FlatForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

DataTable dtFlat, dtRaschet, dtTarif, dtBuildings, dtCustomer;

DataSet dsFlat;

Visit visit = null;

SqlDataAdapter dataAdapterFlat, dataAdapterRaschet;

BindingSource bsVisitBindingSource, bsBuilding, bsTarif, bsCustomer, bsRaschet;

bool add\_items;

bool caninto;

public int ID\_visit { get; set; }

public int ID\_customer { get; set; }

private string fio;

public FlatForm()

{

InitializeComponent();

ID\_visit = -1;

ID\_customer = -1;

add\_items = false;

chbTrener.Checked = false;

cmbWorker.Enabled = false;

loadListBox();

loadComboCustomer();

cmbWorker.SelectedIndex = -1;

}

private void chbTrener\_CheckedChanged(object sender, EventArgs e)

{

if (chbTrener.Checked)

{

cmbWorker.Enabled = true;

}

else

{

cmbWorker.Enabled = false;

cmbWorker.SelectedIndex = -1;

}

}

public FlatForm(int ID)

{

InitializeComponent();

ID\_visit = ID;

ID\_customer = -1;

add\_items = true;

loadListBox();

loadComboCustomer();

}

private void tbCustomer\_TextChanged(object sender, EventArgs e)

{

bsBuilding.RemoveFilter();

bsBuilding.Filter = string.Format("[customer] LIKE '%{0}%'", tbCustomer.Text);

}

private void lstCustomer\_SelectedValueChanged(object sender, EventArgs e)

{

}

void loadListBox()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter dtaBuildings = new SqlDataAdapter("SELECT ID\_customer, passport + ' ' + LTRIM(RTRIM(sFIO)) as customer, sFiO from tCustomer Order by passport, sFIO", connection);

dtBuildings = new DataTable();

dtaBuildings.Fill(dtBuildings);

bsBuilding = new BindingSource();

bsBuilding.DataSource = dtBuildings;

lstCustomer.DataSource = bsBuilding;

lstCustomer.ValueMember = "ID\_customer";

lstCustomer.DisplayMember = "customer";

}

}

private void dtpTimeLeft\_ValueChanged(object sender, EventArgs e)

{

if (dtpTimeLeft.Value.TimeOfDay < dtpTimeIn.Value.TimeOfDay)

dtpTimeLeft.Value = dtpTimeIn.Value;

}

private void dtpTimeIn\_ValueChanged(object sender, EventArgs e)

{

if (dtpTimeLeft.Value.TimeOfDay < dtpTimeIn.Value.TimeOfDay)

dtpTimeLeft.Value = dtpTimeIn.Value;

}

void loadComboCustomer()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter dataAdapterCustomer = new SqlDataAdapter("SELECT ID\_worker, passport + ' ' + sFIO as worker from tWorker Order by passport, sFIO", connection);

dtCustomer = new DataTable();

dataAdapterCustomer.Fill(dtCustomer);

bsCustomer = new BindingSource();

bsCustomer.DataSource = dtCustomer;

cmbWorker.DataSource = bsCustomer;

cmbWorker.ValueMember = "ID\_worker";

cmbWorker.DisplayMember = "worker";

}

}

void LoadDataFromTable()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

SqlCommand SelectCommand = new SqlCommand(" SELECT tVisit.ID\_visit, tVisit.visitdate, "+

" tVisit.timein, tVisit.timeleft, tVisit.ID\_customer, " +

" tCustomer.sFIO," +

" ISNULL(tVisit.ID\_worker, 0) ID\_worker, ISNULL(tWorker.sFIO, 'без тренера') Worker" +

" FROM tWorker RIGHT join(tVisit join tCustomer on tVisit.ID\_customer = tCustomer.ID\_customer) " +

" on tVisit.ID\_worker = tWorker.ID\_worker WHERE ID\_visit = " + ID\_visit+ "Order by tVisit.visitdate, tVisit.timein, tVisit.timeleft", connection);

SqlDataReader reader = SelectCommand.ExecuteReader();

if (reader.HasRows)

{

reader.Read();

visit = new Visit();

visit.ID\_visit = Convert.ToInt32(reader.GetValue(0));

visit.visitdate = Convert.ToDateTime(reader.GetValue(1));

TimeSpan myTimeSpan = ((SqlDataReader)reader).GetTimeSpan(reader.GetOrdinal("timein"));

visit.timein = new DateTime(myTimeSpan.Ticks);

// visit.timein = Convert.ToDateTime(reader.GetValue(2));

//visit.timeleft = Convert.ToDateTime(reader.GetValue(3));

myTimeSpan = ((SqlDataReader)reader).GetTimeSpan(reader.GetOrdinal("timeleft"));

visit.timeleft = new DateTime(myTimeSpan.Ticks);

visit.ID\_customer = Convert.ToInt32(reader.GetValue(4));

visit.customer\_name = Convert.ToString(reader.GetValue(5));

visit.ID\_worker = Convert.ToInt32(reader.GetValue(6));

visit.worker\_name = Convert.ToString(reader.GetValue(7));

if (visit.ID\_worker != 0)

chbTrener.Checked = true;

else

{

chbTrener.Checked = false;

}

tbCustomer.Text = visit.customer\_name;

dtpDate.Value = visit.visitdate;

dtpTimeIn.Value = dtpDate.Value.Add(visit.timein.TimeOfDay);

dtpTimeLeft.Value = dtpDate.Value.Add(visit.timeleft.TimeOfDay);

cmbWorker.SelectedValue = visit.ID\_worker;

ID\_customer = visit.ID\_customer;

tbTimeLeft.Text = String.Format("{0}ч. {1}мин.", GetMinutes() / 60, GetMinutes() % 60);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

int GetMinutes()

{

if (lstCustomer.SelectedIndex == -1)

return 0;

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

ID\_customer = Convert.ToInt32(lstCustomer.SelectedValue);

SqlCommand commandSelect = new SqlCommand("SELECT (count\_hour\*60+ count\_minutes) as minutes " +

"FROM tCustomer " +

" WHERE ID\_customer= @IDSS", connection);

commandSelect.Parameters.AddWithValue("@IDSS", Convert.ToInt32(ID\_customer));

int x = Convert.ToInt32(commandSelect.ExecuteScalar());

// MessageBox.Show(x.ToString());

return x;

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

return 0;

}

void UpdateHours(int x)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

ID\_customer = Convert.ToInt32(lstCustomer.SelectedValue);

SqlCommand commandUpdate = new SqlCommand("UPDATE tCustomer SET" +

" count\_hour=@count\_hour, count\_minutes=@count\_minutes " +

" WHERE ID\_customer= @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@count\_hour", x / 60 );

commandUpdate.Parameters.AddWithValue("@count\_minutes", x % 60);

commandUpdate.Parameters.AddWithValue("@IDSS", Convert.ToInt32(ID\_customer));

commandUpdate.ExecuteNonQuery();

// MessageBox.Show("Запись обновлена");

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

}

void SaveData()

{

int ID\_SS = 0;

try

{

connection.Close();

connection.Open();

SqlCommand commandInsert = new SqlCommand("INSERT INTO [tVisit] VALUES(" +

"@visitdate," +

"@timein," +

"@timeleft," +

"@ID\_customer," +

"@ID\_worker" +

") ; SELECT SCOPE\_IDENTITY()", connection);

commandInsert.Parameters.AddWithValue("@visitdate", dtpDate.Value.Date);

commandInsert.Parameters.AddWithValue("@timein", dtpTimeIn.Value.ToString("HH:mm"));

commandInsert.Parameters.AddWithValue("@timeleft", dtpTimeLeft.Value.ToString("HH:mm"));

ID\_customer = Convert.ToInt32(lstCustomer.SelectedValue);

commandInsert.Parameters.AddWithValue("@ID\_customer",ID\_customer);

if (cmbWorker.SelectedIndex == -1)

commandInsert.Parameters.AddWithValue("@ID\_worker", DBNull.Value);

else

commandInsert.Parameters.AddWithValue("@ID\_worker", cmbWorker.SelectedValue);

ID\_visit = Convert.ToInt32(commandInsert.ExecuteScalar());

MessageBox.Show("Запись добавлена");

add\_items = true;

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

} finally

{ connection.Close();

LoadDataFromTable();

} }

void UpdateData()

{

try

{

connection.Close();

connection.Open();

SqlCommand commandUpdate = new SqlCommand("UPDATE tVisit SET" +

" visitdate=@visitdate," +

"timein=@timein," +

"timeleft=@timeleft," +

"ID\_customer=@ID\_customer," +

"ID\_worker=@ID\_worker" +

" WHERE ID\_Visit= @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@visitdate", dtpDate.Value.Date);

commandUpdate.Parameters.AddWithValue("@timein", dtpTimeIn.Value.ToString("HH:mm"));

commandUpdate.Parameters.AddWithValue("@timeleft", dtpTimeLeft.Value.ToString("HH:mm"));

ID\_customer = Convert.ToInt32(lstCustomer.SelectedValue);

commandUpdate.Parameters.AddWithValue("@ID\_customer", ID\_customer);

if (cmbWorker.SelectedIndex ==-1)

commandUpdate.Parameters.AddWithValue("@ID\_worker", DBNull.Value);

else

commandUpdate.Parameters.AddWithValue("@ID\_worker", cmbWorker.SelectedValue);

commandUpdate.Parameters.AddWithValue("@IDSS", visit.ID\_visit);

commandUpdate.ExecuteNonQuery();

MessageBox.Show("Запись обновлена");

} catch (SqlException exception)

{ MessageBox.Show(exception.ToString());

} finally

{ connection.Close();

LoadDataFromTable();

} }

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if (bsBuilding.Count <= 0)

return;

if (GetMinutes() == 0)

{

MessageBox.Show("Купите абонемент", "Вход запрещен", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (dtpTimeLeft.Value.TimeOfDay < dtpTimeIn.Value.TimeOfDay)

{

MessageBox.Show("Время входа и выхода не корректны", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if ((tbCustomer.Text == ""))

{

MessageBox.Show("Ключевые поля пустые", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int h = GetMinutes();

int x = Convert.ToInt32(dtpTimeLeft.Value.Subtract(dtpTimeIn.Value).TotalMinutes);

if (add\_items == true)

{

int y = Convert.ToInt32(visit.timeleft.Subtract(visit.timein).TotalMinutes);

x = x - y;

if (x > h)

{

MessageBox.Show("Не достаточно времени", "Вход запрещен", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

UpdateData();

//MessageBox.Show(h.ToString() + " " + x.ToString()+" " + y.ToString());

UpdateHours(h - x);

}

else

{

if (x > h)

{

MessageBox.Show("Не достаточно времени", "Вход запрещен", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SaveData();

//MessageBox.Show(h.ToString() + " " + x.ToString());

UpdateHours(h - x);

}

tbTimeLeft.Text = String.Format("{0}ч. {1}мин.", GetMinutes() / 60, GetMinutes() % 60);

}

private void FlatForm\_Load(object sender, EventArgs e)

{

caninto = false;

LoadDataFromTable();

}

private void lstBuilding\_DoubleClick(object sender, EventArgs e)

{

if (lstCustomer.Items.Count <= 0)

return;

if (lstCustomer.SelectedIndex != -1)

{

tbCustomer.Text = lstCustomer.GetItemText(lstCustomer.SelectedItem);

tbTimeLeft.Text = String.Format("{0}ч. {1}мин.", GetMinutes() / 60, GetMinutes()% 60);

// MessageBox.Show(lstCustomer.SelectedValue.ToString());

} } }}

**Форма Клиенты**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

using System.Drawing.Imaging;

using System.IO;

namespace Kvartirs

{

public partial class CustomerForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

BindingSource bsPeople;

private DataTable dt;

SqlDataAdapter dataAdapter, dataAdapterPost;

bool add\_items;

public CustomerForm()

{

InitializeComponent();

LoadDataFromTable();

add\_items = false;

}

void LoadDataFromTable()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

ClearItems();

connection.Open();

SqlDataAdapter sda = new SqlDataAdapter("SELECT \* FROM tCustomer Order by sFIO, passport ", connection);

dt = new DataTable();

sda.Fill(dt);

bsPeople = new BindingSource();

//= dataTable;

bsPeople.DataSource = dt;

bindingNavigator1.BindingSource = bsPeople;

// Очистка

// textBox1

tbSurName.DataBindings.Clear();

tbSurName.DataBindings.Add(new Binding("Text", bsPeople, "sFIO"));

dtpBithday.DataBindings.Clear();

dtpBithday.DataBindings.Add(new Binding("Value", bsPeople, "Bithday"));

mtbPassport.DataBindings.Clear();

mtbPassport.DataBindings.Add(new Binding("Text", bsPeople, "Passport"));

tbPassportinfo.DataBindings.Clear();

tbPassportinfo.DataBindings.Add(new Binding("Text", bsPeople, "Passportinfo"));

tbAdress.DataBindings.Clear();

tbAdress.DataBindings.Add(new Binding("Text", bsPeople, "Address"));

tbPhone.DataBindings.Clear();

tbPhone.DataBindings.Add(new Binding("Text", bsPeople, "Phone"));

nudHours.DataBindings.Clear();

nudHours.DataBindings.Add(new Binding("Text", bsPeople, "count\_hour"));

nudMinutes.DataBindings.Clear();

nudMinutes.DataBindings.Add(new Binding("Text", bsPeople, "count\_minutes"));

dgvTypeTO.DataSource = bsPeople;

dgvTypeTO.Columns[0].Visible = false;

dgvTypeTO.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dgvTypeTO.Columns[1].HeaderText = "ФИО";

dgvTypeTO.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgvTypeTO.Columns[2].HeaderText = "Дата рождения";

dgvTypeTO.Columns[3].HeaderText = "Серия и номер паспорта";

dgvTypeTO.Columns[4].HeaderText = "Кем и когда выдан";

dgvTypeTO.Columns[5].HeaderText = "Адрес";

dgvTypeTO.Columns[6].HeaderText = "Телефон";

dgvTypeTO.Columns[7].HeaderText = "Фото";

((DataGridViewImageColumn)dgvTypeTO.Columns[7]).ImageLayout = DataGridViewImageCellLayout.Zoom;

dgvTypeTO.Columns[8].HeaderText = "остаток часов";

dgvTypeTO.Columns[9].HeaderText = "Остаток минут";

add\_items = false;

if (bsPeople.Count > 0) toolStripButton2.Enabled = true;

else toolStripButton2.Enabled = false;

} }

private void tbSurName\_KeyPress(object sender, KeyPressEventArgs e)

{

if (Char.IsLetter(e.KeyChar)) return;

if (Char.IsSeparator(e.KeyChar)) return;

if (Char.IsControl(e.KeyChar)) return;

e.Handled = true; }

private void ClearItems()

{ tbSurName.Text = "";

dtpBithday.Value = DateTime.Now;

mtbPassport.Text = "";

tbAdress.Text = "";

tbPhone.Text = "";

tbPassportinfo.Text = "";

nudHours.Value = 0;

nudMinutes.Value = 0;

pctPhoto.Image = pctPhoto.InitialImage;

}

private void AddStripButton\_Click(object sender, EventArgs e)

{

ClearItems();

add\_items = true;

}

private void toolStripButton2\_Click(object sender, EventArgs e)

{ if (add\_items)

{ add\_items = false;

saveToolStripButton.Enabled = true;

LoadDataFromTable();

} else if (bsPeople.Count > 0)

{ int i = bsPeople.Position;

string ID\_SS = ((DataRowView)this.bsPeople.Current).Row["ID\_customer"].ToString();

using (SqlConnection connection = new SqlConnection(connectionString))

{ connection.Open();

try

{ DialogResult result = MessageBox.Show("Вы действительно хотите удалить запись", "Внимание",

MessageBoxButtons.YesNo);

if (result == DialogResult.No)

{

LoadDataFromTable();

return; }

if (result == DialogResult.Yes)

{

SqlCommand commandDelete = new SqlCommand("Delete From tCustomer where ID\_customer = @ID\_customer", connection);

commandDelete.Parameters.AddWithValue("@ID\_customer", ID\_SS);

commandDelete.ExecuteNonQuery();

} }

catch (SqlException exception)

{ if ((uint)exception.ErrorCode == 0x80004005)

MessageBox.Show("Удаление прервано, есть связанные записи", "Error", MessageBoxButtons.OK,

MessageBoxIcon.Error);

else

MessageBox.Show(exception.ToString());

}

finally

{

LoadDataFromTable();

} } } }

void InsertData()

{

string s = "";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

SqlCommand commandInsert = new SqlCommand("INSERT INTO [tCustomer] VALUES(" +

"@sFIO," +

"@bithday," +

"@passport," +

"@passportinfo," +

"@address," +

"@phone," +

"@photo," +

"@count\_hour," +

"@count\_minutes" +

") ", connection);

commandInsert.Parameters.AddWithValue("@sFIO", tbSurName.Text);

commandInsert.Parameters.AddWithValue("@bithday", dtpBithday.Value.Date);

commandInsert.Parameters.AddWithValue("@passport", mtbPassport.Text);

commandInsert.Parameters.AddWithValue("@passportinfo", tbPassportinfo.Text);

commandInsert.Parameters.AddWithValue("@address", tbAdress.Text);

commandInsert.Parameters.AddWithValue("@phone", tbPhone.Text);

commandInsert.Parameters.AddWithValue("@photo", ConvertInBytes(pctPhoto.Image));

commandInsert.Parameters.AddWithValue("@count\_hour", Convert.ToInt32(nudHours.Text));

commandInsert.Parameters.AddWithValue("@count\_minutes", Convert.ToInt32(nudMinutes.Text));

s = mtbPassport.Text;

commandInsert.ExecuteNonQuery();

MessageBox.Show("Запись добавлена");

add\_items = false;

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

connection.Close();

LoadDataFromTable();

bsPeople.Position = bsPeople.Find("passport", s);

} } }

void UpdateData()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string x = "";

connection.Open();

try

{ string ID\_SS = ((DataRowView)this.bsPeople.Current).Row["ID\_customer"].ToString();

x = ID\_SS;

SqlCommand commandUpdate = new SqlCommand("UPDATE tCustomer SET" +

" sFIO=@sFIO," +

"bithday=@bithday," +

"passport=@passport," +

"passportinfo=@passportinfo," +

"address=@address," +

"phone=@phone," +

"photo=@photo," +

"count\_hour=@count\_hour," +

"count\_minutes=@count\_minutes" +

" WHERE ID\_customer= @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@sFIO", tbSurName.Text);

commandUpdate.Parameters.AddWithValue("@bithday", dtpBithday.Value.Date);

commandUpdate.Parameters.AddWithValue("@passport", mtbPassport.Text);

commandUpdate.Parameters.AddWithValue("@passportinfo", tbPassportinfo.Text);

commandUpdate.Parameters.AddWithValue("@address", tbAdress.Text);

commandUpdate.Parameters.AddWithValue("@phone", tbPhone.Text);

commandUpdate.Parameters.AddWithValue("@photo", ConvertInBytes(pctPhoto.Image));

commandUpdate.Parameters.AddWithValue("@count\_hour", Convert.ToInt32(nudHours.Text));

commandUpdate.Parameters.AddWithValue("@count\_minutes", Convert.ToInt32(nudMinutes.Text));

commandUpdate.Parameters.AddWithValue("@IDSS", ID\_SS);

commandUpdate.ExecuteNonQuery();

MessageBox.Show("Запись обновлена");

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

connection.Close();

LoadDataFromTable();

bsPeople.Position = bsPeople.Find("ID\_customer", x);

} } }

private void saveToolStripButton\_Click(object sender, EventArgs e)

{

if ((mtbPassport.Text == "") || (tbSurName.Text == ""))

{

MessageBox.Show("Ключевые поля пустые", "error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (add\_items)

{

InsertData();

}

else

{

UpdateData(); } }

private void tbPassport\_TextChanged(object sender, EventArgs e)

{ /if (mtbPassport.Text == "" || tbSurName.Text == "") saveToolStripButton.Enabled = false; //else saveToolStripButton.Enabled = true;

}

private void dgvTypeTO\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{ var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{ // right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat); }

private void dgvTypeTO\_SelectionChanged(object sender, EventArgs e)

{ pctPhoto.Image = LoadImage();

} byte[] ConvertInBytes(Image img)

{ byte[] bytes;

using (MemoryStream ms = new MemoryStream())

{ img.Save(ms, ImageFormat.Png);

bytes = ms.ToArray();

} return bytes;

} Image LoadImage()

{ if (bsPeople.Count > 0)

{

int i = bsPeople.Position;

if ((((DataRowView)this.bsPeople.Current).Row["photo"] != System.DBNull.Value))

{

byte[] byteArrayIn = (byte[])(((DataRowView)this.bsPeople.Current).Row["photo"]);

using (var ms = new MemoryStream(byteArrayIn))

{

return Image.FromStream(ms);

}

}

return null;

}

else return null;

}

private void pctPhoto\_DoubleClick(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

// получаем выбранный файл

try

{

string filename = openFileDialog1.FileName;

pctPhoto.Image = Image.FromFile(filename);// читаем файл в строку

}

catch

{

MessageBox.Show("Ошибка загрузки файла", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgGrid\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{

// right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat); } }}

**Приложение Б**

Автоматизированная информационная система «Фитнес-клуб S-Fitnes»

Спецификация

460.ЗМК.00004-01 01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Обозначение** | | |  | | --- | | **Наименование** | | |  | | --- | | **Примечание** | |
|  | |  | | --- | | Документация | |  |
| 460.ЗМК.00027-01 12 01 | Автоматизированная информационная система |  |
|  | Фитнес-клуб S-Fitnes |  |
|  | Текст программы |  |
| 460.ЗМК.00027-01 13 01 | Автоматизированная информационная система |  |
|  | Фитнес-клуб S-Fitnes |  |
|  | Описание программы |  |
| 460.ЗМК.00027-01 34 01 | Автоматизированная информационная система |  |
|  | Фитнес-клуб S-Fitnes |  |
|  | Руководство пользователя |  |
| 460.ЗМК.00027-01 81 01 | Автоматизированная информационная система |  |
|  | Фитнес-клуб S-Fitnes |  |
|  | Пояснительная записка |  |

**Приложение B**

**Диск с программным средством**