РЕФЕРАТ

Пояснительная записка: 75 с., 55 рис., 17 табл., 12 источников, 2 прил.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ ФИТНЕС-КЛУБОМ "ANTARES X-FIT PREMIUM", УЧЕТ АБОНЕМЕНТОВ, ФИЗИЧЕСКАЯ МОДЕЛЬ БД, ЛОГИЧЕСКАЯ МОДЕЛЬ БД, ЦЕЛОСТНОСТЬ ДАННЫХ, IDEF0, КОНТЕКСТНАЯ ДИАГРАММА, ИНТЕРФЕЙС, WINDOWS 7, MS ACCESS, ИНФОРМАЦИОННАЯ МОДЕЛЬ, КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ, ER-МОДЕЛЬ, ИНТЕРФЕЙС, BPWIN, ERWIN, VISUAL STUDIO.

Объектом исследования является автоматизированная система управления фитнес-клубом «ANTARES X-FIT PREMIUM». Цель выпускной квалификационной работы разработать приложение, позволяющее автоматизировать учет абонементов, занятий, клиентов, сотрудников фитнес-клуба.

В процессе работы выполнено: изучены способы получения и хранения информации в базах ms access и организация пользовательского интерфейса для работы с базой данных в среде разработки Visual Studio 2010.

Областью возможного практического применения является фитнес-клуб.

Результатом внедрения приложения является своевременное отображение информации о расписании, абонементах фитнес-клуба «ANTARES X-FIT PREMIUM».

Приведенный в выпускной квалификационной работе расчетно-аналитический материал объективно отражает состояние разработок автоматизированных систем управления фитнес-клубом на современном этапе. Все заимствованные из литературных и других источников теоретические и методологические положения и концепции сопровождаются ссылками на их авторов.

Содержание

[ВВЕДЕНИЕ 6](#_Toc71408818)

[1 Анализ и обоснование актуальности тематики 8](#_Toc71408819)

[1.1 Описание предметной области 8](#_Toc71408820)

[1.2 Обзор программ-аналогов 11](#_Toc71408821)

[1.2.1 Fitness Pro - программа для фитнес клубов и тренажерных залов 11](#_Toc71408822)

[1.2.2 Программа «Учет клиентов». Конфигурация «Фитнес-клуб» 13](#_Toc71408823)

[2 Проектирование структуры и компонентов программного продукта 18](#_Toc71408824)

[2.1 Анализ штатной структуры организации 18](#_Toc71408825)

[2.2 Задачи, подлежащие автоматизации 19](#_Toc71408826)

[2.3 Разработка программной реализации моделей компонентов 21](#_Toc71408827)

[2.3.1 Выбор СУБД 21](#_Toc71408828)

[2.3.2 Выбор и обоснование языка и среды программирования 23](#_Toc71408829)

[2.4 Проектирование и Разработка базы данных 27](#_Toc71408830)

[2.5 Построение функциональной модели предметной области в программной среде BPWin 27](#_Toc71408831)

[2.6 Проектирование логической и физической модели данных 28](#_Toc71408832)

[2.6.1 Разработка инфологической модели предметной области 28](#_Toc71408833)

[2.6.2 Разработка даталогической модели БД 31](#_Toc71408834)

[2.6.3 Построение логической модели базы данных в программе Erwin 36](#_Toc71408835)

[2.6.4 Построение физической модели базы данных в программе Erwin 37](#_Toc71408836)

[2.7 Функциональная структура системы 39](#_Toc71408837)

[3 Разработка технической документации для сопровождения программного приложения 41](#_Toc71408838)

[3.1 Постановка информационной задачи 41](#_Toc71408839)

[3.2 Организация входных и выходных данных 41](#_Toc71408840)

[3.3 Требования к функциям (задачам), выполняемым системой 42](#_Toc71408841)

[3.4 Формализованное описание процесса оформления абонемента 43](#_Toc71408842)

[3.5 Требования к надежности. Условия эксплуатации и требования к составу и параметрам технических средств 44](#_Toc71408843)

[3.6 Требования к интерфейсу. Функциональная структура информационной системы 45](#_Toc71408844)

[3.7 Требования к маркировке и упаковке 45](#_Toc71408845)

[4 Тестирование программного продукта 46](#_Toc71408846)

[4.1 Авторизация 46](#_Toc71408847)

[4.2 Главное окно. Вкладка Заставка 47](#_Toc71408848)

[4.3 Главное окно. Вкладка Расписание 49](#_Toc71408849)

[4.4 Главное окно. Вкладка Абонементы 52](#_Toc71408850)

[4.5 Главное окно. Вкладка Журнал занятий 56](#_Toc71408851)

[4.6 Главное окно. Вкладка Сотрудники 58](#_Toc71408852)

[4.7 Главное окно. Вкладка Клиенты 60](#_Toc71408853)

[4.8 Главное окно. Вкладка Справочники 61](#_Toc71408854)

[4.9 Вспомогательные формы. Форма Клиент 63](#_Toc71408855)

[4.10 Вспомогательные формы. Форма Группа 64](#_Toc71408856)

[4.11 Вспомогательные формы. Форма Зал 65](#_Toc71408857)

[4.12 Вспомогательные формы. Форма Вид абонемента 66](#_Toc71408858)

[4.13 Вспомогательные формы. Период действия расписания 66](#_Toc71408859)

[4.14 Вспомогательные формы. Формы Сотрудник, форма смены пароля сотрудника 67](#_Toc71408860)

[4.15 Вспомогательные формы. Формы смены пароля 68](#_Toc71408861)

[4.16 Вспомогательные формы. Формы редактирования атрибутов в справочниках 69](#_Toc71408862)

[4.17 Экспорт данных из таблиц 70](#_Toc71408863)

[4.18 Справочная система 71](#_Toc71408864)

[Заключение 73](#_Toc71408865)

[Литература 76](#_Toc71408866)

[Приложение А. Листинг модулей 77](#_Toc71408867)

[Приложение Б. Руководство пользователя 111](#_Toc71408868)

# ВВЕДЕНИЕ

Целью данной выпускной квалификационной работы является проектирование и разработка автоматизированной системы управления фитнес-клубом. Разработку данной системы проведем на примере фитнес-клуба «ANTARES X-FIT PREMIUM». Фитнес-клубу «ANTARES X-FIT PREMIUM» необходимо вести учет выданных абонементов, зарегистрированных клиентов, проведенных занятий, а так же составлять расписание занятий. Для достижения целей, которые стоят перед фитнес-клубом «ANTARES X-FIT PREMIUM», необходимо современное программное обеспечение. Внедрение новых программ позволяет облегчить все этапы составления расписания, учет абонементов, что повлечет за собой экономию времени и средств[1]. Автоматизированная система управления фитнес-клубом "ANTARES X-FIT PREMIUM" позволит облегчить работу всех сотрудников данной организации.

Актуальностью данной программы является помощь менеджеру отдела продаж, который занимается оформлением абонементов, фитнес-менеджеру, составляющему расписание. Благодаря данной программе трудоёмкость процесса учёта абонементов, занятий снижается в разы. Раньше менеджеру отдела продаж, отвечающему за учёт абонементов и клиентов необходимо было перебирать и заполнять кучу бумаг, которые занимали много места и часто терялись. Позже данная задача решалась с помощью программы Microsoft Excel. Файл MS Excel, конечно, упрощал работу, но содержал очень много листов, которые были связаны между собой. Настройки взаимосвязи листов данной книги требовало высокой концентрации от программиста, и появление ошибок от неправильного ввода формулы тяжело было отследить. С внедрением автоматизированной системы управления фитнес-клубом работа специалистов значительно упростится. Любая операция занимает не более пяти минут, а информация хранится в электронном виде, что делает её хранение компактным и надёжным. Получение данных, отчетов становится практически мгновенным делом.

Программу предполагается использовать на рабочем месте менеджера отдела продаж. Автоматизированная система управления фитнес-клубом "ANTARES X-FIT PREMIUM" позволит облегчить процесс поиска, добавления, удаления информации, выдачи абонементов, а также мониторинг проведения занятий.

Так как система позволит увеличить скорость обработки информации, то уменьшится трудоёмкость процесса и увеличится количество работы выполненной работником.

На сегодняшний день мало программ, которые были бы довольно просты в использовании. Данная программа проста в обращении и не требует углубленных знаний компьютера, пользователю достаточно уметь открывать файл.

Этому вопросу и посвящена данная работа.

Цель исследования: проектирование и разработка автоматизированной системы управления фитнес-клубом "ANTARES X-FIT PREMIUM".

# 1 Анализ и обоснование актуальности тематики

## 1.1 Описание предметной области

В данной работе рассматривается предметная область – фитнес-клуб «ANTARES X-FIT PREMIUM». Данный фитнес-клуб имеет два собственных спортивных комплекса по адресу г. Екатеринбург ул. Шейкмана, 121 и ул. Шейкмана, 124. В данном комплексе занятия проводятся занятия в 6 спортивных залах [1].

Таблица – Информация о залах фитнес-клуба «ANTARES X-FIT PREMIUM»

|  |  |  |  |
| --- | --- | --- | --- |
| **Зал** | **Адрес** | **Площадь** | **Вместимость** |
| Большой зал групповых программ | ул. Шейкмана, 121 | 300 | 50 |
| Зал фитнеса, танцев | ул. Шейкмана, 121 | 120 | 30 |
| Зал для силовых тренировок | ул. Шейкмана, 124 | 153 | 50 |
| Малый зал групповых программ | ул. Шейкмана, 124 | 60 | 15 |
| Зал новые направления фитнеса. Силовые тренировки | ул. Шейкмана, 124 | 115 | 30 |
| Мини-бассейн | ул. Шейкмана, 121 | 120 | 20 |

Залы различное назначение и возможности по количеству одновременно занимающихся клиентов.

В фитнес-клубе «ANTARES X-FIT PREMIUM» проводятся занятия по следующим направлениям

1. Направление;
2. Силовые направления;
3. Пилатес;
4. Аэробика;
5. TRX;
6. Йога;
7. Танцевальные направления;
8. Детский фитнес;
9. Дыхательные направления;
10. Фитнес диетолог;
11. Женский бокс
12. Мужской бокс

Расписание занятий составляется для каждого отдельного зала, на каждый день недели. Расписание действует в заданный временной период, который имеет дату начала периода и дату конца периода. В фитнес-клубе «ANTARES X-FIT PREMIUM» принято фиксированное время начала занятий: 8:00, 9:00, 10:00, 11:00, 12:00, 13:00, 14:00, 15:00, 16:00, 17:00, 18:00, 19:00, 20:00, 21:00.

Длительность занятия: 60 минут.

Организация посещения занятий, продажи абонементов и проведения занятий

1. Посещение занятий осуществляется по абонементам:
   1. Дневная – 1 занятие
   2. Без заморозки – 8 занятий в месяц
   3. Индивидуальная на полгода – 12 занятий в месяц
   4. Детский AQUA – 1 занятие в неделю
   5. VIP-абонемент (60 занятий)
   6. Семейная – 6 занятий в месяц
2. Абонемент является универсальным (клиент имеет право посещать любые форматы уроков по сетке расписания, привязка абонемента осуществляется к спортивному залу)
3. Начало занятий может осуществляться с любого числа месяца. Срок действия абонемента со дня первой тренировки до такой же даты следующего месяца (включительно). Пропущенные занятия сохраняются при предоставлении медицинской справки.
4. Первое занятие – бесплатно (Одно, в любом из филиалов)
5. Посещение занятий осуществляется без предварительной записи (кроме занятий TRX и групп в стадии формирования)
6. Скидки пенсионерам, подросткам, многодетным мамам (от 3 детей), и нескольким членам семьи (родители плюс дети до 18 лет)

Правила оплаты Абонемента [2].

1. Оплата услуг производится единовременно путем 100% предварительной оплаты НЕ ПОЗДНЕЕ ДНЯ ОПЛАЧИВАЕМОГО ПЕРИОДА. Оплата осуществляется на рецепции клуба в наличной форме или по пластиковой карте исключительно в валюте РФ – российский рубль.
2. После оплаты услуг выдается АБОНЕМЕНТ, который подтверждает право пользования данного Клиента фитнес-услугами в соответствии с видом приобретаемого абонемента и одновременно является пропуском в выбранный зал. Клиент не имеет право передавать свой абонемент другому лицу.
3. В случае досрочного прекращения посещения фитнес-клуба «ANTARES X-FIT PREMIUM» стоимость абонемента не возвращается
4. Оплата разового занятия или приобретение абонемента подтверждает согласие клиента со всеми перечисленными правилами фитнес-клуба «ANTARES X-FIT PREMIUM» и обязывает клиента к выполнению таковых. При нарушении правил, администрация оставляет за собой право рассмотреть вопрос о дальнейшем посещении клиентом занятий фитнес-клуба «ANTARES X-FIT PREMIUM», в случае чего стоимость абонемента и разового занятия не возвращается.
5. Администрация оставляет за собой право вносить изменения в действующее расписание занятий. Занятия могут переноситься в связи с выходными и праздничными днями, совпадающими с государственными.
6. Указанные в расписании занятия не предполагают привязки к конкретному преподавателю. Это означает, что в случае замены действие абонемента не продлевается, и деньги за абонемент не возвращаются.
7. Посещение фитнес-клуб «ANTARES X-FIT PREMIUM» осуществляется по системе групповых занятий, в определенном зале, с определенными днями и временем посещения.
8. Инструктор имеет право не проводить занятие, если на групповую тренировку пришли только два человека.

## 1.2 Обзор программ-аналогов

1.2.1 Fitness Pro - программа для фитнес клубов и тренажерных залов

Функционал программы Fitness Pro [3]:

1. Работа с клиентами
   1. мгновенный поиск информации
   2. полная история визитов и покупок
   3. карты и абонементы
   4. программы лояльности и акции
   5. СМС-уведомления о визитах
2. Расписание клуба
   1. интуитивно-понятный интерфейс
   2. быстрая запись и расчет
   3. контроль загруженности клуба
   4. исключение ошибок
   5. планирование повторных визитов
3. Онлайн запись
   1. возможность записи 24/7
   2. запись из Facebook и Instagram
   3. запись с сайта клуба
   4. личный кабинет клиента
   5. генерирует дополнительные продажи
4. Работа с товарами
   1. полный контроль товарооборота
   2. учет расходных материалов
   3. взаиморасчеты с поставщиками
   4. планирование закупок
   5. простой процесс инвентаризации
5. Управление сотрудниками
   1. управление графиками работы
   2. автоматический расчет зарплат
   3. аналитика эффективности сотрудников
   4. система напоминаний и обмен сообщениями
   5. планирование показателей по сотрудникам
6. Отчёты и статистика
   1. возможность контроля бизнеса 24/7
   2. более 150 отчетов
   3. возможность получения отчетов по email
   4. финансовый анализ и планирование
   5. мобильное приложение для сотрудников

Таблица – Стоимость пользования

|  |  |  |  |
| --- | --- | --- | --- |
| Цена | Версия | | |
| Light | Standard | Ultimate |
| Цена за месяц | 969 | 1938 | 4522 |
| Цена за год | 9690 | 19381 | 45222 |

Интерфейс Приложения приведен на рисунках 1-2.

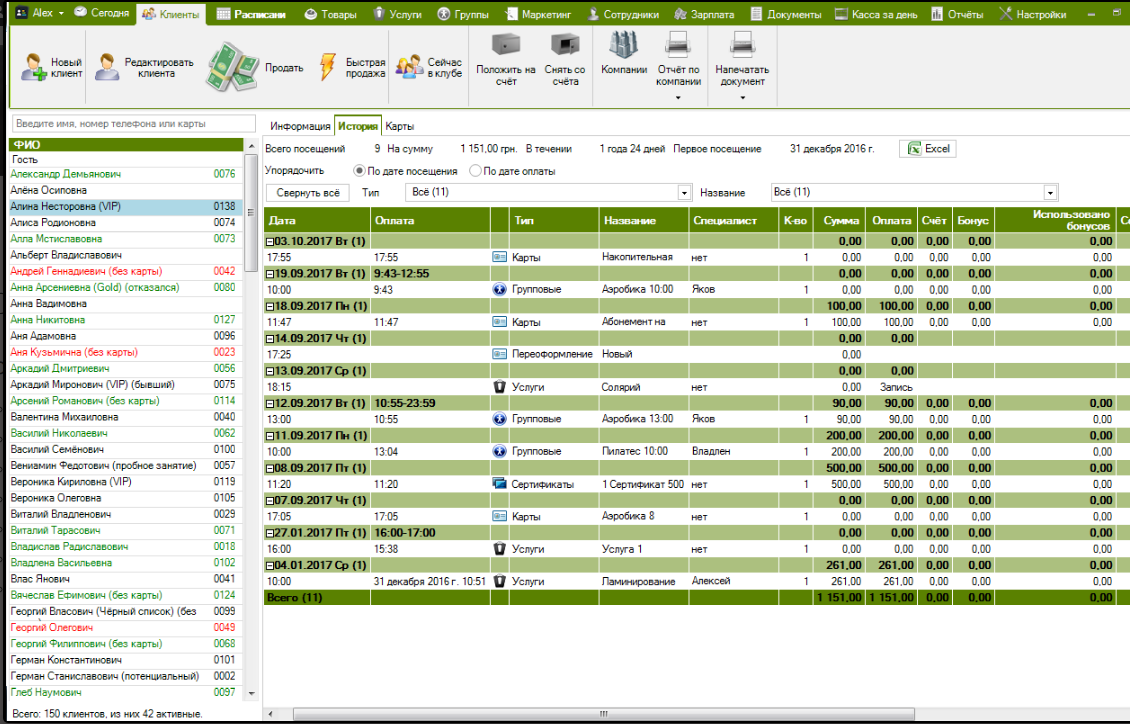


Рисунок – Автоматизация работы с клиентами

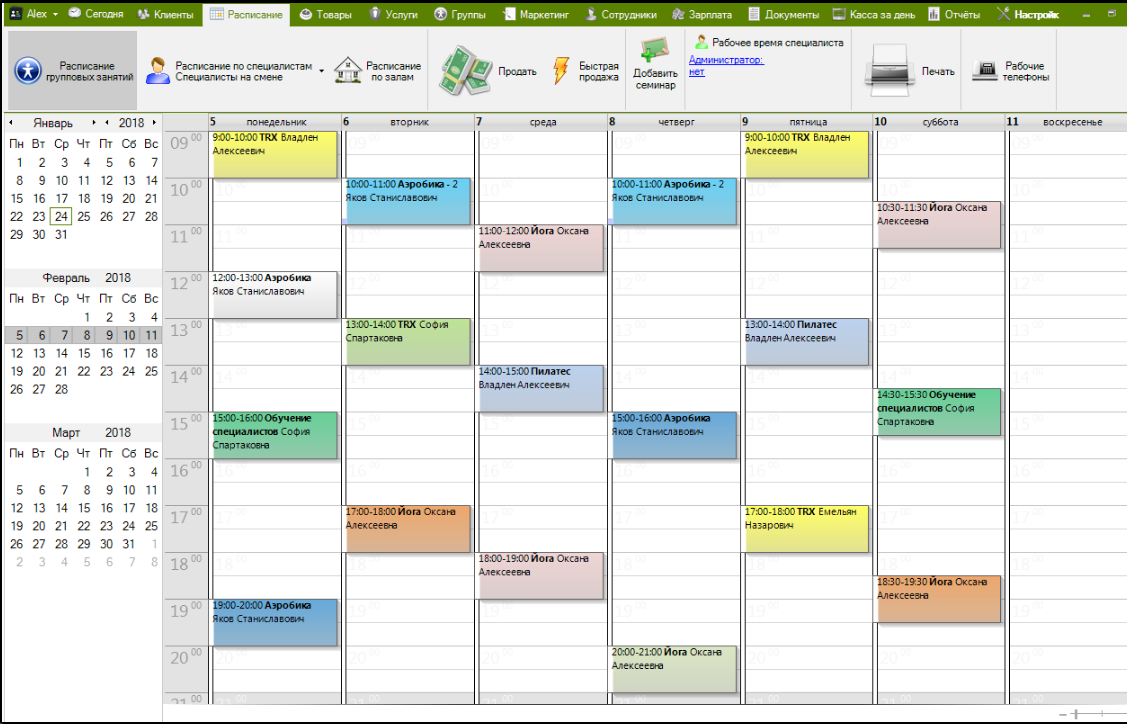


Рисунок – Расписание

Средства программной реализации приложения - windows–приложение. База данных – Microsoft Access (\*.mdb)

Требования к оборудованию

1. ОС MAC - требуется предварительно установленный Virtual Box с ОС Windows;
2. наличие интернет соединения со скоростью не менее 10 Мб/с;
3. USB разъёмы в количестве, соответствующем оборудованию, которое Вы хотите подключить;
4. рекомендуемая для комфортной работы диагональ дисплея не менее 14 дюймов.

1.2.2 Программа «Учет клиентов». Конфигурация «Фитнес-клуб»

Конфигурация «Фитнес-клуб» создана для комплексной автоматизации работы фитнес-клубов, спортивных центров и бассейнов. Главная ее задача – создать полноценный учет посещаемости клуба, сформировав при этом базу данных клиентов [4].

В системе содержаться не только личные данные посетителя, но и тип его клубной карты, дата заключения и окончания контракта, а также другая важная для администратора и руководителя информация.

Основные функции:

1. Учет клиентов и сбор их контактной информации
2. Учет работы всех филиалов сети
3. Учет частоты посещений
4. Учет числа абонементов, клубных карт и разовых посещений
5. Идентификация посетителя по клубной карте
6. Учет оказанных услуг
7. Создание расписаний групповых занятий в разных залах
8. Контроль оплаты
9. Создание отчетов

Интерфейс программы приведен на рис. 3-6.

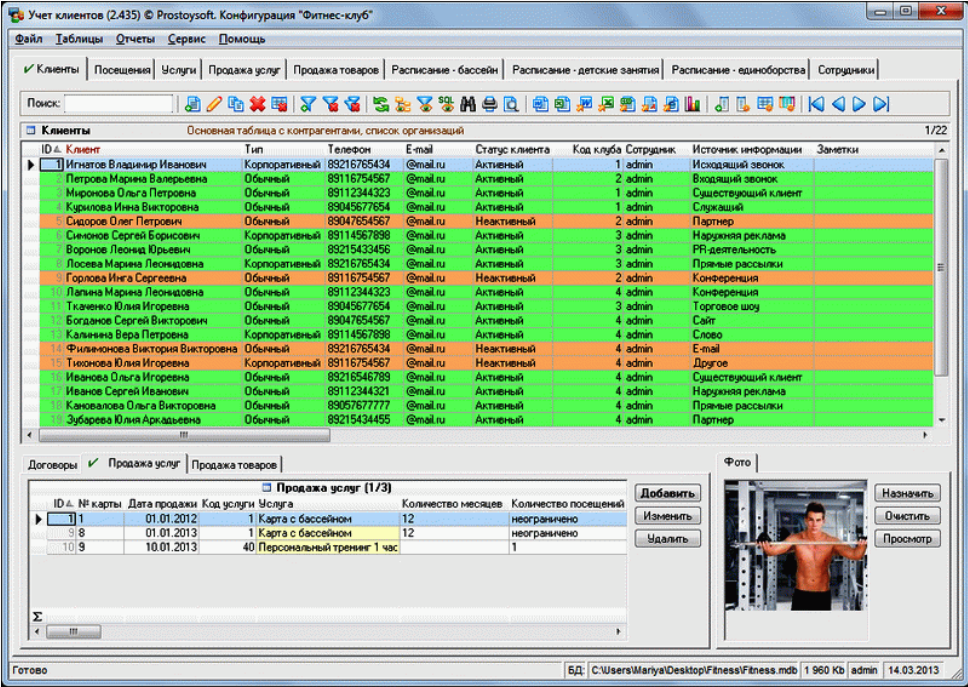


Рисунок – Ведение списка клиентов фитнес-клуба и их контактных данных

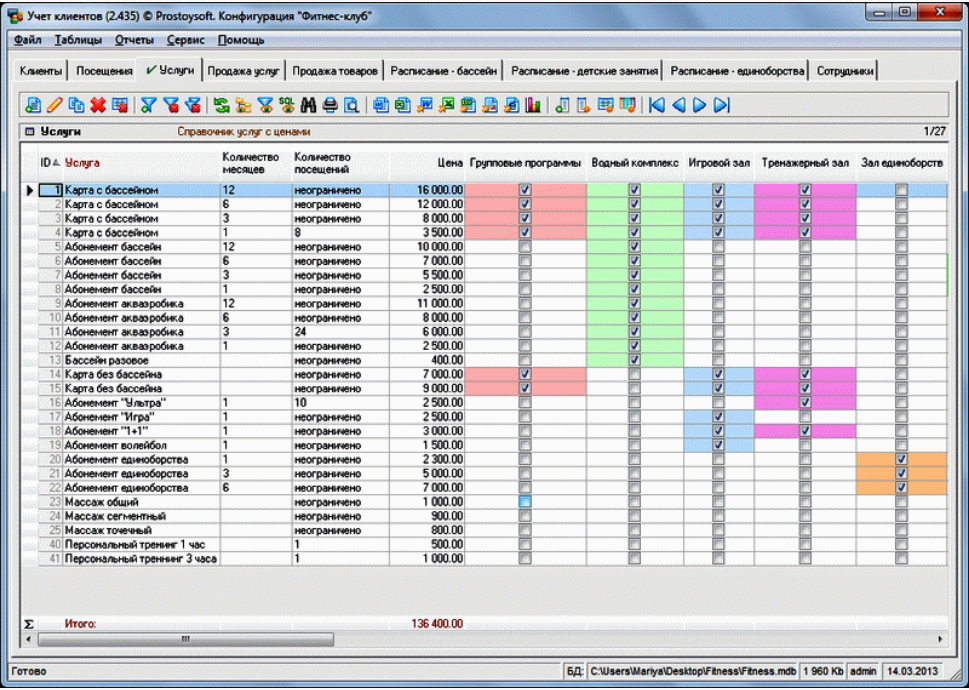


Рисунок – Справочники услуг и товаров клуба

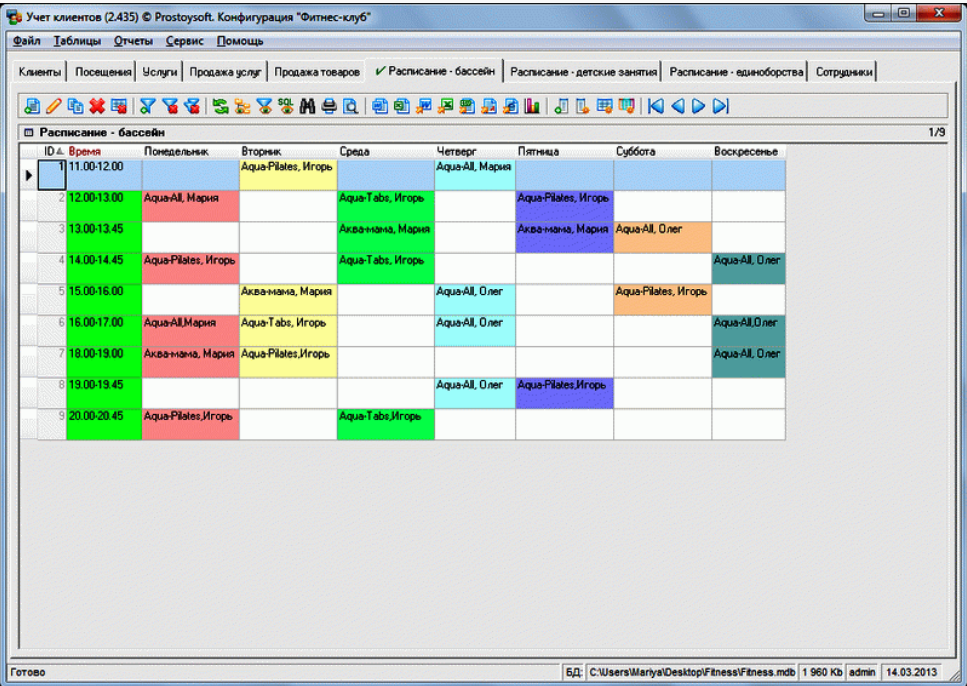


Рисунок – Ведение расписания занятий по отдельным залам и видам спорта

Средства программной реализации модели – windows-приложение. База данных – Microsoft Access (\*.mdb)

Таблица – Стоимость пользования

|  |  |  |  |
| --- | --- | --- | --- |
| Легкий | Базовый | Стандартный | Супер |
| 450 руб. | 890 руб. | 1890 руб. | 3900 руб. |
| 1 Сотрудник | До 6 сотрудников | До 15 сотрудников | Неограниченно сотрудников |
| 1 Офис | До 3 офисов | Неограниченно офисов | Неограниченно офисов |
| Неограниченно событий | Неограниченно событий | Неограниченно событий | Неограниченно событий |
| Неограниченно сервисов | Неограниченно сервисов | Неограниченно сервисов | Неограниченно сервисов |
| Неограниченно клиентов | Неограниченно клиентов | Неограниченно клиентов | Неограниченно клиентов |
| Организация классов/групп | Организация классов/групп | Организация классов/групп | Организация классов/групп |
| Доступны все возможности | Доступны все возможности | Доступны все возможности | Доступны все возможности |

Приведенные программы обладают излишним функционалом и дороги в эксплуатации. Большинство функций реализованы в 1С: бухгалтерия, используемой в фитнес-клубе "ANTARES X-FIT PREMIUM". А для нескольких рабочих мест: фитнес-менеджер, финансовый администратор, инструктор для учета выдачи абонементов достаточно более простой программы, которая бы учитывала все тонкости систему управления фитнес-клуба "ANTARES X-FIT PREMIUM" именно в этом сегменте информационной задачи.

# 2 Проектирование структуры и компонентов программного продукта

## 2.1 Анализ штатной структуры организации

Далее необходимо рассмотреть штатную структуру фитнес-клуба «ANTARES X-FIT PREMIUM» - рисунок 6.



Рисунок – Штатная структура фитнес-клуба «ANTARES X-FIT PREMIUM»

Анализируя организационную структуру фитнес-клуба «ANTARES X-FIT PREMIUM», в части касающейся составления расписания и учета абонементов, можно сделать следующие выводы:

1. Фитнес-клуб «ANTARES X-FIT PREMIUM» имеет всего двух сотрудников бухгалтерской службы. Это связано с малым количество сотрудников и большой степени автоматизации бухгалтерского учета. В фитнес-клубе «ANTARES X-FIT PREMIUM» для автоматизации бухгалтерского учета используется программа 1С: Предприятие.
2. Задачей Фитнес-менеджера является управление инструкторами, составление расписания занятий.
3. Тренерский состав включает в себя инструкторов (11 человек). Весь тренерский состав окончил специальные курсы по оказанию первой медицинской помощи и ежегодно проходит переподготовку, это является обязательным условием работы в тренажерном зале.
4. Администратор на рецепции регистрирует посещение клиентов
5. Заместители директора осуществляют руководство осуществлением деятельности по хозяйственной части и организацией занятий.
6. Финансовый администратор регистрирует клиентов, осуществляет оформление и выдачу абонементов, регистрирует оплату по абонементам.

## 2.2 Задачи, подлежащие автоматизации

При анализе задач, подлежащих автоматизации следует, учитывать факт, того, что бухгалтерское подразделение фитнес-клуба «ANTARES X-FIT PREMIUM» уже автоматизировано. Оценка повторной автоматизации является сложным комплексом мер, не гарантирующих повышение эффективности работы тренажерного зала в целом, в тоже время, требующее расходов как минимум на исследование, анализ и внедрение.

Центральными бизнес-процессами, увеличивающими фитнес-клуба зала, являются:

1. добавление нового клиента в систему;
2. продажа абонемента клиента;
3. проведение занятия в том или ином зале, тем или иным инструктором;
4. Добавление расписания.

Можно выделить следующие информационные объекты, связанные с указанными бизнес-процессами:

1. Зал. Информация о зале должна содержать Название, Адрес, Описание, Площадь;
2. Клиент. Информация о клиенте минимум должна содержать – фамилию, имя, отчество и контактные данные: телефон и, при наличии, адрес электронной почты.
3. Сотрудник. Информация о сотруднике должна в себя включать должность (инструктор, фитнес-менеджер, администратор и т.д.), мобильный телефон для оперативной связи и другую информацию.
4. Группы занятий. Необходимы: наименование группы, направление.
5. Направление занятий. Необходимы: название направления, описание.
6. Должности сотрудников. Необходимо название должности.
7. Расписание занятий. Должно содержать: период действия расписания, день недели, время начала занятий, длительность занятия, информацию о группе, информацию об инструкторе.
8. Журнал проведения занятий. Должен отражать проведение текущих занятий и содержать данные о дате проведения занятия, отметку о выполнении, количество человек на занятии. Учитывать места проведения занятий в зале, инструктора, осуществляющего проведение занятия.

Учитывая выделенные информационные объекты, пользователями данной автоматизированной информационной системы будут являться:

1. Фитнес-менеджер. В его задачи входит контроль над деятельностью инструкторов, ведение журнала занятий.
2. Администратор фитнес-клуба. Он должен заниматься регистрацией новых клиентов, оформлять абонементы, регистрировать оплату.
3. Директор тренажерного зала. Выполняет общий контроль и получает аналитику по работе тренажерного зала.
4. Системный администратор. Настройка приложения, редактирование справочников базы данных по указаниям директора и заявкам фитнес-менеджера, администратора фитнес-клуба.
5. Начальник подчиненного фитнес клуба. Просмотр расписания занятий в подведомственном ему помещении.
6. Заместитель директора: просмотр расписания занятий. Получение отчетов.
7. Бухгалтер: просмотр расписания занятий. Получение отчетов.
8. Инструктор: просмотр расписания занятий. Занесение данных о проведенных занятиях в журнал занятий.

## 2.3 Разработка программной реализации моделей компонентов

2.3.1 Выбор СУБД

Выбор СУБД, как средства для создания функционирования базы данных автоматизированной системы, осуществляется на основе анализа ряда характеристик современных СУБД, представленных в таблице 4.

Таблица 4 – Сравнительная характеристика СУБД

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Функция | SQL Server 2008 | Oracle | MySQL | Access | Visual FoxPro |
| Операционная система | Windows 2000 или NT и выше | OS/2, UNIX, Windows, MS-DOS | Windows NT и выше | Windows 95 и выше | Windows 95 или NT |
| Поддержка SQL | **+** | **+** | **+** | **+** | **+** |
| Поддержка функций ODBC | **+** | **+** | **+** | **+** | **-** |
| Поддержка Visual Basic | **-** | **-** | **-** | **+** |  |
| Поддержка СОМ | **+** | **+** |  | **+** | **+** |
| Представления (Views) | **+** | **-** | **+** | **-** | **-** |
| Индексированные представления | **+** | **-** | **-** | **-** | **-** |
| Использование OLE DB | **+** | **+** | **-** | **+** | **+** |
| Интеграция с пакетом Office 2000 | **+** | **-** | **-** | **+** | **-** |
| Средства создания форм и отчетов | **-** | **-** | **-** | **+** | **+** |
| Средства анализа данных OLAP | **+** | **+** | **-** | **-** | **-** |
| Копирование БД | **+** | **+** | **-** | **+** | **+** |
| Развитая поддержка XML | **+** | **+** | **-** | **-** | **-** |
| Безопасность | **+** | **+** | **-** | **-** | **-** |
| Администрирование базы данных | **+** | **+** | **-** | **+** | **-** |
| Доступ к данным через интернет | **+** | **+** | **+** | **+** | **+** |
| Импорт и экспорт | **+** | **+** | **-** | **+** | **+** |
| Поддержка JDBC - драйвера | **+** | **+** | **+** | **-** | **-** |
| Хранимые процедуры | **+** | **+** | **-** | **-** | **-** |
| Триггеры | **+** | **+** | **-** | **-** | **-** |

В качестве СУБД для хранения данных выбирается Microsoft Office Access. Это обуславливается прежде всего тем, что на всех компьютерах фитнес-клуба размещены лицензионные пакеты Microsoft Office, включающие в том числе MS Access. Для организации нет смысла покупать какие-то новые СУБД.

Microsoft Office Access или просто Microsoft Access — реляционная СУБД корпорации Microsoft. Имеет широкий спектр функций, включая связанные запросы, связь с внешними таблицами и базами данных. Благодаря встроенному языку VBA, в самом Access можно писать приложения, работающие с базами данных.

Access можно использовать для хранения больших объемов данных различных типов. На более высоком уровне с помощью Access можно создавать небольшие программы, позволяющие решать определенные задачи. Access не только позволит сохранить информацию, но также позволит сортировать ее, обрабатывать и анализировать в соответствии с очередными тенденциями или по ключевым данным записям.

1. Достоинства Access:
2. Access хранит все данные в одном файле, хотя и распределяет их по разным таблицам, как и положено реляционной СУБД. К этим данным относится не только информация в таблицах, но и другие объекты базы данных.
3. Распространенность, которая обусловлена тем, что Access является продуктом компании Microsoft.
4. Постоянно обновляется производителем, поддерживает множество языков.
5. Полностью совместим с операционной системой Windows.
6. Ориентированность на пользователя с разной профессиональной подготовкой, что выражается в наличии большого количества Мастеров, развитую систему справки и понятный интерфейс.

Вывод: итог выбора СУБД: Microsoft Office Access.

2.3.2 Выбор и обоснование языка и среды программирования

В настоящее время существует огромное количество языков программирования, с которыми можно работать, у каждого есть свои достоинства и недостатки, которые выявляются в процессе работы с ними.

Правильный выбор языка программирования поможет создать простое решение с возможностью отладки, документирования и исправления ошибок.

Для создания удобной и многофункциональной информационной системы лучше всего использовать язык программирования, который можно будет связывать с другими языками, и при этом связывать программу с базами данных.

Borland Delphi — интегрированная среда разработки программного обеспечения для Microsoft Windows, Mac OS, iOS и Android на языке Delphi. Среда предназначена для быстрой (RAD) разработки прикладного программного обеспечения для операционных систем Windows, Mac OS X, а также IOS и Android. Благодаря уникальной совокупности простоты языка и генерации машинного кода, позволяет непосредственно, и, при желании, достаточно низкоуровневого взаимодействовать с операционной системой, а также с библиотеками, написанными на C/C++. Созданные программы не зависимы от стороннего программного обеспечения, как-то Microsoft.NET Framework, или Java Virtual Machine.

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

С# - относительно новый объектно-ориентированный язык программирования, разработанный компанией Microsoft. На нем можно писать приложения для Web, приложения под Windows, консольные программы, запускаемые из командной строки, работать с базами данных[10].

C# - это лишь один из языков платформы .NET Framework (помимо Visual Basic.NET, Visual C++.NET, J#, а с появлением Delphi 8/2005 еще и Delphi). В целом С# впитал в себя многое из того лучшего, что есть в самых разных языках программирования. Это простой, надежный, полностью объектно-ориентированный язык, используемый платформой .NET Framework.

Платформа .NET - это новая технология разработки программного обеспечения. В ее основе лежит идея универсального программного кода, который может быть выполнен любым устройством, вне зависимости от используемой этим устройством операционной системы (ОС должна поддерживать технологию .NET). Универсальность программного кода обеспечивается за счет предварительной, на этапе разработки, компиляции исходной программы в промежуточный код на языке CIL (Common Intermediate Language), который во время загрузки транслируется в выполняемую программу. Платформа .NET является независимой от используемых языков программирования. Можно использовать несколько .NET-совместимых языков даже в рамках одного проекта.

Данный язык является более простым и привлекательным для начинающих программистов, чем С++. На C# можно написать что угодно: веб-сервисы, мобильные ПО, серверные приложения и так далее Его основная среда программирования в которой он используется VisualStudio – очень удобная и понятная среда разработки приложений.

В комплект VisualStudioвходят следующие основные компоненты:

1. Visual C# - наязыке C# (Microsoft);

2. Visual F# - на F# (Microsoft Developer Division).

3. Visual Basic.NET - для разработки приложений на VisualBasic;

4. Visual C++ - на традиционном языке C++;

Функциональная структура среды включает в себя:

– редактор исходного кода;

– редактор форм, предназначенный для упрощённого конструирования графических интерфейсов;

– веб-редактор;

– отладчик кода;

– дизайнер схем баз данных;

– дизайнер классов.

Visual Studio также позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов (для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментов для прочих аспектов процесса разработки программного обеспечения).

Вывод: Так как выпускная квалификационная работа нацелена на разработку продукта, имеющего гибкий графический интерфейс, и основной задачей которого является работа с базой данных, выбирается язык C#. Проект будет реализован в среде программирования Visual Studio 2010.

## 2.4 Проектирование и Разработка базы данных

Выделяются следующие этапы разработки информационно-логической модели автоматизированной системы управления фитнес-клубом:

1. Построение функциональной модели предметной области в программной среде BPWin с помощью диаграмм:
2. Построение диаграмм IDEF0, которые имеют иерархическую структуру. На самом верхнем уровне располагается контекстная диаграмма, затем она декомпозируется;
3. Проектирование логической и физической модели данных в программной среде ERWin, с использованием метода «сущность-связь».

## 2.5 Построение функциональной модели предметной области в программной среде BPWin

На основе описания предметной области строится функциональная диаграмма организации проведений занятий в фитнес-клубе "ANTARES X-FIT PREMIUM".

Для построения функциональной схемы воспользуемся программой Bpwin, рисунок 7-8.

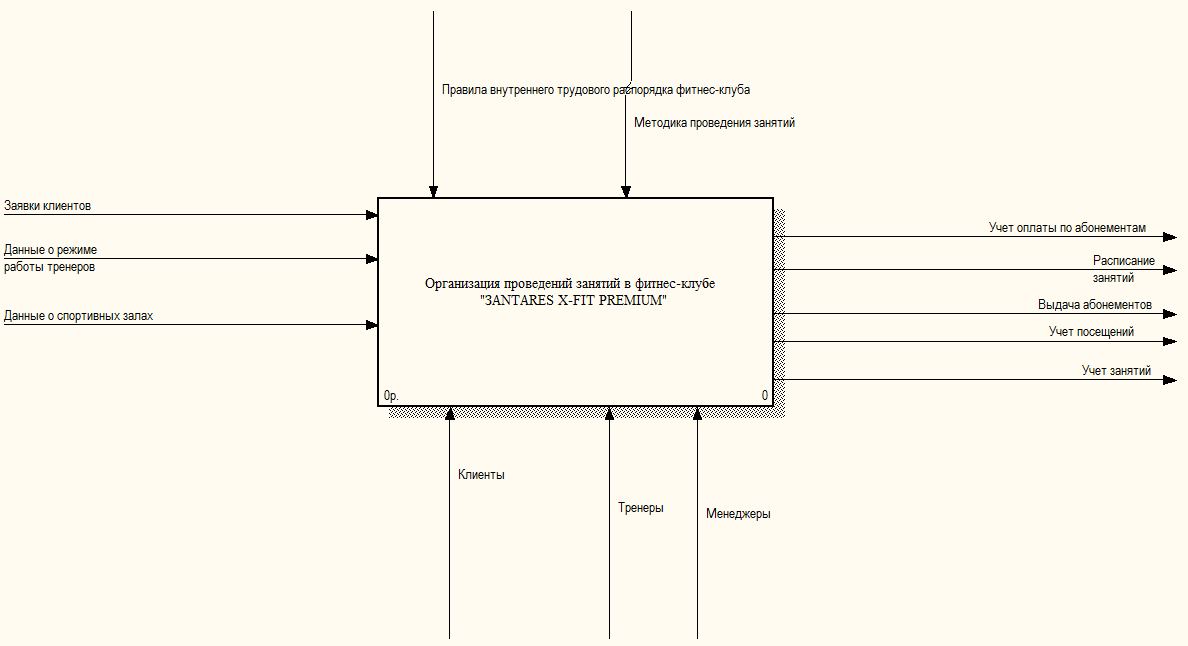


Рисунок – Контекстная диаграмма организации проведений занятий

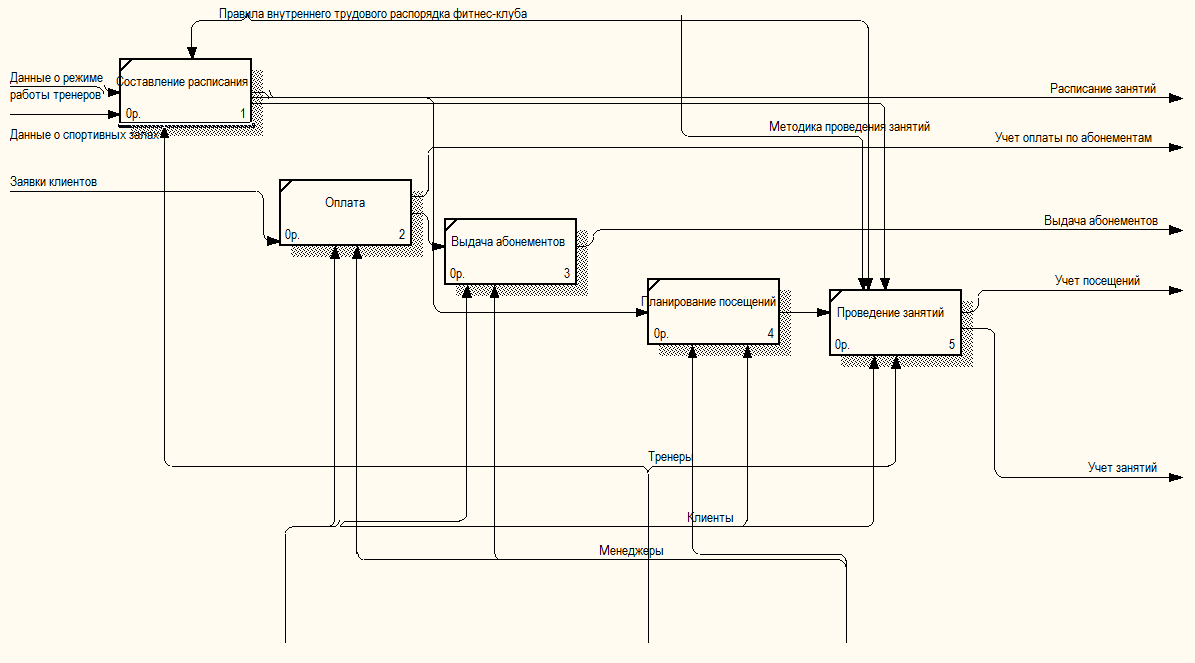


Рисунок – Декомпозиция контекстной диаграммы

## 2.6 Проектирование логической и физической модели данных

2.6.1 Разработка инфологической модели предметной области

Прежде чем разрабатывать ER-модель предметной области, необходимо выделяются основные сущности, характерные для данной предметной области, определяется информация, которая должна храниться в БД и строится инфологическая модель.

Выделяются базовые сущности этой предметной области:

Сущности: Абонемент, Вид Абонемента, Группа, Должность, Журнал занятий, Зал, Клиент, Направление, Период Действия, Расписание, Роль, Сотрудник ;.

Далее строится инфологическая модель БД в программе DIA. Инфологическая модель автоматизированной системы управления фитнес-клубом, отношения между сущностями приведены на рисунке 9.

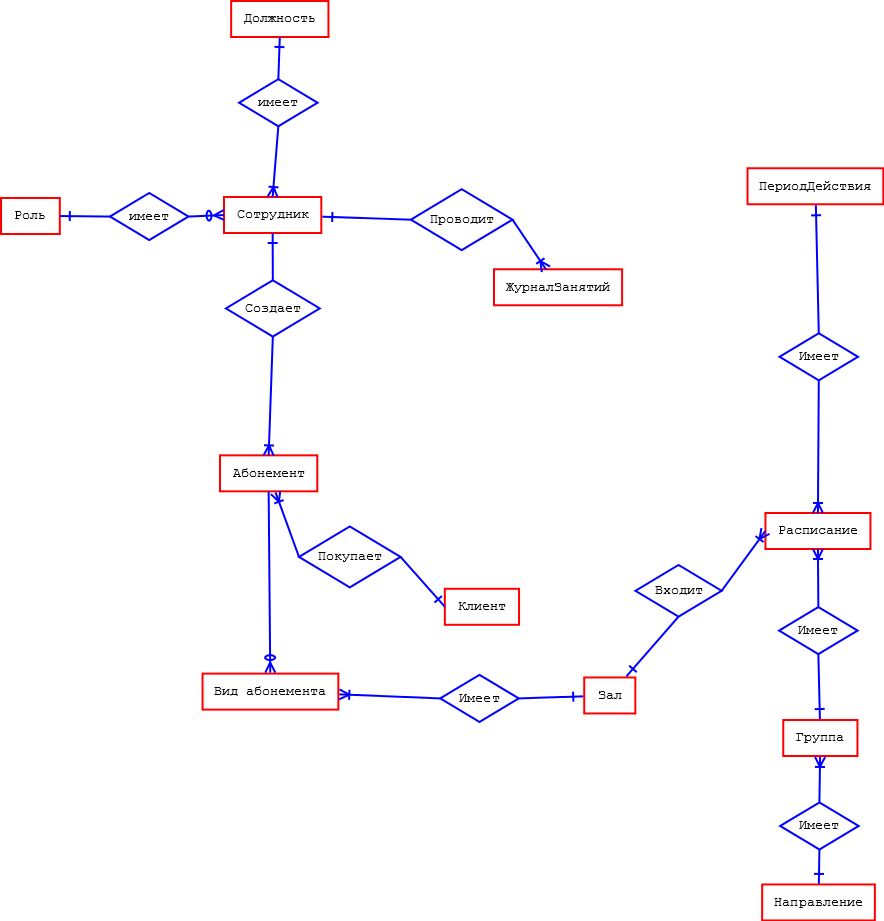


Рисунок – Инфологическая модель БД

Далее строится уточнённая ER–диаграмма, содержащая атрибуты сущностей, выделяются атрибуты сущностей этой предметной области, составляется словесное описание:

1. Сущность **Абонемент**: Код Абонемента, Код Клиента, Цена, Код Вида Абонемента, Скидка, Итого К Оплате, Оплачено, Дата Начала, Дата Конца;
2. Сущность **Вид Абонемента**: Код Вида Абонемента, Код Зала, Код названия, Цена, Количество Посещений;
3. Сущность **Группа**: Код Группы, Группа, Код Направления, Описание;
4. Сущность **Должность**: Код Должности, Должность;
5. Сущность **Журнал Занятия**: Код Занятия, Дата Занятия, Код Расписания, Выполнено, код сотрудника, который провел занятие;
6. Сущность **Зал**: Код Зала, Зал, Адрес, Описание, Площадь, Вместимость, Фото;
7. Сущность **Клиент**: Код Клиента, Клиент, Телефон, Email, Примечания;
8. Сущность **Направление**: Код Направления, Направление, Описание;
9. Сущность **Период Действия**: Код Периода, Дата Начала, Дата Конца, Название Периода;
10. Сущность **Расписание**: Код Расписания, Код Периода, Код Зала, День недели, Время Начала, Код Группы, Код Сотрудника, который должен проводить занятие;
11. Сущность **Роли**: Код Роли, Роль;
12. Сущность **Сотрудник**: Код Сотрудника, ФИО, Телефон, Образование, Работает, Имя Короткое, Код Роли, Логин, Код Должности, Фото.

Уточнённая ER–диаграмма, содержащая атрибуты сущностей приведена на рисунке (рисунок 10).

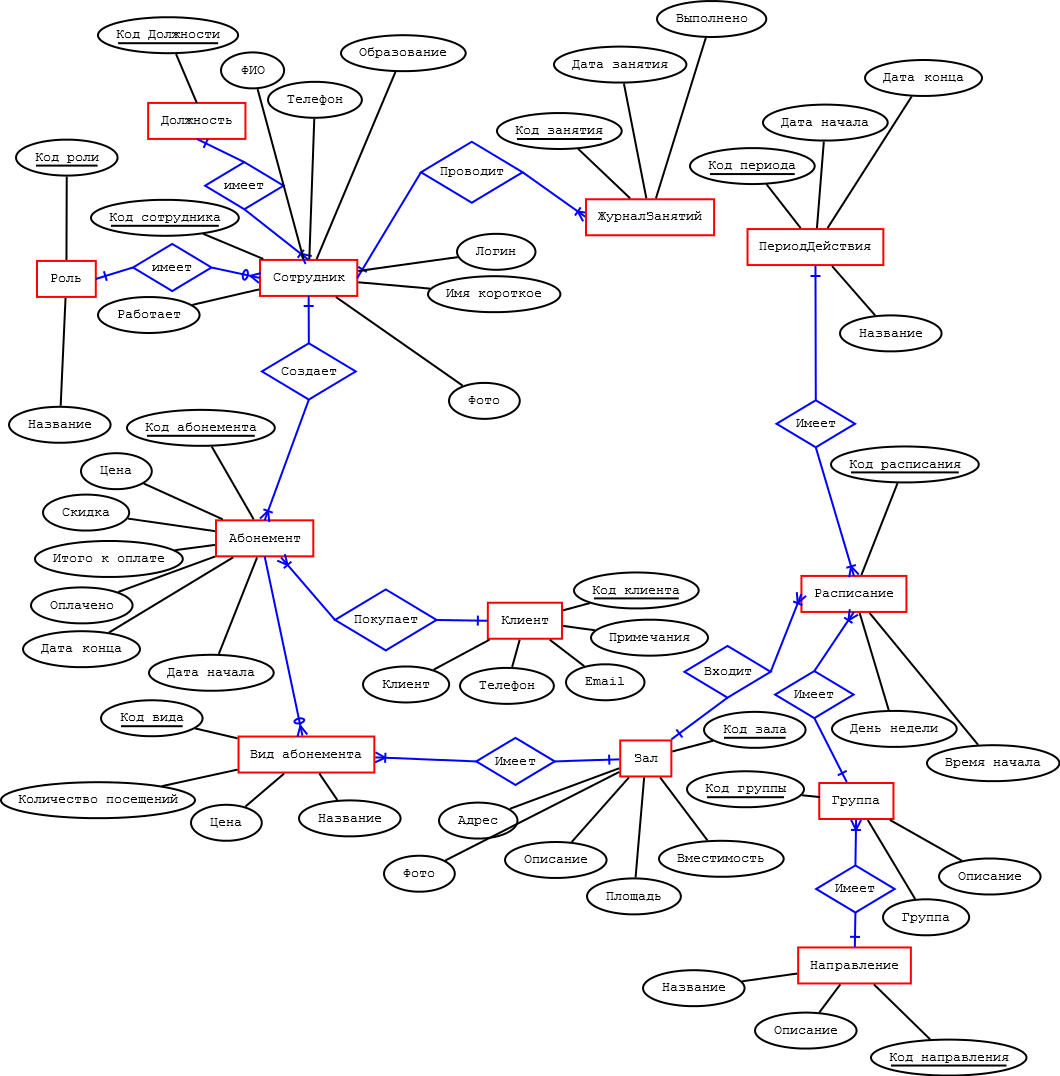


Рис. 10 Уточненная ER–диаграмма

2.6.2 Разработка даталогической модели БД

Преобразование ER–диаграммы в схему БД выполняется путем сопоставления каждой сущности и каждой связи, имеющей атрибуты, отношения (таблицы БД).

Преобразуются описание сущностей к 3-ей нормальной форме для того чтобы избежать избыточности данных и даются названия таблицам и атрибутам так, как того требует проектирование базы данных (без пробелов, недопустимых знаков).

1. Таблица: **Абонементы**
2. Таблица: **ВидыАбонементов**
3. Таблица: **Группы**
4. Таблица: **Должности**
5. Таблица: **ЖурналЗанятий**
6. Таблица: **НазванияВидовАбонементов**
7. Таблица: **Залы**
8. Таблица: **Клиенты**
9. Таблица: **Роли**
10. Таблица: **Направления**
11. Таблица: **Сотрудники**
12. Таблица: **ПериодДействия**
13. Таблица: **Расписание**
14. Таблица:  **ВремяНачала**

Преобразуются названия атрибутов в названия полей таблиц.

1. Таблица **Абонементы: КодАбонемента**, КодКлиента, Цена, КодЗала, КодВидаАбонемента, Скидка, ИтогоКОплате, Оплачено, ДатаОплаты, ДатаНачала, ДатаКонца;
2. Таблица **ВидыАбонементов: КодВидаАбонемента**, КодЗала, Название, Цена, КоличествоПосещений;
3. Таблица **Группы: КодГруппы,** Группа, КодНаправления, Описание;
4. Таблица **Должности: КодДолжности**, Должность;
5. Таблица **ЖурналЗанятий: КодЗанятия**, ДатаЗанятия, КодРасписания, Выполнено;
6. Таблица **Залы: КодЗала**, Зал, Адрес, Описание, Площадь, Вместимость;
7. Таблица **Клиенты: КодКлиента,** Клиент, Телефон, Email, Примечания;
8. Таблица **Роли**: КодРоли, Роль;
9. Таблица **Направления: КодНаправления,** Направление, Описание;
10. Таблица **Сотрудники: КодСотрудника,** ФИО, Телефон, Образование, ДопИнформация, НачалоДеятельности, Работает, Стаж, ИмяКороткое, КодРоли, Логин, КодДолжности;
11. Таблица **ПериодДействия: КодПериода,** ДатаНачала, ДатаКонца, НазваниеПериода;
12. Таблица **Расписание: КодРасписания,** КодПериода, КодЗала, Понедельник, Вторник, Среда, Четверг, Пятница, Суббота, Воскресенье, ВремяНачала, КодГруппы, КодСотрудника.

Детальные характеристики даталогической модели сводятся в таблицы 5-16. Ключевые поля выделены жирным курсивом.

Таблица 5 – ВидыАбонементов

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| ***КодВидаАбонемента*** | ***Счетчик*** |
| КодЗала | 32-битное целое поле (типа integer) |
| КодНазвания | Текстовый |
| Цена | 32-битное целое поле (типа integer) |
| КоличествоПосещений | 32-битное целое поле (типа integer) |

Таблица 6 – Абонементы

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип Данных | |
| ***КодАбонемента*** | ***Счетчик*** | |
| КодКлиента | 32-битное целое поле (типа integer) | |
| Цена | 32-битное целое поле (типа integer) | |
| КодВидаАбонемента | 32-битное целое поле (типа integer) | |
| Скидка | 16-битное беззнаковое целое поле | |
| ИтогоКОплате | 32-битное целое поле (типа integer) | |
| Оплачено | Логический | |
| ДатаНачала | Дата/время | |
| ДатаКонца | Дата/время | |
| КодСотрудника | | 32-битное целое поле (типа integer) |

Таблица 7 – Группы

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| ***КодГруппы*** | ***Счетчик*** |
| Группа | Текстовый |
| КодНаправления | 32-битное целое поле (типа integer) |
| Описание | Поле Memo (строка неограниченной длины) |
| Цвет | 32-битное целое поле (типа integer) |

Таблица 8 – ЖурналЗанятий

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| ***КодЗанятия*** | ***Счетчик*** |
| ДатаЗанятия | Дата/время |
| КодРасписания | 32-битное целое поле (типа integer) |
| Выполнено | Логический |
| КодСотрудника | 32-битное целое поле (типа integer) |
| Количество | 32-битное целое поле (типа integer) |

Таблица 9 – Должности

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| ***КодДолжности*** | ***Счетчик*** |
| Должность | Текстовый |

Таблица 10 – Залы

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| ***КодЗала*** | ***Счетчик*** |
| Зал | Текстовый |
| Адрес | Текстовый |
| Описание | Поле Memo (строка неограниченной длины) |
| Площадь | Числа с плавающей запятой (типа double) |
| Вместимость | 32-битное целое поле (типа integer) |
| Фото | Поле объекта OLE |

Таблица 12 – Клиенты

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| ***КодКлиента*** | ***Счетчик*** |
| Клиент | Текстовый |
| Телефон | Текстовый |
| Email | Текстовый |
| Примечания | Поле Memo (строка неограниченной длины) |

Таблица 13 – Направления

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| ***КодНаправления*** | ***Счетчик*** |
| Направление | Текстовый |
| Описание | Поле Memo (строка неограниченной длины) |

Таблица 14 – Роли

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| ***КодРоли*** | ***Счетчик*** |
| Роль | Текстовый |

Таблица 15 – ПериодДействия

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| ***КодПериода*** | ***Счетчик*** |
| ДатаНачала | Дата/время |
| ДатаКонца | Дата/время |
| НазваниеПериода | Текстовый |

Таблица 15 – Сотрудники

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| ***КодСотрудника*** | ***Счетчик*** |
| ФИО | Текстовый |
| Телефон | Текстовый |
| Образование | Поле Memo (строка неограниченной длины) |
| Работает | Логический |
| ИмяКороткое | Текстовый |
| КодРоли | 32-битное целое поле (типа integer) |
| Логин | Текстовый |
| КодДолжности | 32-битное целое поле (типа integer) |
| Пароль | Текстовый |

Таблица 16 – Расписание

|  |  |
| --- | --- |
| Имя поля | Тип Данных |
| КодРасписания | Счетчик |
| КодПериода | 32-битное целое поле (типа integer) |
| КодЗала | 32-битное целое поле (типа integer) |
| НомерДняНедели | 32-битное целое поле (типа integer) |
| ВремяНачала | Текстовый |
| КодГруппы | 32-битное целое поле (типа integer) |
| КодСотрудника | 32-битное целое поле (типа integer) |

Связи между сущностями представлены в таблице 17.

Таблица 17

Связи между сущностями

|  |  |  |  |
| --- | --- | --- | --- |
| **Сущность1** | **Тип связи** | **Связь** | **Сущность2** |
| Должность | Один – ко многим | Имеет | Сотрудник |
| Сотрудник | Один – ко многим | Регистрирует | Клиент |
| Роль | Один – ко многим | Имеет | Сотрудник |
| Сотрудник | Один – ко многим | Создает | Абонемент |
| Сотрудник | Один – ко многим | Проводит занятие | ЖурналЗанятий |
| ВидАбонемента | Один – ко многим | Имеет | Название |
| Клиент | Один – ко многим | Покупает | Абонемент |
| Зал | Один – ко многим | Имеет | ВидАбонемента |
| Зал | Один – ко многим | Входит | Расписание |
| Расписание | Один – ко многим | Детализируется | ЖурналЗанятий |
| Группа | Один – ко многим | Имеет | Расписание |
| Направление | Один – ко многим | Имеет | Группа |

2.6.3 Построение логической модели базы данных в программе Erwin

На основании разработанной даталогической схемы строится логическая модель БД в программе ERWin. Логическая модель базы данных представлена на рисунке 12.

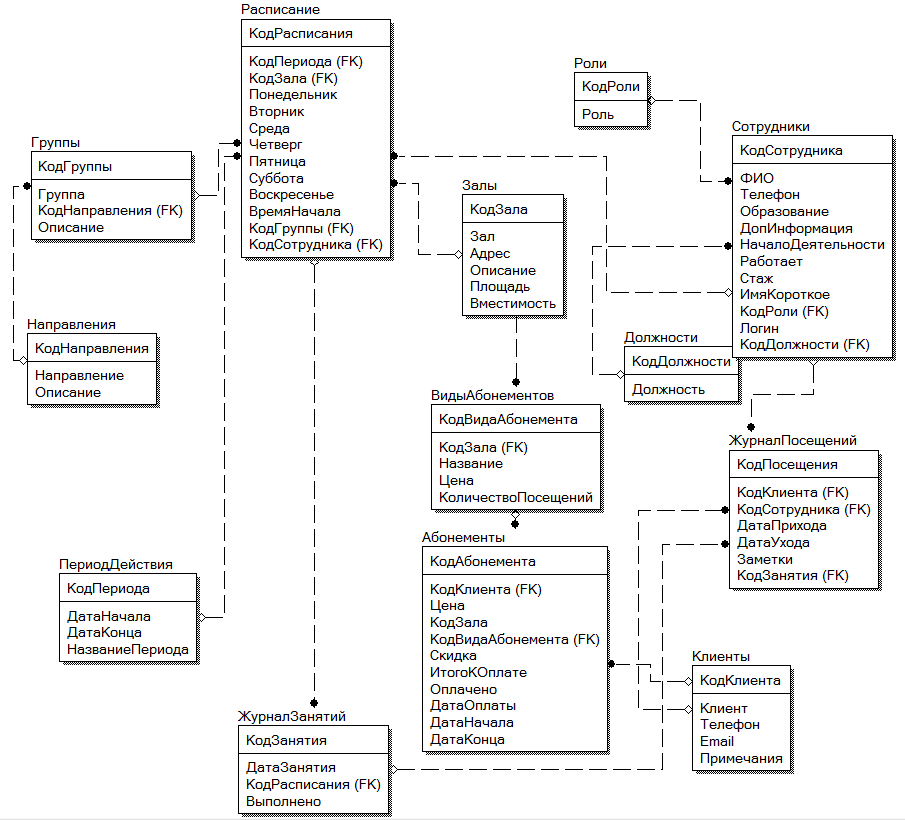


Рисунок – Логическая модель базы

2.6.4 Построение физической модели базы данных в программе Erwin

Физическая модель базы данных представлена на рисунке 12.

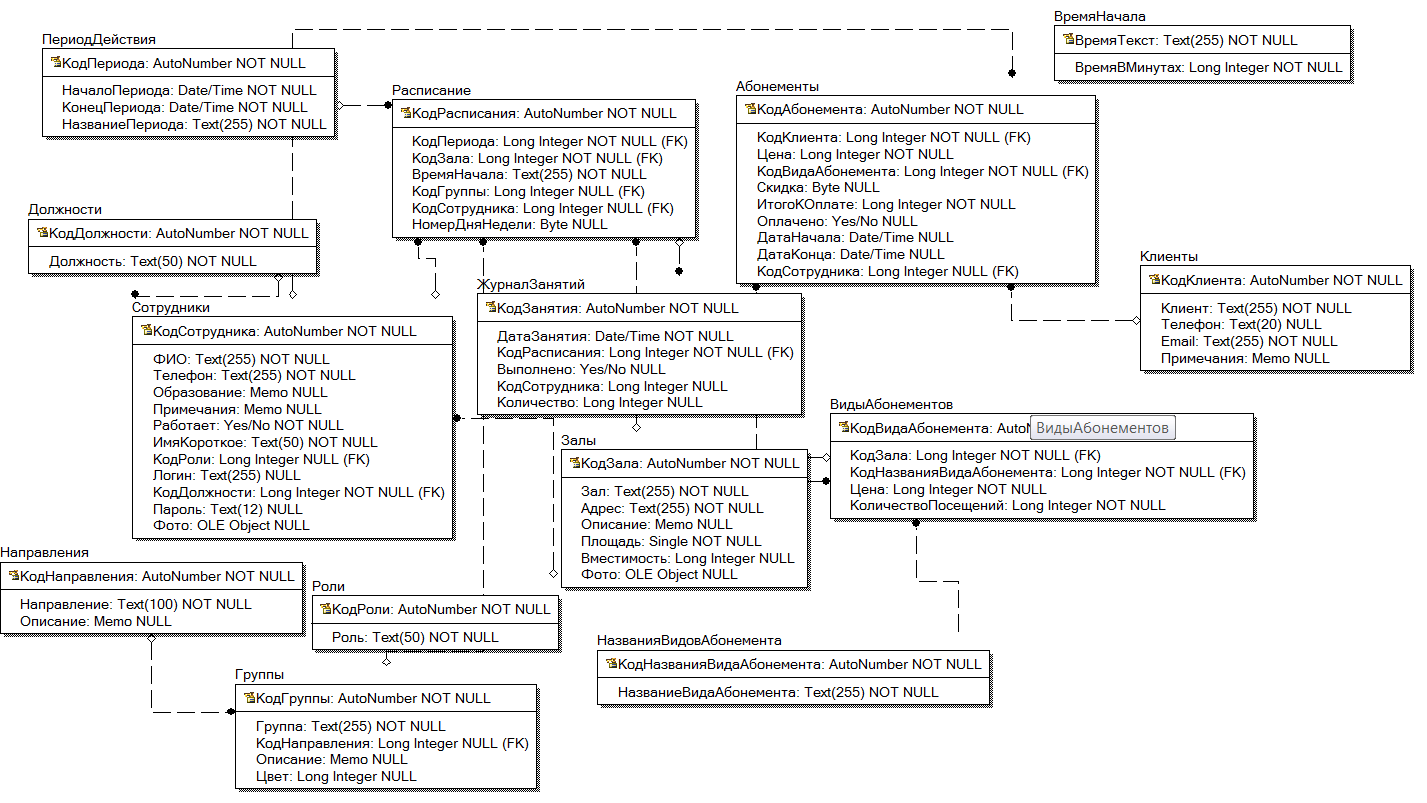


Рисунок – Физическая модель базы данных

На основании разработанной схемы генерируется база данных BaseFit.mdb. Сначала создается пустую базу Access 2000-2003 BaseFit.mdb. Производится переключение на физическую модель. Производится соединение с базой данных: вкладка Database - Database conection... Вводится имя Admin. Далее указывается путь к базе Access. Нажимается Connect. Далее пустая база заполняется таблицами: Tools-Forward Engineer-Schema generatio: generate.

Схема данных базы BaseFit.mdb представлена на рисунке 13.

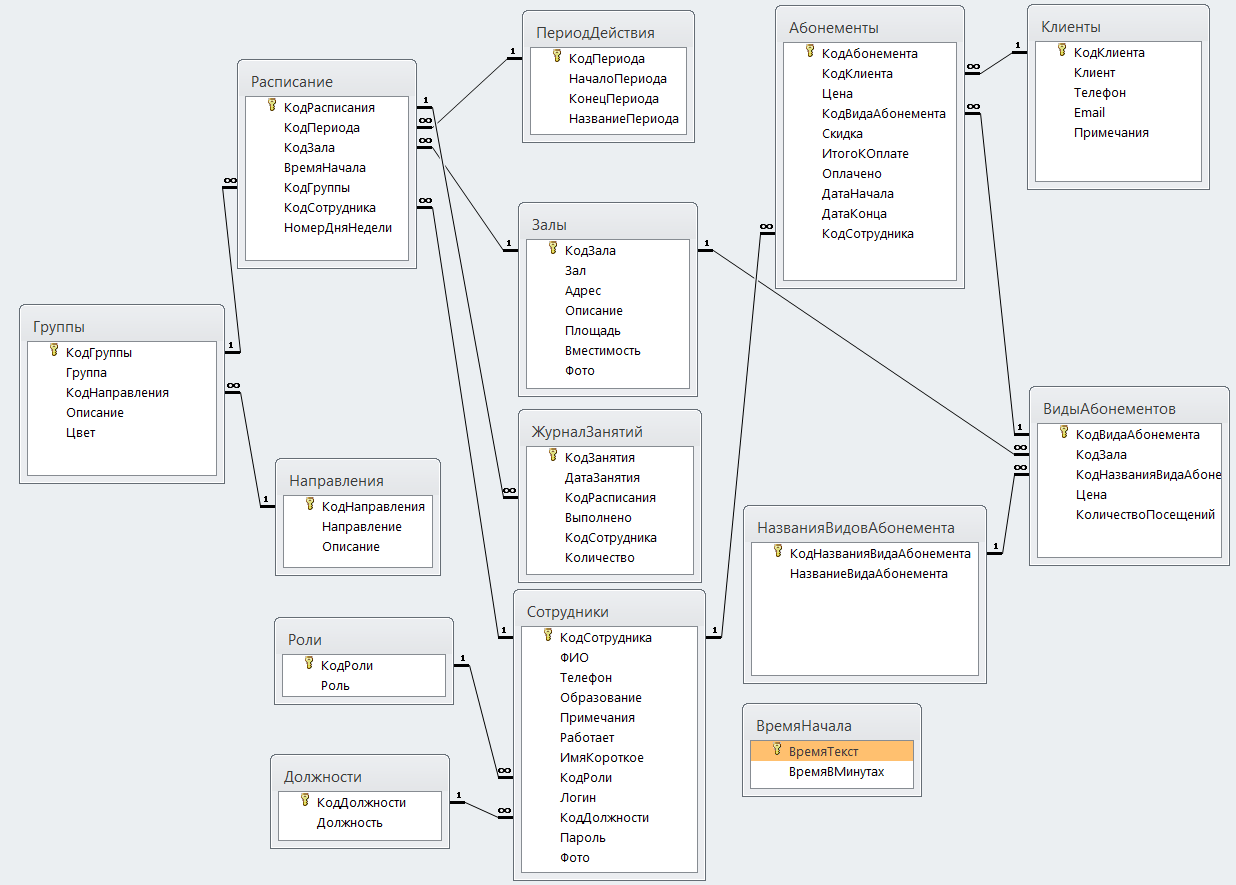


Рисунок – Схема данных базы BaseFit.mdb

Исходя из постановки задачи определяется функциональная структура системы. Функциональная структура автоматизированной системы управления фитнес-клубом представлена на рисунке 15.

## 2.7 Функциональная структура системы

Функциональная структура системы представлена на рисунке 14.



Рисунок 14 - Функциональная структура системы учета абонементов и занятий фитнес-клуб "ANTARES X-FIT PREMIUM"

Далее создается проект в Visual Studio 2010 FitnessClub.sln. В ходе разработки приложения созданы следующие модули:

1. модуль авторизации (LoginForm.cs) - обеспечивает идентификацию и аутентификацию пользователей;
2. модуль пользовательского интерфейса (Form1.cs) – отображает данные;
3. модуль формирования абонемента (fabonement.cs) - обеспечивает введение необходимых данных при формировании абонемента;
4. модуль выбора (formsprcase.cs) - обеспечивает выбор данных;
5. модуль запросов к БД (ClassBaseFIT.cs) - позволяет получать записи из БД, удалять, добавлять записи и редактировать;
6. модули редактирования и добавления записей справочников (fgrupp.cs, fklient.cs, fnapr.cs, fobject.cs, FormSpr.cs, fsotrudnik.cs, fperiod.cs) – позволяют работать с записями справочников;

* модуль вывода информации о проекте (fabout.cs) - обеспечивает вывод информации о приложении;

Схема классов разработанного приложения приведена на рисунке 15.

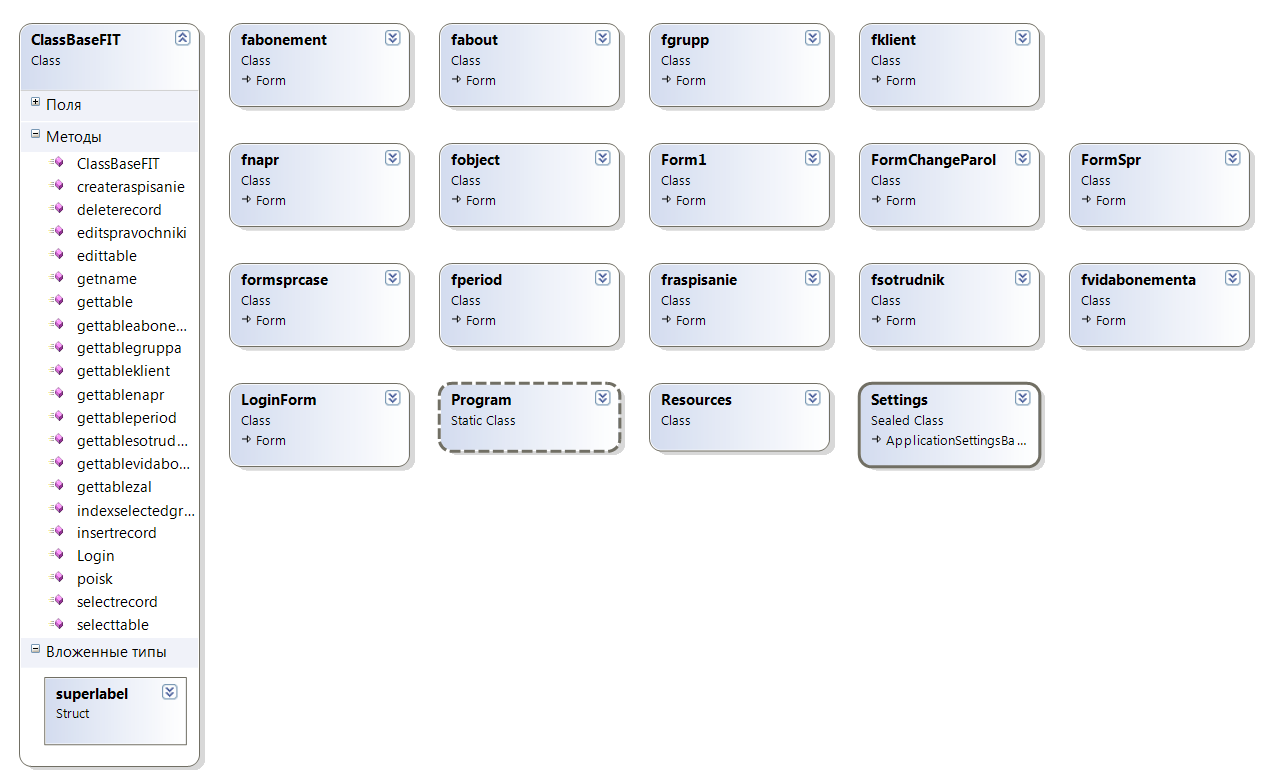


Рисунок – Схема классов

Аутентификация пользователей осуществляется с помощью модуля авторизации, обеспечивающего разграничение прав использования программного комплекса. Подключение к БД происходит через модуль связи, далее осуществляется переход в модуль пользовательского интерфейса, где отображаются данные.

# 3 Разработка технической документации для сопровождения программного приложения

Полное наименование программной разработки: «Приложение с графическим интерфейсом «Автоматизированная система управления фитнес-клубом ANTARES X-FIT PREMIUM»»

## 3.1 Постановка информационной задачи

Назначением разрабатываемой информационной системы фитнес-клуба "ANTARES X-FIT PREMIUM:

1. хранение информации об абонементах клиентов на групповые занятия;
2. хранение информации об оплатах по абонементам;
3. упрощение составления расписания занятий
4. Организация и ведение журнала занятий
5. Ведение справочников

Приложение должно обеспечивать бессбойную работу в круглосуточном режиме. Все ошибки при работе приложения должны приводить к выводу предупреждающего сообщения, а не к фатальному прекращению работы системы. Все данные вводимые пользователями системы должны проверяться на корректность.

## 3.2 Организация входных и выходных данных

Входные данные поступают в систему от пользователя при входе и проверяются на корректность. После данные обрабатываются, записываются в базу данных и выводятся на экран. Алгоритм работы системы в общем виде:

1. Финансовый администратор информирует клиента о предоставляемых услугах фитнес-клуба "ANTARES X-FIT PREMIUM".
2. Клиент определяется с выбором абонемента.
3. Финансовый администратор принимает, регистрирует клиента, оформляет, регистрирует и выдает абонемент, принимает от клиента квитанцию об оплате (оплата производится у кассира).
4. Клиент знакомится с расписанием и определяется с датами посещения занятий.
5. Инструктор проводит занятие по выбранному направлению в выбранном зале, согласно расписания.
6. После проведения занятия, фитнес-менеджер или сам инструктор делает запись в журнале проведения занятий.

Входные данные, требуемые непосредственно от пользователя это абонемент, выбранный зал, вид занятия, расписание, а также логин и пароль для входа в программу для сотрудников. Остальные данные хранятся в базе данных. В базе данных хранятся данные о сотрудниках, должностях, ролях, клиентах, типах занятий и других данных.

Выходными данными, основанными на введённой информации финансовым администратором, фитнес-менеджером являются: успешно созданный абонемент, регистрация проведенного занятия в журнале, успешный вход в программу пользователя для дальнейшего пользования программой. Основной режим использования системы — регулярная работа.

Необходимо реализовать защиту паролем для всех пользователей.

## 3.3 Требования к функциям (задачам), выполняемым системой

Разрабатываемое ПО должно обеспечивать Осуществление многопользовательского режима.

Для всех пользователей должен быть организован вход по логину и паролю, предоставлена возможность смены пароля.

Требования к функциям, выполняемым системой:

1. Функции Финансового администратора
2. Регистрация клиента
3. Оформление абонемента
4. Регистрация оплаты по абонементу
5. Просмотр расписания
6. Функции Фитнес-менеджера
7. Составление расписания
8. Регистрация занятий в журнале проведенных занятий
9. Получение отчетов по занятиям
10. Функции Администратора на ресепшене
11. Просмотр расписания
12. Просмотр и редактирование журнала занятий
13. Функции Бухгалтера
14. Просмотр расписания
15. Получение отчетов по абонементам, оплатам
16. Функции Директора, Заместителя директора
    1. Просмотр расписания
    2. Получение отчетов по абонементам, оплатам
    3. Просмотр журнала проведенных занятий
17. Функции администратора
18. Редактирование справочников
19. Функции инструктора
20. Просмотр, составление расписания
21. Просмотр журнала проведенных занятий

В информационное обеспечение входит база данных, состоящая из таблиц. В них хранится информация о абонементах, сотрудниках, клиентах, направлениях занятий, группах, залах, должностях и другая информация.

## 3.4 Формализованное описание процесса оформления абонемента

Формализованное описание процесса регистрации (создания) и исполнения заявки можно представлено на рисунке 16.

Процесс регистрации клиента и оформления абонемента

X={А, В, С, D, E, F} J={ G, H, I}

Входные параметры Выходные параметры

Рисунок 16 – Формализованное описание процесса создания и оформления абонемента

**Входные параметры: Х= {А, В, С, D, E, F}**

где А - ФИО сотрудника;

В – роль пользователя (финансовый администратор);

С – абонемент;

D – клиент;

E – спецификация абонемента;

F – ФИО сотрудника исполнителя.

**Выходные параметры: J ={**G, H, R**}**

G – регистрация клиента;

Н – регистрация абонемента

I – счет на оплату абонемента;

## 3.5 Требования к надежности. Условия эксплуатации и требования к составу и параметрам технических средств

Приложение обеспечивает бессбойную работу в круглосуточном режиме. Все ошибки при работе приложения должны приводить к выводу предупреждающего сообщения, а не к фатальному прекращению работы системы. Все данные вводимые пользователями системы должны проверяться на корректность.

Для успешного запуска и выполнения программы необходимы следующие программные и аппаратные характеристики:

* процессор с тактовой частотой 1500 МГц;
* оперативная память не менее 512МБ;
* не менее 20 ГБ свободного пространства на жестком диске;
* видеокарта и монитор SVGA с разрешением не менее 1024x768 точек;
* мышь;
* клавиатура;
* привод CD-ROM;
* операционная система Windows 7 и выше.

Для установки приложения необходимо наличие привода CD-ROM либо USB-порт.

## 3.6 Требования к интерфейсу. Функциональная структура информационной системы

Программное обеспечение должно иметь интуитивно понятный интерфейс, рассчитанный на пользователя средней квалификации в сфере компьютерной грамотности.

Основные модули приложения:

1. модуль авторизации - обеспечивает идентификацию и аутентификацию пользователей;
2. модуль пользовательского интерфейса – отображает данные по расписанию, спецификации абонементов, справочникам;
3. модуль формирования абонемента - обеспечивает введение необходимых данных при формировании абонемента;
4. модуль запросов к БД - позволяет получать записи из БД, удалять, добавлять записи и редактировать;
5. модуль создания, редактирования и просмотра расписания – позволяет создавать, редактировать и просматривать расписание занятий;
6. модуль связи с БД – обеспечивает подключение к базе данных.

## 3.7 Требования к маркировке и упаковке

Программа должна поставляться на диске в виде исполняемого (еxе) файла, базы данных, документации и проекта. На диске должна быть наклейка с надписью «Приложение с графическим интерфейсом «Автоматизированная система управления фитнес-клубом ANTARES X-FIT PREMIUM»». Диск должна быть упакован в пластиковую коробку.

# 4 Тестирование программного продукта

## 4.1 Авторизация

Окно авторизация в программе представлено на рисунке 17.

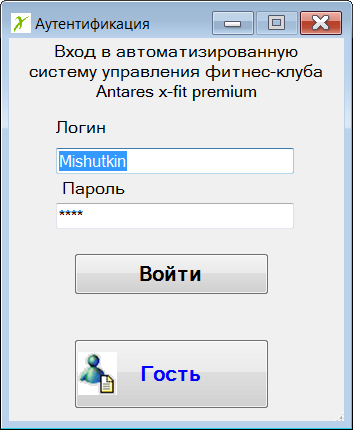


Рисунок – Окно авторизации

При входе в программу сотрудника сначала проверяется путь к базе, затем наличие данного логина в таблице Сотрудник и соответствие пароля. Если не выполняются какие-либо пункты проверки – выдается сообщение (рис. 18).

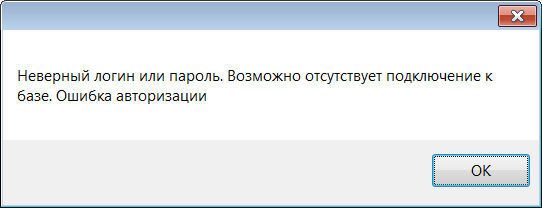
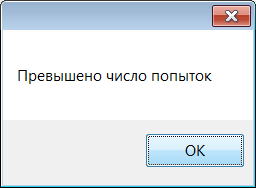
 

Рисунок – Сообщение

Для доступа только к просмотру расписания осуществляется вход по кнопке Гость.

## 4.2 Главное окно. Вкладка Заставка

Главное окно программы (рис.19).



Рисунок – Главное окно. Вкладка Заказы

Интерфейс главного окна выполнен с помощью элемента TabControl. Окно имеет 7 вкладок:

1. Заставка
2. Расписание;
3. Абонементы;
4. Журнал занятий;
5. Клиенты;
6. Сотрудники;
7. Справочники.

Для гостя возможен доступ только к расписанию в режиме просмотра, поэтому вкладка Заставка для гостя имеет вид (рис. 20).



Рисунок – вкладка Заставка для гостя

Переключение между вкладками осуществляется с помощью текстового меню Переход и боковой панели управляющих кнопок (рис. 21). С помощью меню Вид можно скрыть надписи на боковой кнопочной панели для увеличения рабочей области (рис. 22).

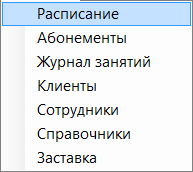
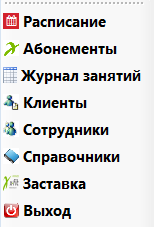
  

Рисунок – Управляющие элементы переключения между вкладками



Рисунок – Меню Вид: для изменения режима отображения текста на боковой панели

В нижней части окна выводится служебная информация (рис. 23).



Рисунок – служебная информация

Доступные функции на вкладке Заставка: Для Гостя переход к расписанию, для остальных пользователей: переход к различным вкладкам в зависимости от роли.

## 4.3 Главное окно. Вкладка Расписание

Интерфейс вкладки Расписание представлен на рисунке 24.

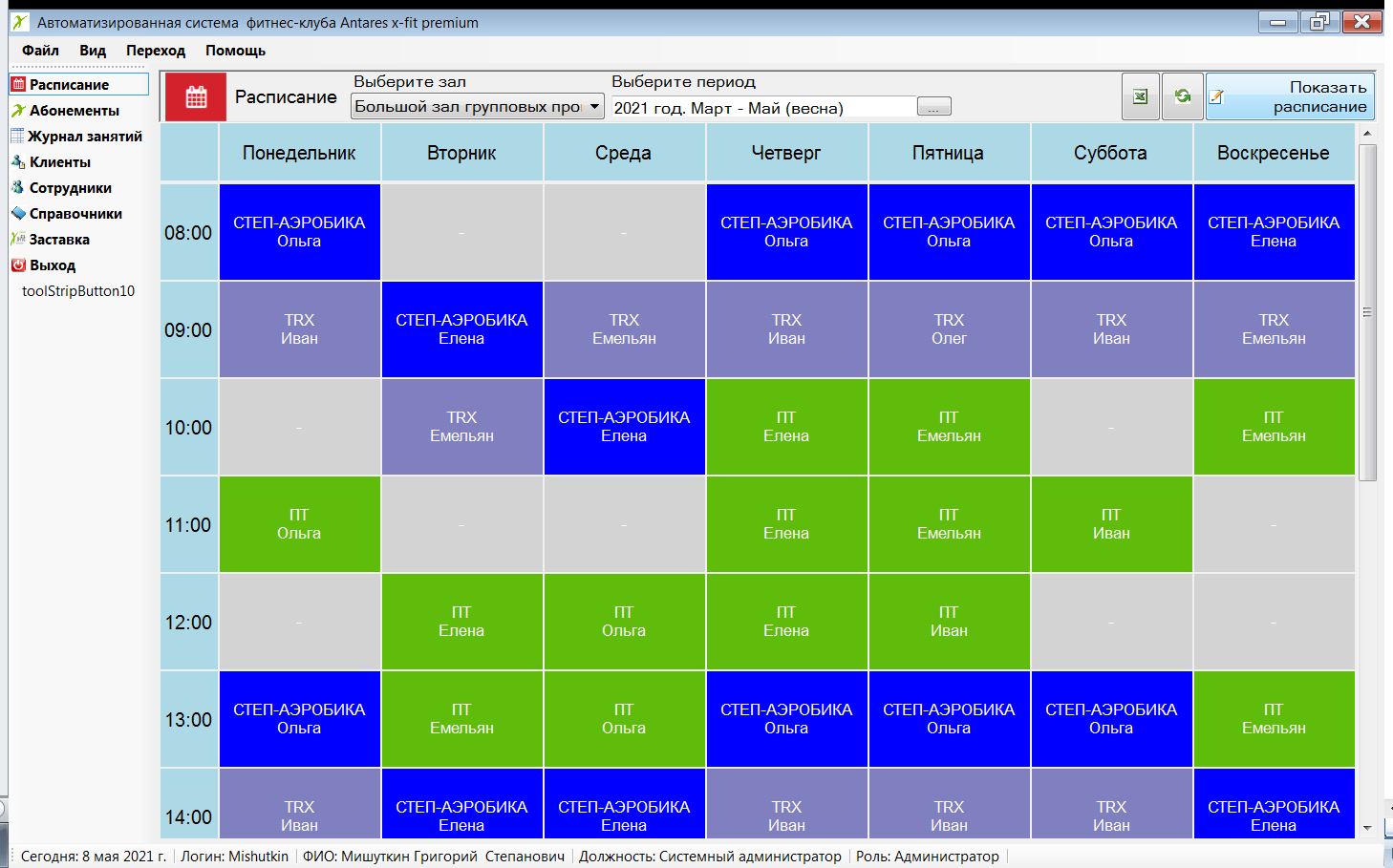


Рисунок – Главное окно. Вкладка Расписание

Доступные функции на вкладке Расписание:

1. Просмотр расписания по выбранному залу и периоду;
2. Переход к редактированию выбранного расписания за определенный день недели и определенное время в отдельном окне;
3. Экспорт данных в Excel.

При После нажатия на кнопку расписания, если для данного зала и периода отсутствует расписание будет выведен запрос (рисунок 25) и будет создано новое расписание.

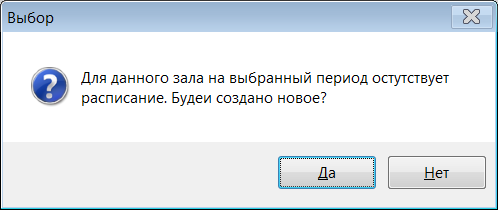
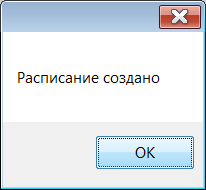
 

Рисунок – Запрос на создание расписания

Расписание редактируется в отдельном окне: форма Расписание – рисунок 26.

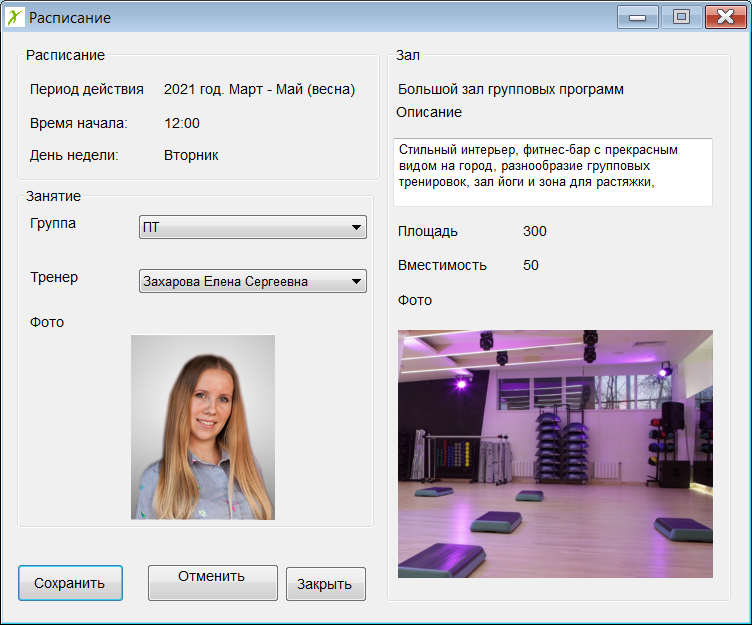


Рисунок – Форма Расписание

В данном окне пользователю необходимо задать группу и выбрать тренера (период действия, зал, время и день недели задается автоматически при формировании расписания). Для гостя доступ к редактированию расписания ограничен. Вид формы Расписания для гостя приведен на рисунке 27.

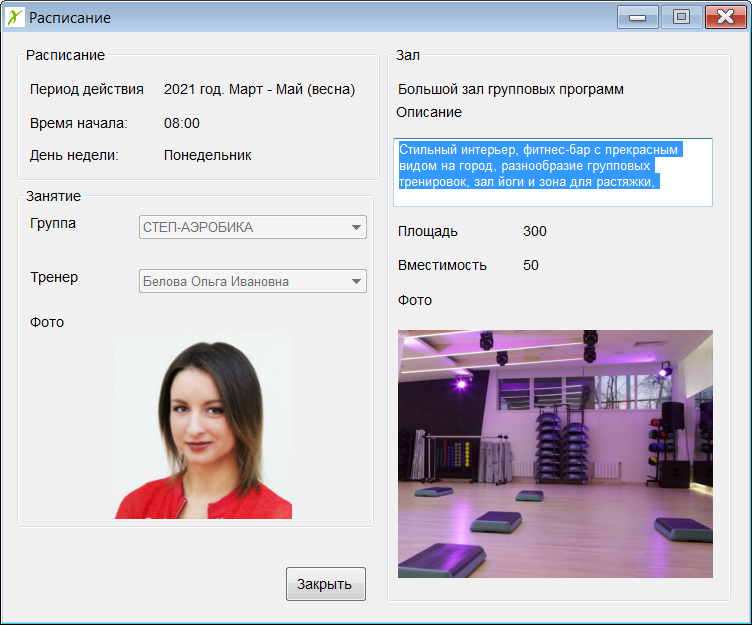


Рисунок – Вид формы расписания для гостя

При нажатии кнопки Сохранить для данного расписания будут созданы записи в таблице ЖурналЗанятий с кодом расписания, датой.

При нажатии кнопки Отменить занятие будут удалены сведения о группе и тренере в таблице Расписания и удалены соответствующие записи в таблице ЖурналЗанятий, перед этим будет выведен запрос – рисунок 28.

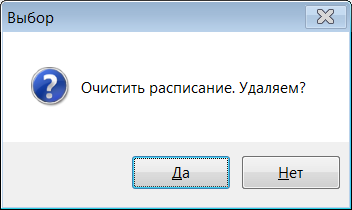


Рисунок – Запрос на удаление расписания

Пример экспорта расписания в MS Excel приведен на рисунке .

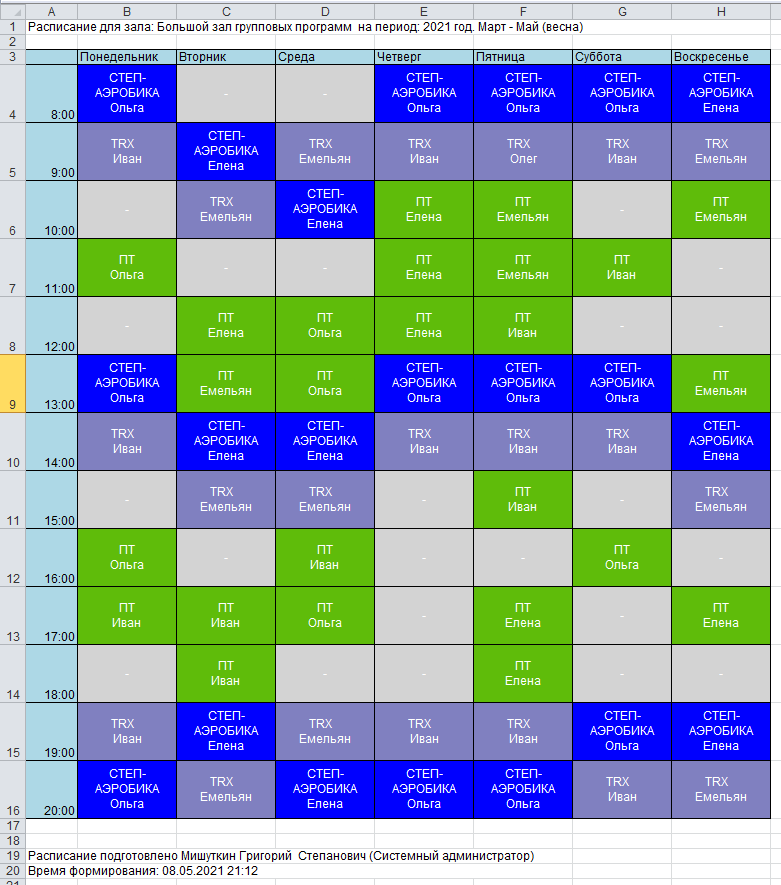


Рисунок – Пример экспорта расписания в MS Excel

## 4.4 Главное окно. Вкладка Абонементы

Интерфейс вкладки Абонементы представлен на рисунке 30.

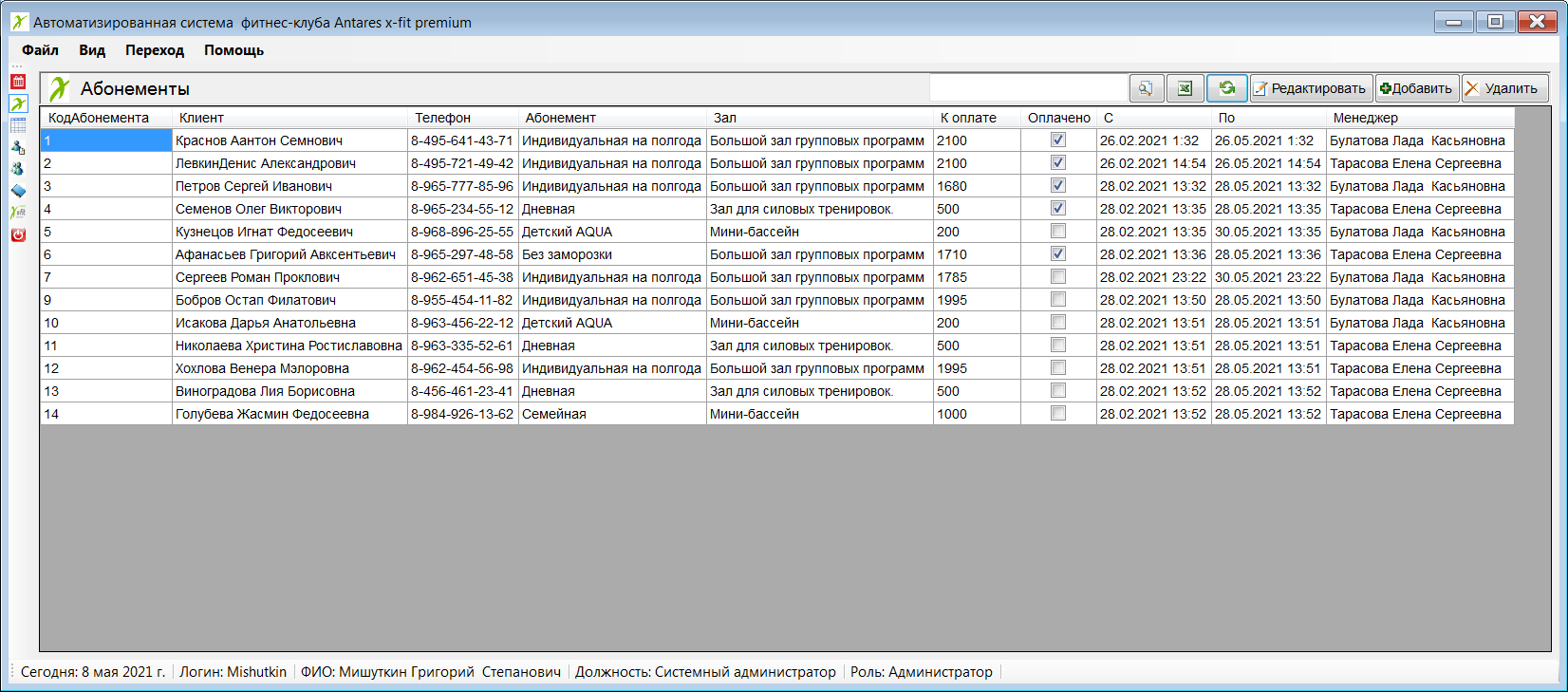


Рисунок – Главное окно. Вкладка Абонементы

Доступные функции на вкладке Абонементы:

1. Поиск по всем полям таблицы Абонементы (поля интерпретируются как текстовые);
2. Удаление Абонемента (удалять можно только не оплаченные абонементы);
3. Переход к редактированию или просмотру Абонемента в отдельном окне;
4. Переход к добавлению Абонемента в отдельном окне;
5. Обновление таблицы Абонементы;
6. Экспорт данных в Excel.

При удалении Абонемента, в случае если абонемент оплачен выводится сообщение об отказе – рисунок . Если удаление возможно – будет выведен запрос на подтверждение удаления – рисунки 31-32.

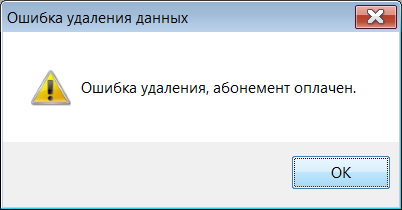


Рисунок – Сообщение об отказе

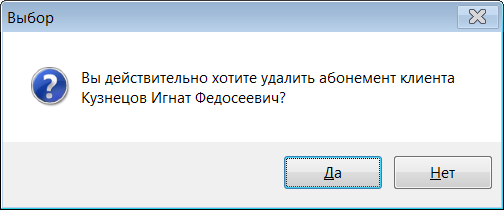


Рисунок - запрос на подтверждение удаления

Форма просмотра, редактирования, добавления абонемента представлена на рисунках 33-34.

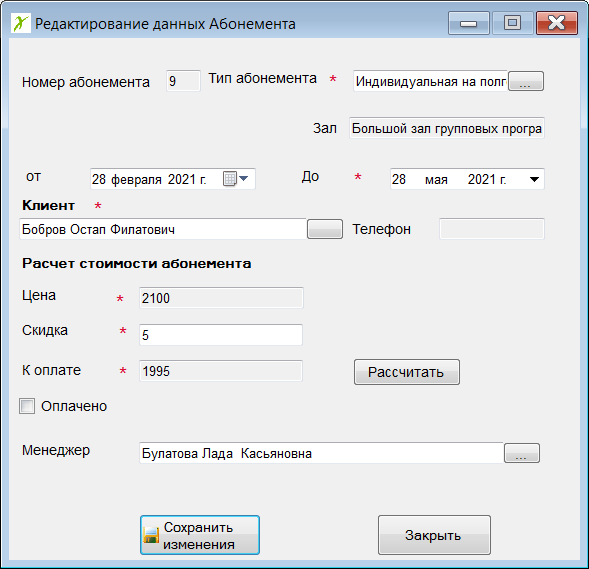


Рисунок – Форма абонемента в режиме редактирования

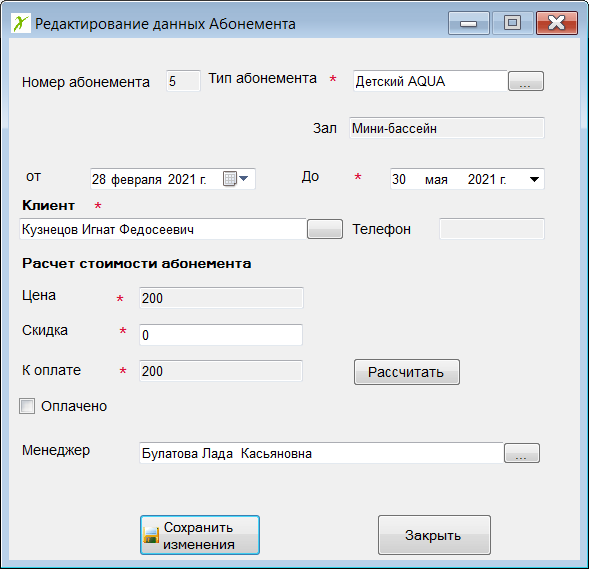
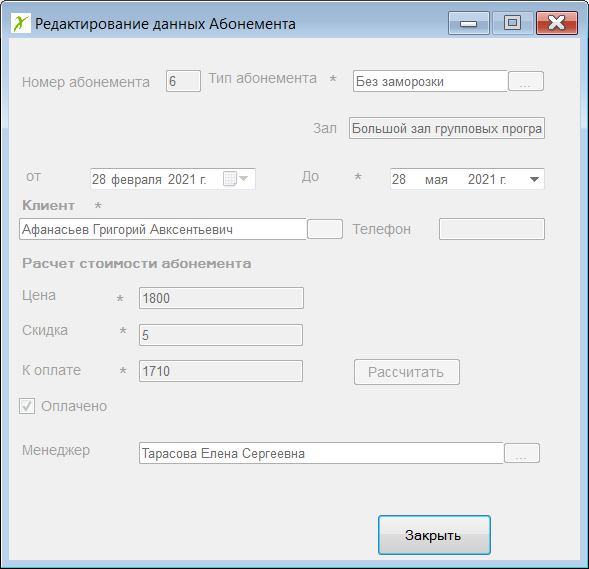
 

Рисунок – Форма абонемента в режиме добавления и просмотра

Выбор клиента, типа абонемента, менеджера осуществляется с помощью универсальной формы выбора – рисунок .35

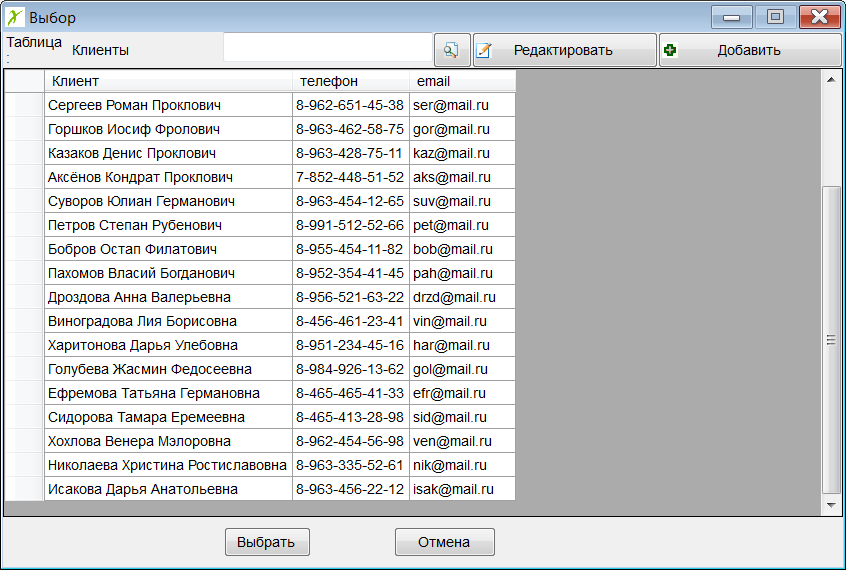


Рисунок – Универсальная форма выбора

После нажатия кнопки сохранить осуществляется проверка – если введенные сведения будут неполные или ошибочные будут выведены предупреждающие сообщения рисунок 36.

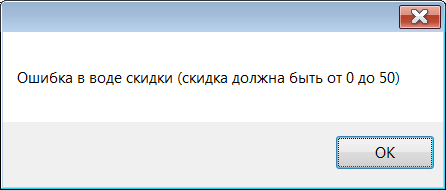
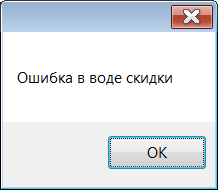
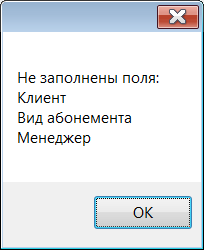


Рисунок – Предупреждающие сообщения

Пример экспорта таблицы Абонементы в MS Excel приведен на рисунке 37.

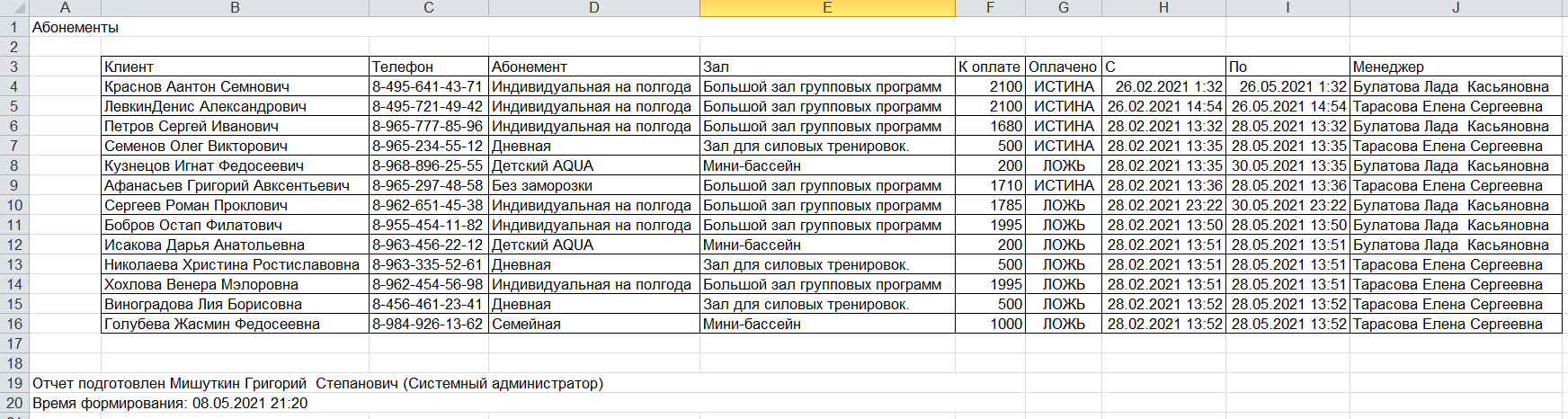


Рисунок – Пример экспорта таблицы Абонементы в MS Excel

## 4.5 Главное окно. Вкладка Журнал занятий

Интерфейс вкладки Журнал занятий представлен на рисунке 38.

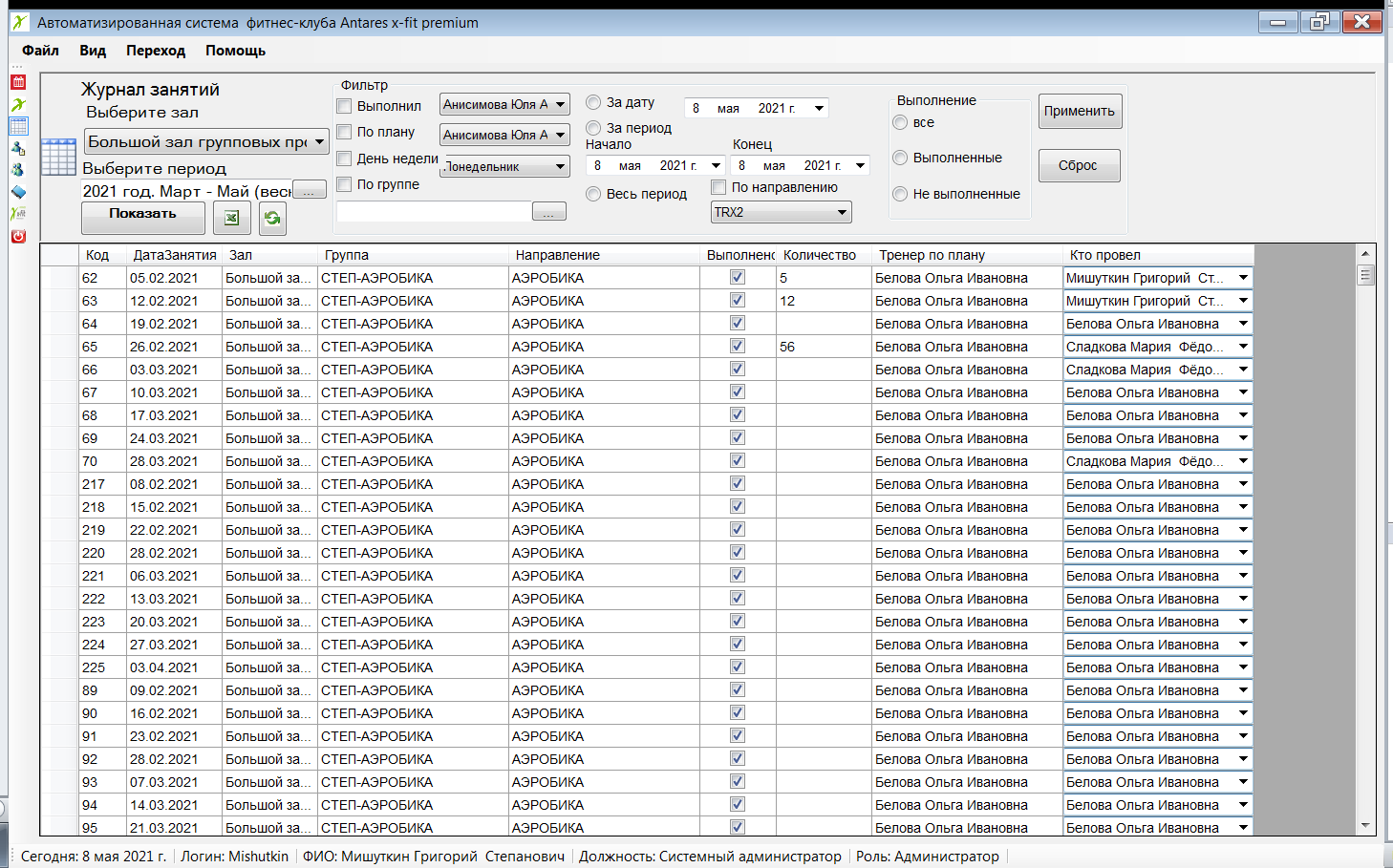


Рисунок – Интерфейс вкладки Журнал занятий

Доступные функции на вкладке Журнал занятий:

1. Просмотр журнала занятий по выбранному залу и периоду;
2. Поиск по всем полям таблицы Журнал занятий (поля интерпретируются как текстовые)
3. Редактирование данных прямо в таблице: поля Выполнено, Количество и Кто провел
4. Фильтрация данных по следующим параметрам:
   1. Кто провел занятие;
   2. Кто должен проводить по плану;
   3. День недели (понедельник, вторник…);
   4. По группе;
   5. За выбранную дату;
   6. За выбранный период;
   7. По направлению;
   8. Проведенные, не проведённые;
5. Обновление таблицы Журнал занятий
6. Экспорт данных в Excel.

Особенности редактирования:

1. Если Выполнено ставится в значение Истина и если не заполнено поле кто провел: полю кто провел автоматически ставится значение тренера по плану;
2. Если Полю выполнено ставится значение ложь, то значение поля кто провел очищается.

Пример экспорта таблицы Журнал занятий в MS Excel приведен на рисунке 39.

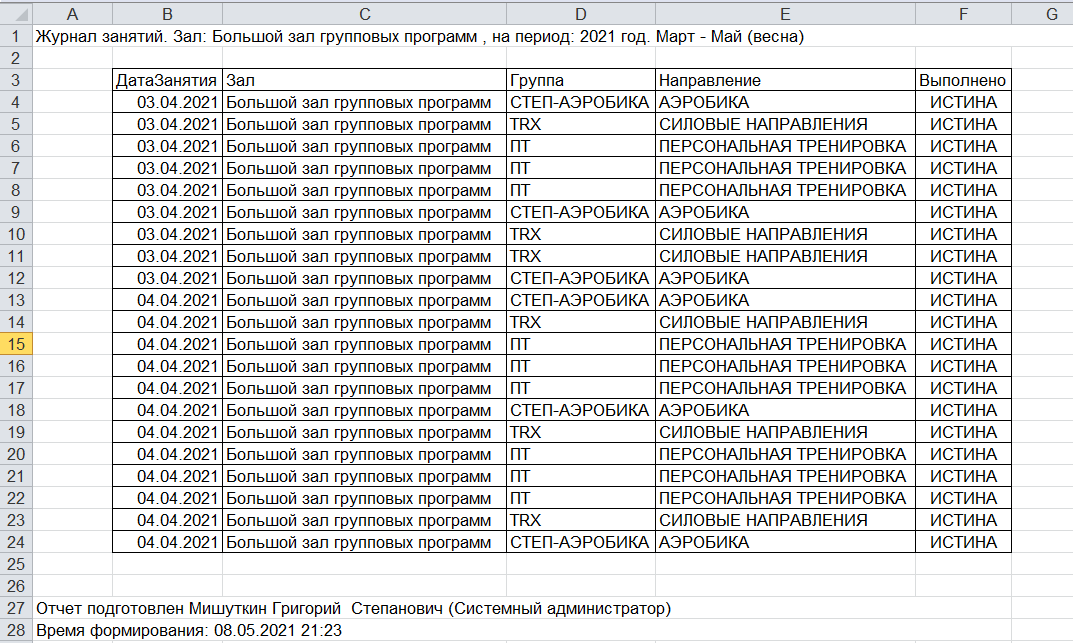


Рисунок – Пример экспорта таблицы Журнал занятий в MS Excel

## 4.6 Главное окно. Вкладка Сотрудники

Интерфейс вкладки Сотрудники представлен на рисунке 40.

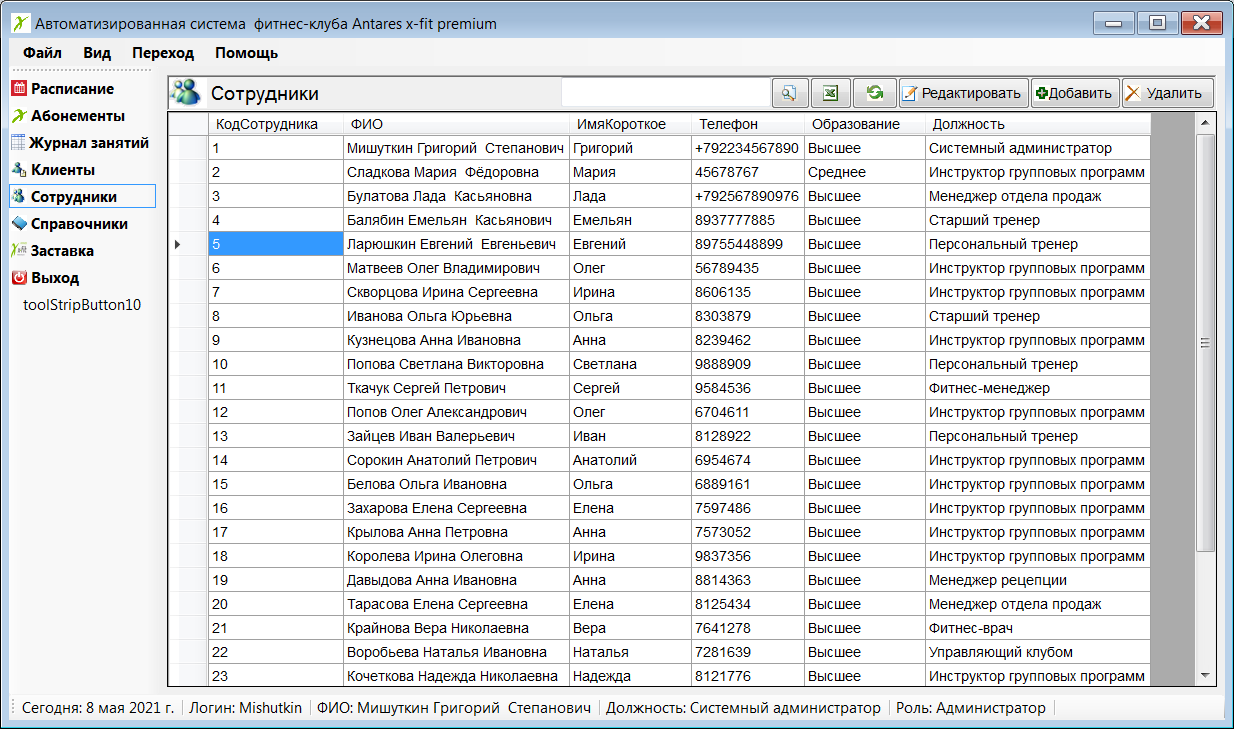


Рисунок – Главное окно. Вкладка Сотрудники

Доступные функции на вкладке Сотрудники:

1. Поиск по всем полям таблицы Сотрудники (поля интерпретируются как текстовые)
2. Удаление Сотрудника (удалять можно только тех сотрудников, которые не задействованы в таблице Расписание в качестве Тренера, В таблице Журнал занятий, или в таблице Абонементы)
3. Переход к редактированию или просмотру Сотрудника в отдельном окне
4. Переход к добавлению Сотрудника в отдельном окне
5. Обновление таблицы Сотрудники
6. Экспорт данных в Excel

При удалении Сотрудника, в случае если Сотрудник задействован в таблицах Расписание, Журнал занятий или Абонементы, выводится сообщение об отказе, в противном случае выводится предупреждающее сообщение и запрашивается подтверждение пользователя на удаление Сотрудника.

## 4.7 Главное окно. Вкладка Клиенты

Интерфейс вкладки Клиенты представлен на рисунке 41.

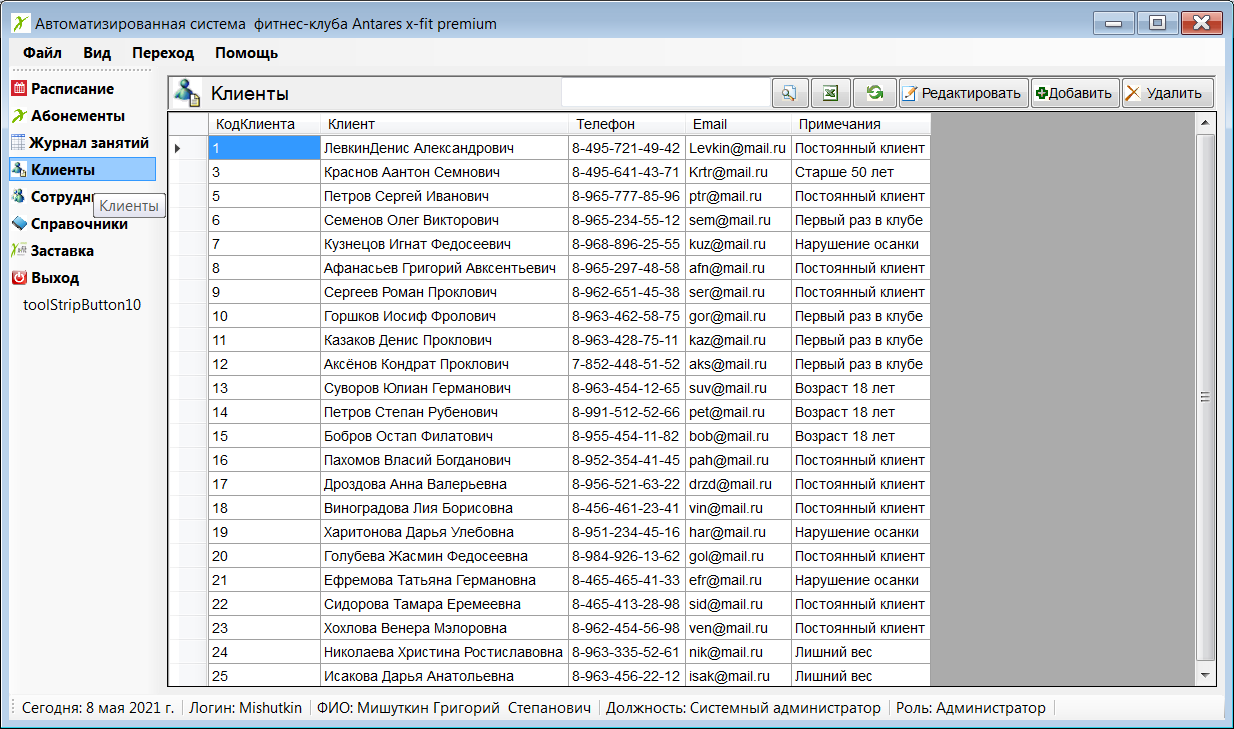


Рисунок – Главное окно. Вкладка Клиенты

Доступные функции на вкладке Сотрудники:

1. Поиск по всем полям таблицы Клиенты (поля интерпретируются как текстовые)
2. Удаление Клиента (удалять можно только тех клиентов, которые не задействованы в таблице Абонементы)
3. Переход к редактированию или просмотру Клиента в отдельном окне
4. Переход к добавлению Клиента в отдельном окне
5. Обновление таблицы Клиенты
6. Экспорт данных в Excel

При удалении Клиента, в случае если Клиент задействован в таблице Абонементы, выводится сообщение об отказе, в противном случае выводится предупреждающее сообщение и запрашивается подтверждение пользователя на удаление Клиента.

## 4.8 Главное окно. Вкладка Справочники

Интерфейс вкладки Справочники представлен на рисунке 42.

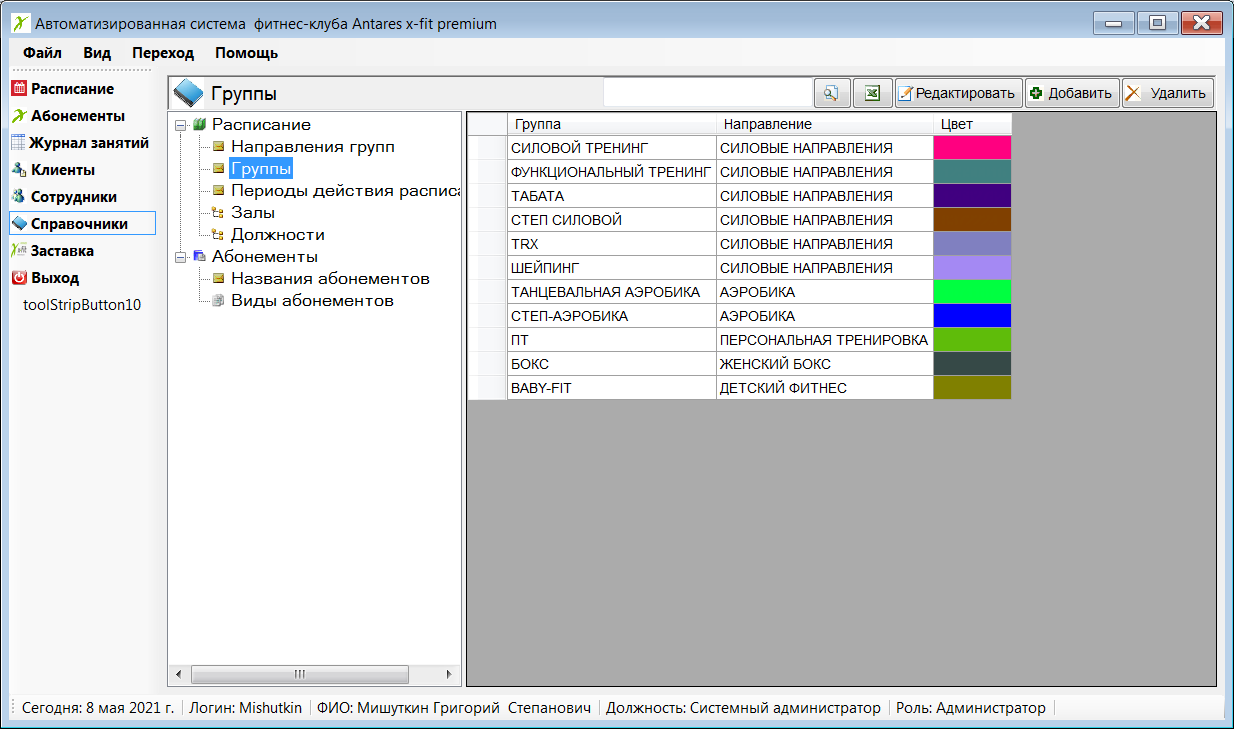


Рисунок – Главное окно. Вкладка Справочники

Доступные функции на вкладке Справочники:

1. Операции поиска, удаления, редактирования, удаления в универсальной форме по справочникам:
   1. Направления групп
   2. Группы
   3. Период действия расписания
   4. Залы
   5. Должности
   6. Названия абонементов
   7. Виды абонементов
2. Экспорт данных в Excel

Для ограничения целостности при добавлении данных выполняется проверка на наличие таких данных в таблице (рис.43).

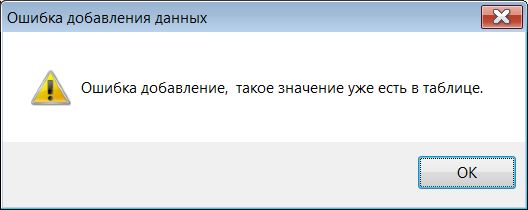


Рисунок – Сообщение об отказе

Ограничения целостности

При удалении проверяется:

1. Из таблицы Должности наличие связанных записей в таблице Сотрудники
2. Из таблицы Направление групп наличие связанных записей в таблице Группы
3. Из таблицы Группы наличие связанных записей в таблице Расписание
4. Из таблицы Период действия расписания наличие связанных записей в таблице Расписание
5. Из таблицы Залы деятельности наличие связанных записей в таблицах Виды абонементов и Расписание
6. Из таблицы Названия абонементов заказа наличие связанных записей в таблице Виды абонементов
7. Из таблицы Виды абонементов наличие связанных записей в таблице Абонементы

Это выполняется в методе класса ClassBaseRC.cs: public void deleterecord(string nametable, DataGridView grid). Сообщение об ошибке удаления приведено на рисунке 44.

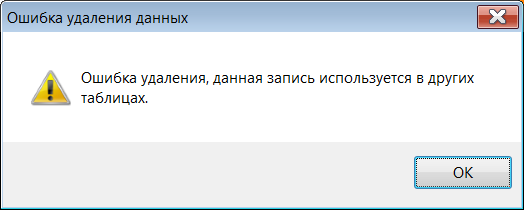


Рисунок – Сообщение об ошибке удаления

При удалении, в случае если запись задействована в других таблицах, выводится сообщение об отказе, в противном случае выводится предупреждающее сообщение и запрашивается подтверждение пользователя на удаление записи (рис. 45).

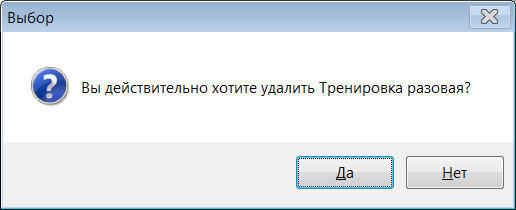


Рисунок – Запрос на подтверждение удаления

## 4.9 Вспомогательные формы. Форма Клиент

Всего в программе разработано 14 форм кроме главной формы. Формы аутентификации, абонемента и расписания рассматривались выше.

Форма Клиент предназначена для работы с данными заказчика. Интерфейс формы которой приведен на рисунке 46.



Рисунок – Интерфейс формы Клиент

Доступные функции на форме Клиент:

1. Ввод названия (имени) клиента
2. Ввод телефона клиента
3. Ввод Email клиента
4. Вод примечаний

## 4.10 Вспомогательные формы. Форма Группа

Форма Поставщик предназначена для редактирования групп. Интерфейс формы, которой приведен на рисунке 47.

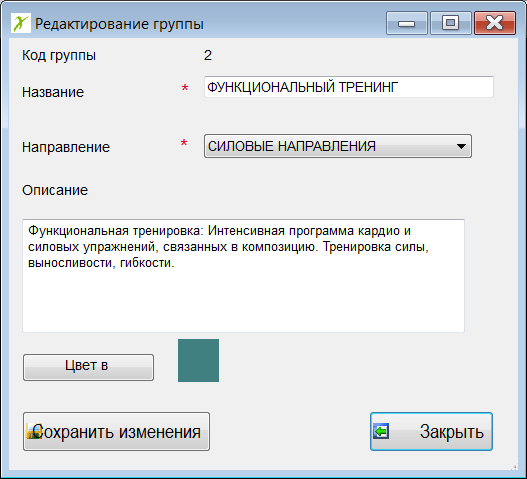


Рис. Интерфейс формы Группа

Доступные функции на форме Группа:

1. Ввод названия группы
2. Выбор направления
3. Выбор цвета отображения группы в расписании
4. Вод описания

## 4.11 Вспомогательные формы. Форма Зал

Форма Зал предназначена для редактирования данных по залам. Интерфейс формы, которой приведен на рисунке 47.

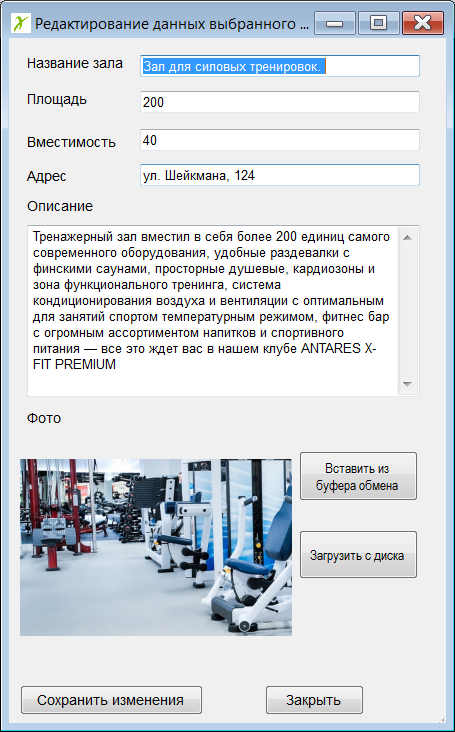


Рисунок – Интерфейс формы Зал

Доступные функции на форме Зал:

1. Ввод названия зала
2. Ввод площади, вместимости, адреса, описания
3. Выбор или вставка фотографии
4. Вод описания

## 4.12 Вспомогательные формы. Форма Вид абонемента

Форма Вид абонемента предназначена для редактирования данных по виду абонемента. Интерфейс формы, которой приведен на рисунке 48.

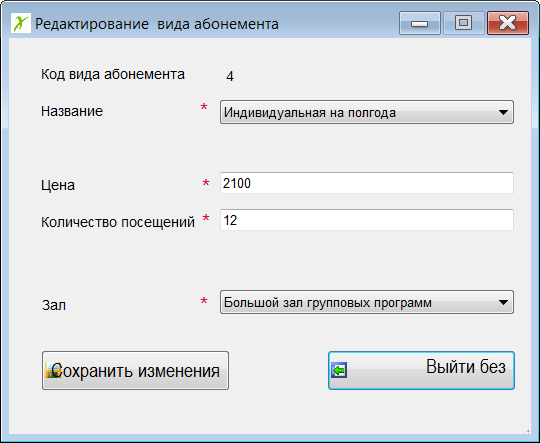


Рисунок – Интерфейс формы Зал

Доступные функции на форме Вид абонемента:

1. Выбор названия абонемента (формируется в отдельном справочнике), зала
2. Ввод цены, количества посещений

## 4.13 Вспомогательные формы. Период действия расписания

Форма Период действия расписания предназначена для работы с периодами. Интерфейс формы приведен на рисунке 49.

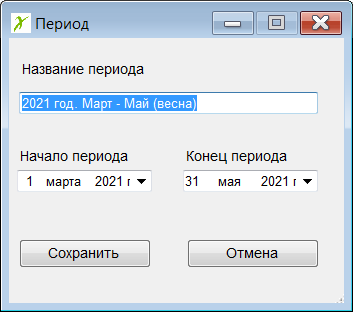


Рисунок – Интерфейс формы Период действия расписания

1. Ввод названия периода
2. Ввод начала и конца периода

При добавлении и редактировании проводится проверка, чтобы не было пересекающихся периодов и дата конца не была меньше даты начала.

## 4.14 Вспомогательные формы. Формы Сотрудник, форма смены пароля сотрудника

Форма Сотрудник предназначена для работы с персональными данными сотрудников. Интерфейс формы приведен на рисунке 50.

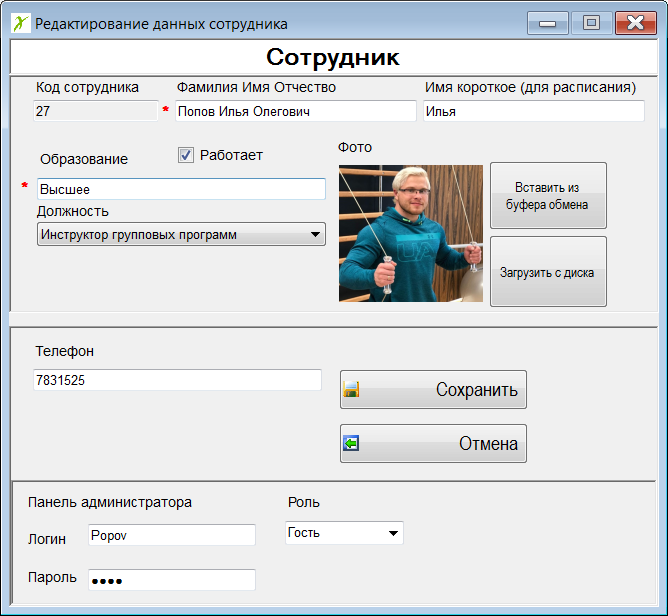


Рисунок – Интерфейс формы Сотрудники

Доступные функции на форме Сотрудники

1. Ввод ФИО
2. Ввод короткого имени (для расписания)
3. Ввод Email
4. Установка отметки работает / не работает
5. Выбор должности из раскрывающегося списка
6. Выбор Роли из раскрывающегося списка
7. Ввод телефона
8. Ввод логина
9. Ввод пароля

Примечание: панель администратора доступна только сотруднику с ролью Администратор.

## 4.15 Вспомогательные формы. Формы смены пароля

Форма смены пароля позволяет сменить ткущий пароль сотрудника (рис. 51).

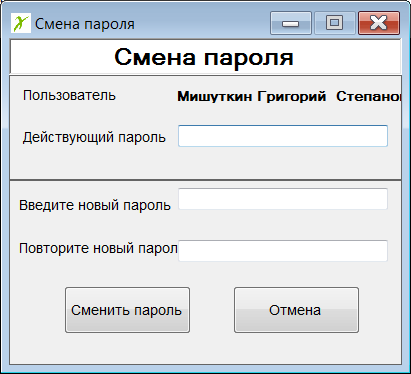


Рисунок – Форма смены пароля

Ошибки пользователя обрабатываются и выводятся соответствующие сообщения (рис. 52).

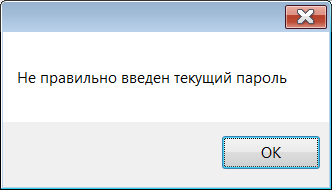
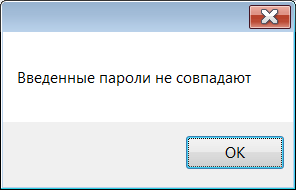
 

Рисунок – Сообщения об ошибках

## 4.16 Вспомогательные формы. Формы редактирования атрибутов в справочниках

В тех справочниках, где необходимо отредактировать только один атрибут, например Название должности, или название вида абонементов используется универсальная форма (formspr). В качестве входного параметра форме передается из основной формы (form1) название таблицы, id записи, название ключевого поля, название редактируемого поля, текущее значение редактируемого поля.

Интерфейс формы редактирования полей таблиц приведен на рисунке 53.

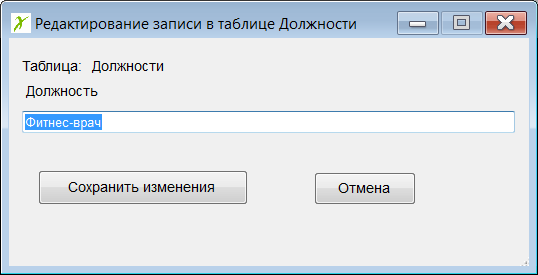


Рисунок – Интерфейс формы редактирования полей таблиц

При сохранении записи осуществляется проверка на отсутствие в таблице записи с такими же значениями, в случае ошибки выдается сообщение (рис. 54).



Рисунок – Сообщение об ошибке

## 4.17 Экспорт данных из таблиц

Все таблицы можно экспортировать в MS Excel, для этого необходимо нажать кнопку  (рис. 55).



Рис. Экспорт данных в Excel

## 4.18 Справочная система

В программе разработан файл справки SpravkaFit.pdf. Фаул открывается по пункт меню Помощь – Справка (рис. 55).

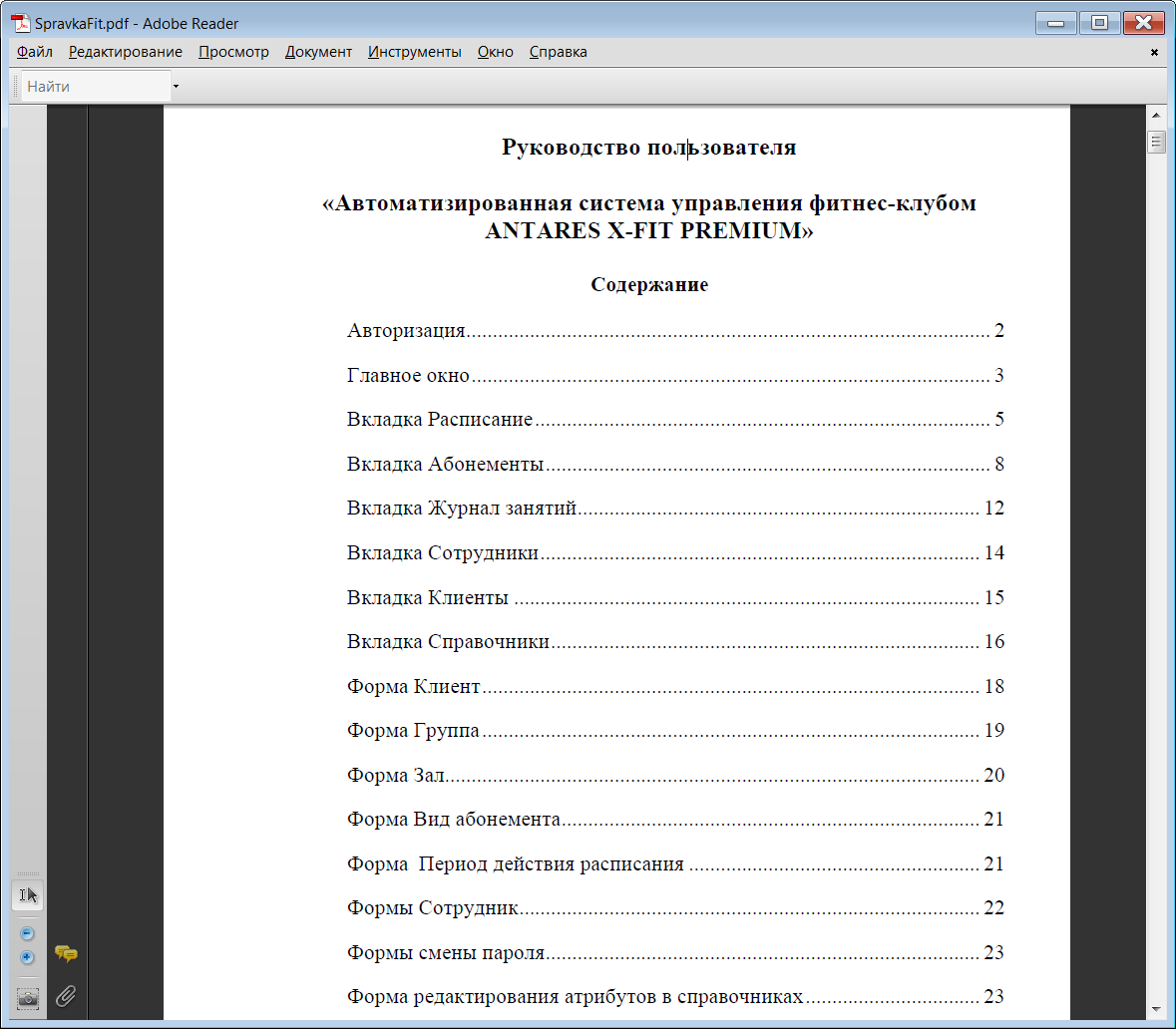


Рисунок – файл справки SpravkaFit.pdf

Выводы

В ходе проектирование было разработано сложное приложение, включающее 15 форм для фитнес-клуба «ANTARES X-FIT PREMIUM» включающее в себя исполняемый файл FitnessClub.exe и базу данных BaseFit.mdb. В качестве руководства пользователя выступает файл SpravkaFit.pdf. Открытие файла осуществляется по пункту меню Помощь – Справка. Полностью содержание файла приведено в приложении Б.

Приложение выполняет все функции которые были определены при постановке задачи.

# Заключение

Объектом исследования данной выпускной квалификационной работы явилась проектирование и разработка автоматизированной системы управления фитнес-клубом .

В процессе выполнения выпускной квалификационной работы был проведен анализ организационной структуры фитнес-клуба «ANTARES X-FIT PREMIUM» , анализ системы для хранения и обработки данных по учету абонементов, были определены функции, выполняемые в работниками фитнес-клуба, была изучена технология сбора, обработки и передачи информации.

Создание системы проходило в несколько этапов. На этапе анализа предметной области была изучена деятельность организации и основные процессы, связанные с оформлением и учетом абонементов дилера. На этапе проектирования было определено, что основу разрабатываемой системы должна составлять база данных. При создании БД были разработаны инфологическая, логическая и физическая модели базы данных. На этапе технического проектирования были выяснены технические аспекты работы фитнес-клуба «ANTARES X-FIT PREMIUM», а так же способы возможной модернизации. На этапе непосредственного создания системы был разработан конечный работоспособный программный продукт.

Итогом выпускной квалификационной работы является спроектированная автоматизированная система управления фитнес-клубом "ANTARES X-FIT PREMIUM".

В процессе разработки приложения были поэтапно реализованы следующие задачи:

1. Проведен анализ предметной области, изучена организационная структура и объекты управления

2. Обоснована необходимость использования автоматизированной системы управления фитнес-клубом.

3. Проведен анализ существующих разработок и обоснование выбора технологии проектирования

5. Определены бизнес-процессы и построена информационная модель. Разработана функциональная структура информационной системы учета выдачи абонементов и оплаты по ним фитнес-клуба «ANTARES X-FIT PREMIUM».

6. Разработано формализованное описание объекта автоматизированной информационной системы

7. Осуществлена разработка инфологической модели

8. Осуществлена разработка логической модели БД

9. Осуществлена разработка физической модели БД

10. Определены системные требования к разрабатываемой программе и выбрана операционная система.

11. В приложении MS Access разработано информационное хранилище данных (база данных): файл BaseFIT.mdb

12. В приложении Visual Studio 2010 на языке программирования C# разработан интерфейс пользователя для доступа к базе данных и оперативного получения всей необходимой информации: файл FitnessClub.exe.

В приложении разработано 15 различных форм, что позволяет сотрудникам обращаться работать со всеми объектами автоматизированной системы управления фитнес-клубом. Данная программа удовлетворяет всем требованиям, которые к ней предъявлялись. Для запуска и стабильной работы не требуются большие системные ресурсы, что обусловлено хорошей оптимизацией кода.

Программа написана на простом уровне, доступном программистам с начальным уровнем, и имеет большие возможности для творчества программистов в плане модернизации.

Можно выделить основные пути модернизации:

1. Разработка отчетов, позволяющих получать более уточненную специализированную информацию по абонементам и проведению занятий;
2. Разработать журнал учета посещений занятий клиентами.

В результате работы над ВКР была проведена самостоятельная исследовательская работа, а также систематизированы, закреплены и углублены теоретические и практические знания, отработаны практические навыки, сформированные на протяжении учебы по специальности, которые были применены для решения конкретной задачи.

# Литература

1. Интернет ресурс https://antares-xfit.ru/
2. Интернет ресурс <https://antares-xfit.ru/about/club-cards/>
3. Интернет ресурс https://fprosoftware.com/
4. Интернет ресурс http://maxtarget.by/crm-sistemy/konfiguratsiya-fitnes-klub
5. Дейт, К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2016. – 1328 с.
6. Кузнецов С.Д. Базы данных: учебник для студ. Учреждений высшего проф. Образования / С.Д. Кузнецов. – М. : Издательский центр «Академия», 2012. – 496 с.
7. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений / Под ред. Проф. А.Д. Хоменко. – 4-е изд., доп.и перераб. – СПб.: КОРОНА принт, 2014, 736 с.
8. Павловская Т. А. C#. Программирование на языке высокого уровня : [учебник для вузов по направлению "Информатика и вычислительная техника"] - М. [и др.] : Питер , 2007, 2009, 432 с. ил.
9. Нортроп, Тони. Основы разработки приложений на платформе Microsoft .NET Framework : экзамен-536 MCTS; [пер. с англ.] - М. : Русская редакция ; СПб. : Питер , 2007 , 842 с. ил.
10. Марченко А. Л. Основы программирования на С# 2.0 : учебное пособие- М.: Интернет-Университет Информационных Технологий : Бином , 2007, 551 с. ил.
11. Васюткина И.А. Технология разработки объектно-ориентированных программ на С# в Visual Studio.Net. – Новосибирск: НГТУ, 2010
12. Экономическая эффективность информационных систем в России Авторы: К. Скрипкин

# Приложение А. Листинг модулей

Form1.cs – главная форма

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

using System.IO;

using Word = Microsoft.Office.Interop.Word;

namespace FitnessClub

{

public partial class Form1 : Form

{

object CellValue;

ClassBaseFIT BaseFIT = new ClassBaseFIT();

// переменные для передачи в универсальную форму выбора

public static bool modified { get; set; }

public static string nameperiod { get; set; }

public static int idperiod{ get; set; }

public static string namegruppa{ get; set; }

public static int idgruppa{ get; set; }

public static string nameperiod2 { get; set; }

public static int idperiod2 { get; set; }

DataSet dsjournal = new DataSet();

DataView dvjournal = new DataView();

public static Boolean insert { get; set; }

int rowcount;

public Form1()

{

InitializeComponent();

idperiod = -1;

}

private void Form1\_Load(object sender, EventArgs e)

{

//запускаем форму аутентификации

LoginForm w = new LoginForm();

if (!(w.ShowDialog() == DialogResult.OK)) //форма регистрации будет открыта раньше, чем главная форма

{

Application.Exit(); //если форма регистрации закроется с неправильным диалогрезалтом, то программу закрываем

this.Close();

return;

}

if (!Program.guest)

{

LoginLabel.Text = "Логин: " + Program.BaseFIT.loginuser;

FioLabel.Text = "ФИО: " + Program.BaseFIT.fiouser;

doljnlabel.Text = "Должность: " + Program.BaseFIT.doljnuser;

lroluser.Text = "Роль: " + Program.BaseFIT.roleuser;

}

else

{

LoginLabel.Text = "Гость" ;

FioLabel.Text = "" ;

doljnlabel.Text = "";

lroluser.Text = "";

}

//настраиваем вид скрыаем ярлыки листов

tabControl1.DrawMode = TabDrawMode.OwnerDrawFixed;

tabControl1.Appearance = TabAppearance.Buttons;

tabControl1.ItemSize = new System.Drawing.Size(0, 1);

tabControl1.SizeMode = TabSizeMode.Fixed;

tabControl1.TabStop = false;

DataLabel.Text = "Сегодня: " + DateTime.Now.ToLongDateString();

gridspr.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

gridabonement.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

gridklient.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

gridsotrudnik.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

Program.selectspr ="parrent";

for (int i = 0; i < tree.Nodes.Count; i++) { if (i < 2) tree.Nodes[i].Expand(); }

// загружаем на странице Клиенты Сотрудники абонементы

BaseFIT.selecttable("Клиенты", gridklient);

BaseFIT.selecttable("Сотрудники", gridsotrudnik);

BaseFIT.selecttable("Абонементы", gridabonement);

//заполняем период

string sql = @"SELECT ПериодДействия.КодПериода, ПериодДействия.НазваниеПериода, ПериодДействия.НачалоПериода, ПериодДействия.КонецПериода

FROM ПериодДействия

WHERE (([НачалоПериода]<=Date() And [КонецПериода]>=Date()));

";

DataSet ds = new DataSet();

BaseFIT.gettable(sql,ds);

if (ds.Tables[0].Rows.Count > 0)

{

nameperiod = ds.Tables[0].Rows[0]["НазваниеПериода"].ToString();

tbnameperiod.Text = nameperiod;

idperiod = Convert.ToInt32( ds.Tables[0].Rows[0]["КодПериода"].ToString());

nameperiod2 = ds.Tables[0].Rows[0]["НазваниеПериода"].ToString();

tbnameperiod2.Text = nameperiod2;

idperiod2 = Convert.ToInt32( ds.Tables[0].Rows[0]["КодПериода"].ToString());

}

//загружаем combobox

sql = @"SELECT кодзала, зал FROM Залы order by Зал";

ds = new DataSet();

OleDbDataAdapter da = new OleDbDataAdapter(sql, BaseFIT.connection);

BaseFIT.con.Open();

da.Fill(ds);

BaseFIT.con.Close();

cbzall.DataSource = null;

cbzall.DataSource = ds.Tables[0];

cbzall.DisplayMember = "Зал";

cbzall.ValueMember = "КодЗала";

//2-й combobox

DataSet ds2 = new DataSet();

OleDbDataAdapter da2 = new OleDbDataAdapter(sql, BaseFIT.connection);

BaseFIT.con.Open();

da2.Fill(ds2);

BaseFIT.con.Close();

cbzaljournal.DataSource = null;

cbzaljournal.DataSource = ds2.Tables[0];

cbzaljournal.DisplayMember = "Зал";

cbzaljournal.ValueMember = "КодЗала";

//переходим на заставку

tabControl1.SelectedIndex = 1;

zagruzkachb();

idgruppa=0;

prava();

}

//права для гостя

private void prava()

{

if (Program.guest == true)

{

toolStrip2.Visible = false;

button6.Visible = false;

видToolStripMenuItem.Visible = false;

переходToolStripMenuItem.Visible = false;

сменаПароляToolStripMenuItem.Visible = false;

}

}

//выделение пункт в дереве справочников

private void tree\_DrawNode(object sender, DrawTreeNodeEventArgs e)

{

if (e.Node == null) return;

// if treeview HideSelection property is "True",

// this will always returns "False" on unfocused treeview

var selected = (e.State & TreeNodeStates.Selected) == TreeNodeStates.Selected;

var unfocused = !e.Node.TreeView.Focused;

// we need to do owner drawing only on a selected node

// and when the treeview is unfocused, else let the OS do it for us

if (selected && unfocused)

{

var font = e.Node.NodeFont ?? e.Node.TreeView.Font;

e.Graphics.FillRectangle(SystemBrushes.Highlight, e.Bounds);

TextRenderer.DrawText(e.Graphics, e.Node.Text, font, e.Bounds, SystemColors.HighlightText, TextFormatFlags.GlyphOverhangPadding);

}

else

{

e.DrawDefault = true;

}

}

//настраиваем права пользователей

private Boolean knopkispr\_enabled()

{

if (Program.selectspr == "parrent")

{

btneditspr.Enabled = false;

btninsertspr.Enabled = false;

btndeletespr.Enabled = false;

return false;

}

if ((Program.BaseFIT.roleuser == "Администратор") || (Program.BaseFIT.roleuser == "Менеджер") || (Program.BaseFIT.roleuser == "Системный администратор"))

{

{

btneditspr.Enabled = true;

btninsertspr.Enabled = true;

btndeletespr.Enabled = true;

}

}

else

{

btneditspr.Enabled = false;

btninsertspr.Enabled = false;

btndeletespr.Enabled = false;

}

return true;

}

//обработка событий в дереве справочников

private void tree\_AfterSelect(object sender, TreeViewEventArgs e)

{

Program.selectspr = tree.SelectedNode.Tag.ToString();

LabelSpavochnik.Text = tree.SelectedNode.Text;

if (!knopkispr\_enabled()) return;

BaseFIT.selecttable(Program.selectspr, gridspr);

}

//проверка выбрана ли строка в таблице

public int indexselectedgrid(DataGridView grid)

{

if (grid.Rows.Count == 0)

{

return -1;

}

if (grid.CurrentRow != null)

return grid.CurrentRow.Index;

else return -1;

}

//кнопка редактировать на вкладке справочники

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

{

BaseFIT.edittable(Program.selectspr, gridspr, true);

}

//кнопка добавить на вкладке справочники

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

{

BaseFIT.insertrecord(Program.selectspr, gridspr, true);

}

//кнопка удалить на вкладке справочники

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

{

BaseFIT.deleterecord(Program.selectspr, gridspr);

}

//двойной щелчок мыши на вкладке справочники

private void gridspr\_MouseDoubleClick(object sender, MouseEventArgs e)

{

btneditspr\_Click(sender, e);

}

//кнопка обновить на вкладке абонементы

private void button5\_Click(object sender, EventArgs e)

{

BaseFIT.selecttable("Абонементы", gridabonement);

}

//кнопка удалить на вкладке абонементы

private void button3\_Click(object sender, EventArgs e)

{

BaseFIT.deleterecord("Абонементы", gridabonement);

}

//кнопка редактировать на вкладке абонементы

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

{

BaseFIT.edittable("Абонементы", gridabonement,true);

}

//кнопка добавить на вкладке абонементы

private void button2\_Click(object sender, EventArgs e)

{

BaseFIT.insertrecord("Абонементы", gridabonement, true);

}

//мненю выход

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

{

Close();

}

//вызова о программе

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

{

fabout w = new fabout();

w.ShowDialog();

}

//кнопка экспорт в excel на вкладке абонементы

private void button24\_Click(object sender, EventArgs e)

{

ToExcel(gridabonement, lbabonement.Text, 1, 0);

}

//таблицу отчета в excel

public void ToExcel(System.Windows.Forms.DataGridView dgrid, string s, int start,int end)

{

if (dgrid.RowCount == 0)

{

MessageBox.Show("Нет данных для экспорта");

return;

}

Microsoft.Office.Interop.Excel.Application ExcelApp = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook ExcelWorkBook;

Microsoft.Office.Interop.Excel.Worksheet ExcelWorkSheet;

//Книга.

ExcelWorkBook = ExcelApp.Workbooks.Add(System.Reflection.Missing.Value);

//Таблица.

ExcelWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)ExcelWorkBook.Worksheets.get\_Item(1);

// Выделяем диапазон ячеек от H1 до K1

Microsoft.Office.Interop.Excel.Range \_excelCells1 = (Microsoft.Office.Interop.Excel.Range)ExcelWorkSheet.get\_Range("A1", "H1").Cells;

// Производим объединение

\_excelCells1.Merge(Type.Missing);

ExcelWorkSheet.Cells[1, 1] = s;

for (int j = start; j < dgrid.ColumnCount; j++)

if (end != 0)

{

if (j <= end) ExcelApp.Cells[3, j + 1] = dgrid.Columns[j].HeaderText;

}

else ExcelApp.Cells[3, j + 1] = dgrid.Columns[j].HeaderText;

int i;

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

{

for (int j = start; j < dgrid.ColumnCount; j++)

{

if (end != 0)

{

if (j <= end) ExcelApp.Cells[i + 4, j + 1] = dgrid.Rows[i].Cells[j].Value;

}

else ExcelApp.Cells[i + 4, j + 1] = dgrid.Rows[i].Cells[j].Value;

}

}

// Выделяем диапазон ячеек от H1 до K1

\_excelCells1 = (Microsoft.Office.Interop.Excel.Range)ExcelWorkSheet.get\_Range("A" + (i + 6).ToString(), "F" + (i + 6).ToString()).Cells;

// Производим объединение

\_excelCells1.Merge(Type.Missing);

ExcelWorkSheet.Cells[i + 6, 1] = "Отчет подготовлен " + Program.BaseFIT.fiouser + " (" + Program.BaseFIT.doljnuser + ")";

// ExcelApp.Cells[i + 6,1] =

string s1 = DateTime.Now.ToString("dd/MM/yyyy HH:mm");

\_excelCells1 = (Microsoft.Office.Interop.Excel.Range)ExcelWorkSheet.get\_Range("A" + (i + 7).ToString(), "F" + (i + 7).ToString()).Cells;

// Производим объединение

\_excelCells1.Merge(Type.Missing);

ExcelApp.Cells[i + 7, 1] = "Время формирования: " + s1;

//Вызываем нашу созданную эксельку.

ExcelApp.Columns.EntireColumn.AutoFit();

ExcelApp.Visible = true;

ExcelApp.UserControl = true;

}

//таблицу расписания в excel

public void ToExcelRasp(string s, int start, int end)

{

int i;

Boolean flag=false;

int counti=0;

for ( i=0;i<50;i++)

for (int j = 0; j <= 7; j++)

{

if (BaseFIT.rooster[i, j].MyLabel != null)

{

flag = true;

counti = i;

}

}

if (!flag)

{

MessageBox.Show("Не сформировано расписание");

return;

}

Microsoft.Office.Interop.Excel.Application ExcelApp = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook ExcelWorkBook;

Microsoft.Office.Interop.Excel.Worksheet ExcelWorkSheet;

//Книга.

ExcelWorkBook = ExcelApp.Workbooks.Add(System.Reflection.Missing.Value);

//Таблица.

ExcelWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)ExcelWorkBook.Worksheets.get\_Item(1);

// Выделяем диапазон ячеек от H1 до K1

Microsoft.Office.Interop.Excel.Range \_excelCells1 = (Microsoft.Office.Interop.Excel.Range)ExcelWorkSheet.get\_Range("A1", "H1").Cells;

// Производим объединение

\_excelCells1.Merge(Type.Missing);

ExcelWorkSheet.Cells[1, 1] = s;

// дни недели

for (int j = 0; j <= 7; j++)

{

ExcelApp.Cells[3, j + 1] = BaseFIT.zagolvki[j].Text;

ExcelApp.Cells[3, j + 1].Interior.Color = BaseFIT.zagolvki[j].BackColor;

}

for (i = 0; i < counti; i++)

{

for (int j = 0; j <= 7; j++)

{

ExcelApp.Cells[i + 4, j + 1] = BaseFIT.rooster[i, j].MyLabel.Text;

ExcelApp.Cells[i + 4, j + 1].Interior.Color = BaseFIT.rooster[i, j].MyLabel.BackColor;

if (j != 0)

{

Microsoft.Office.Interop.Excel.Range excelRange = ExcelWorkSheet.Cells[i + 4, j + 1] as Microsoft.Office.Interop.Excel.Range;

excelRange.Font.Color = Color.White;

}

}

}

// Выделяем диапазон ячеек от H1 до K1

\_excelCells1 = (Microsoft.Office.Interop.Excel.Range)ExcelWorkSheet.get\_Range("A" + (i + 6).ToString(), "F" + (i + 6).ToString()).Cells;

// Производим объединение

\_excelCells1.Merge(Type.Missing);

ExcelWorkSheet.Cells[i + 6, 1] = "Расписание подготовлено " + Program.BaseFIT.fiouser + " (" + Program.BaseFIT.doljnuser + ")";

// ExcelApp.Cells[i + 6,1] =

string s1 = DateTime.Now.ToString("dd/MM/yyyy HH:mm");

\_excelCells1 = (Microsoft.Office.Interop.Excel.Range)ExcelWorkSheet.get\_Range("A" + (i + 7).ToString(), "F" + (i + 7).ToString()).Cells;

// Производим объединение

\_excelCells1.Merge(Type.Missing);

ExcelApp.Cells[i + 7, 1] = "Время формирования: " + s1;

//Вызываем нашу созданную эксельку.

ExcelApp.Columns.EntireColumn.AutoFit();

ExcelApp.Visible = true;

ExcelApp.UserControl = true;

}

//кнопка в excel

private void button32\_Click(object sender, EventArgs e)

{

ToExcel(gridjournal, "Журнал занятий", 1, 14);

}

// в excel справочник

private void button62\_Click(object sender, EventArgs e)

{

ToExcel(gridspr,LabelSpavochnik.Text, 1, 0);

}

//меню вызов справки

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

{

string commandText = Directory.GetCurrentDirectory() + "\\SpravkaFitnessClub.pdf";

if (File.Exists(commandText))

{

try

{

var proc = new System.Diagnostics.Process();

proc.StartInfo.FileName = commandText;

proc.StartInfo.UseShellExecute = true;

proc.Start();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка открытия файла справки");

}

}

else MessageBox.Show("Файл справки отсутствует");

}

//вызов формы смены пароля

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

{

FormChangeParol w = new FormChangeParol();

w.ShowDialog();

}

bool IsTheSameCellValue(int column, int row,DataGridView grid )

{

if (column != 1) return false;

DataGridViewCell cell1 = grid[column, row];

DataGridViewCell cell2 = grid[column, row - 1];

if (cell1.Value == null || cell2.Value == null)

{

return false;

}

return cell1.Value.ToString() == cell2.Value.ToString();

}

//выделяем кнопку на панели кнопок слева

private void checked\_knopka(object sender)

{

foreach (ToolStripButton item in ((ToolStripButton)sender).GetCurrentParent().Items)

{

if (item == sender)

{

item.Checked = true;

}

if ((item != null) && (item != sender))

{

item.Checked = false;

}

}

}

//расписание

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

{

checked\_knopka(sender);

tabControl1.SelectedIndex = 0;

}

// Клиенты

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

{

checked\_knopka(sender);

tabControl1.SelectedIndex = 4;

}

// Клиенты

private void toolStripButton3\_Click(object sender, EventArgs e)

{

checked\_knopka(sender);

tabControl1.SelectedIndex = 4;

}

// выход

private void toolStripButton6\_Click(object sender, EventArgs e)

{

Close();

}

//заставка

private void toolStripButton4\_Click(object sender, EventArgs e)

{

checked\_knopka(sender);

tabControl1.SelectedIndex = 1;

}

//справочники

private void toolStripButton5\_Click(object sender, EventArgs e)

{

checked\_knopka(sender);

tabControl1.SelectedIndex = 3;

}

//журнал занятий

private void toolStripButton7\_Click(object sender, EventArgs e)

{

checked\_knopka(sender);

tabControl1.SelectedIndex = 2;

}

//обновить клиенты

private void button35\_Click(object sender, EventArgs e)

{

BaseFIT.selecttable("Клиенты", gridklient);

}

private void button36\_Click(object sender, EventArgs e)

{

BaseFIT.edittable("Клиенты", gridklient, true);

}

//вставка клиенты

private void button37\_Click(object sender, EventArgs e)

{

BaseFIT.insertrecord("Клиенты", gridklient, true);

}

//удалить лиенты

private void button38\_Click(object sender, EventArgs e)

{

BaseFIT.deleterecord("Клиенты", gridklient);

}

// в excel rkbtyns

private void button34\_Click(object sender, EventArgs e)

{

ToExcel(gridklient, "Клиенты", 1, 0);

}

//обновить сотрудники

private void button40\_Click(object sender, EventArgs e)

{

BaseFIT.selecttable("Сотрудники", gridsotrudnik);

}

//меню сотрудники

private void toolStripButton8\_Click(object sender, EventArgs e)

{

checked\_knopka(sender);

tabControl1.SelectedIndex = 5;

}

//в excel сотрудники

private void button39\_Click(object sender, EventArgs e)

{

ToExcel(gridsotrudnik, "Сотрудники", 1, 0);

}

//удалить сотрудника

private void button43\_Click(object sender, EventArgs e)

{

BaseFIT.deleterecord("Сотрудники", gridsotrudnik);

}

//добавить сотрудника

private void button42\_Click(object sender, EventArgs e)

{

BaseFIT.insertrecord("Сотрудники", gridsotrudnik, true);

}

//редактировать сотрудника

private void button41\_Click(object sender, EventArgs e)

{

BaseFIT.edittable("Сотрудники", gridsotrudnik, true);

}

//двойной щелчок на таблице сотрудники

private void gridsotrudnik\_MouseDoubleClick(object sender, MouseEventArgs e)

{

button41\_Click(sender,e);

}

//загрузка combobox

private void zagruzkachb()

{

//загружаем combobox

string sql = @"SELECT кодсотрудника, ФИО, имякороткое FROM Сотрудники order by ФИО";

DataSet ds = new DataSet();

OleDbDataAdapter da = new OleDbDataAdapter(sql, BaseFIT.connection);

BaseFIT.con.Open();

da.Fill(ds);

BaseFIT.con.Close();

cbsotrvipolnil.DataSource = null;

cbsotrvipolnil.DataSource = ds.Tables[0];

cbsotrvipolnil.DisplayMember = "ФИО";

cbsotrvipolnil.ValueMember = "кодсотрудника";

DataSet ds2 = new DataSet();

OleDbDataAdapter da2 = new OleDbDataAdapter(sql, BaseFIT.connection);

BaseFIT.con.Open();

da2.Fill(ds2);

BaseFIT.con.Close();

cbsotrplan.DataSource = null;

cbsotrplan.DataSource = ds2.Tables[0];

cbsotrplan.DisplayMember = "ФИО";

cbsotrplan.ValueMember = "кодсотрудника";

sql = @"SELECT КодНаправления, Направление

FROM Направления order by Направление";

DataSet ds3 = new DataSet();

OleDbDataAdapter da3 = new OleDbDataAdapter(sql, BaseFIT.connection);

BaseFIT.con.Open();

da3.Fill(ds3);

BaseFIT.con.Close();

cbnapr.DataSource = null;

cbnapr.DataSource = ds3.Tables[0];

cbnapr.DisplayMember = "Направление";

cbnapr.ValueMember = "КодНаправления";

cbdennedeli.SelectedIndex = 0;

}

//обновить combobox

private void button11\_Click\_1(object sender, EventArgs e)

{

zagruzkachb();

}

//вывод журнала занятий

private void button12\_Click\_1(object sender, EventArgs e)

{

string sql = @"SELECT

ЖурналЗанятий.КодЗанятия as Код,

ЖурналЗанятий.ДатаЗанятия,

Залы.Зал,

Группы.Группа,

Направления.Направление,

ЖурналЗанятий.Выполнено,

Расписание.КодСотрудника as [КодТренераПоПлану],

ЖурналЗанятий.КодСотрудника as [КодТренераПровел],

Расписание.КодПериода,

Расписание.КодРасписания,

Расписание.НомерДняНедели,

Направления.КодНаправления,

Группы.КодГруппы,

Количество

FROM Направления INNER JOIN (Группы INNER JOIN (Залы INNER JOIN (Расписание INNER JOIN ЖурналЗанятий ON Расписание.КодРасписания = ЖурналЗанятий.КодРасписания) ON Залы.КодЗала = Расписание.КодЗала) ON Группы.КодГруппы = Расписание.КодГруппы) ON Направления.КодНаправления = Группы.КодНаправления

WHERE (((Расписание.КодЗала)=" + cbzaljournal.SelectedValue + ") AND ((Расписание.КодПериода)=" + idperiod2.ToString() + "));";

dsjournal=new DataSet();

gridjournal.DataSource = null;

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

{

gridjournal.Columns.RemoveAt(0);

}

BaseFIT.gettable(sql, dsjournal);

dvjournal = new DataView(dsjournal.Tables[0]);

gridjournal.DataSource = dvjournal;

gridjournal.Columns[6].Visible = false; ;

gridjournal.Columns[7].Visible = false; ;

gridjournal.Columns[8].Visible = false; ;

gridjournal.Columns[9].Visible = false; ;

gridjournal.Columns[10].Visible = false; ;

gridjournal.Columns[11].Visible = false; ;

gridjournal.Columns[12].Visible = false; ;

gridjournal.Columns[3].Width = 200;//группа-

gridjournal.Columns[4].Width = 200;//направление

gridjournal.Columns[5].Width = 80;

gridjournal.Columns[0].Width = 50;

gridjournal.Columns[6].Width = 80;

sql = @"

SELECT

КодСотрудника,

ФИО

FROM Сотрудники";

DataSet ds1 = new DataSet();

OleDbDataAdapter da1 = new OleDbDataAdapter(sql, BaseFIT.con);

BaseFIT.con.Open();

da1.Fill(ds1);

BaseFIT.con.Close();

//добавляем столбцы combobox

DataGridViewComboBoxColumn comboBoxColumn =

new DataGridViewComboBoxColumn();

comboBoxColumn.DisplayStyle = DataGridViewComboBoxDisplayStyle.Nothing;

comboBoxColumn.ValueType = typeof(int);

comboBoxColumn.DataSource = ds1.Tables[0];

comboBoxColumn.ValueMember = "КодСотрудника";

comboBoxColumn.DisplayMember = "ФИО";

comboBoxColumn.DataPropertyName = "КодТренераПоПлану";

comboBoxColumn.HeaderText = "Тренер по плану";

comboBoxColumn.Width = 200;

gridjournal.Columns.Add(comboBoxColumn);

DataSet ds2 = new DataSet();

OleDbDataAdapter da2 = new OleDbDataAdapter(sql, BaseFIT.con);

BaseFIT.con.Open();

da2.Fill(ds2);

BaseFIT.con.Close();

DataGridViewComboBoxColumn comboBoxColumn2 =

new DataGridViewComboBoxColumn();

comboBoxColumn2.DisplayStyle = DataGridViewComboBoxDisplayStyle.ComboBox;

comboBoxColumn2.ValueType = typeof(int);

comboBoxColumn2.DataSource = ds2.Tables[0];

comboBoxColumn2.ValueMember = "КодСотрудника";

comboBoxColumn2.DisplayMember = "ФИО";

comboBoxColumn2.DataPropertyName = "КодТренераПровел";

comboBoxColumn2.HeaderText = "Кто провел";

comboBoxColumn2.Width = 200;

gridjournal.Columns.Add(comboBoxColumn2);

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

if ((gridjournal.Columns[i].HeaderText == "Выполнено") || (gridjournal.Columns[i].HeaderText == "Кто провел")

|| (gridjournal.Columns[i].HeaderText == "Количество") )

gridjournal.Columns[i].ReadOnly = false;

else gridjournal.Columns[i].ReadOnly = true;

}

//поиск

private void button44\_Click(object sender, EventArgs e)

{

BaseFIT.poisk(gridabonement, tbpoiskzakaz.Text);

}

private void button45\_Click(object sender, EventArgs e)

{

BaseFIT.poisk(gridsotrudnik, tbpoisksotr.Text);

}

private void button48\_Click(object sender, EventArgs e)

{

BaseFIT.poisk(gridklient, tbpoiskzakazchik.Text);

}

private void button49\_Click(object sender, EventArgs e)

{

BaseFIT.poisk(gridspr, tbpoiskspravochnik.Text);

}

private void toolStripButton9\_Click(object sender, EventArgs e)

{

checked\_knopka(sender);

tabControl1.SelectedIndex = 6;

}

public void proverkaras(int kodzala,int idper)

{

if ((kodzala == -1) || (idper== -1))

{

MessageBox.Show("Не выбран зал или период действия расписания");

return ;

}

OleDbCommand command1 = new OleDbCommand(@" SELECT COUNT(\*) from Расписание where КодПериода = " + idper.ToString()

+ @" and КодЗала =" + kodzala.ToString(), BaseFIT.con);

BaseFIT.con.Open(); //Открываем соединение с БД

object result = command1.ExecuteScalar();

int a = Convert.ToInt32(result);//Выполняем запрос.

BaseFIT.con.Close();

if (a > 0)

{

outraspisanie(kodzala, idper);

}

else

{

var resultt = MessageBox.Show("Для данного зала на выбранный период остутствует расписание. Будеи создано новое?", "Выбор",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

// If the no button was pressed ...

if (resultt == DialogResult.Yes)

{

BaseFIT.createraspisanie(kodzala, idper);

outraspisanie(kodzala, idper);

}

}

}

// формирование и вывод расписания

//делаем с помощью Label

public void outraspisanie(int kodzala, int idper)

{

string sql = @"SELECT DISTINCT ВремяНачала

FROM Расписание

WHERE КодПериода=" + idper.ToString()+" and кодзала=" + kodzala.ToString() + " order by ВремяНачала";

DataTable dttime = new DataTable();

OleDbDataAdapter da = new OleDbDataAdapter(sql, BaseFIT.con);

BaseFIT.con.Open();

da.Fill(dttime);

BaseFIT.con.Close();

string s = "";

rowcount = dttime.Rows.Count;

int interval = 2;

int wtime = 60;

int globalw = (panelrasp.Width - System.Windows.Forms.SystemInformation.VerticalScrollBarWidth - wtime - 8 \* interval) / 7;

int globalh = globalw \* 3 / 5;

int i ;

//создаем заголовки

//очищаем массив

for (i = 0; i <= 7; i++)

if (BaseFIT.zagolvki[i] != null)

{

panelrasp.Controls.Remove(BaseFIT.zagolvki[i]);

BaseFIT.zagolvki[i].Dispose();

}

//очищаем массив

for ( i = 0; i < 50; i++)

for (int jj = 0; jj <= 7; jj++)

{

if (BaseFIT.rooster[i, jj].MyLabel != null)

{

panelrasp.Controls.Remove(BaseFIT.rooster[i, jj].MyLabel);

BaseFIT.rooster[i, jj].MyLabel.Dispose();

}

}

//создаем заголовки массив

for (i = 0; i <=7; i++)

{

BaseFIT.zagolvki[i] = new Label();

BaseFIT.zagolvki[i].AutoSize = false;

if (i != 0) BaseFIT.zagolvki[i].Size = new Size(globalw, wtime);

else BaseFIT.zagolvki[i].Size = new Size(wtime, wtime);

BaseFIT.zagolvki[i].TextAlign = ContentAlignment.MiddleCenter;

BaseFIT.zagolvki[i].ForeColor = Color.White;

BaseFIT.zagolvki[i].Font = new System.Drawing.Font("Arial", 12);

if (i != 0)

BaseFIT.zagolvki[i].Location = new Point(interval + ((globalw + interval) \* i) - (globalw - wtime), 0);//+ Panel1.Size.Height + Panel2.Size.Height)

else BaseFIT.zagolvki[i].Location = new Point(interval + ((globalw + interval) \* i), 0);//+ Panel1.Size.Height + Panel2.Size.Height)

//new Point(5, (i + 1) \* 15);

string den = "";

switch (i)

{

case 1:

den = "Понедельник";

break;

case 2:

den = "Вторник";

break;

case 3:

den = "Среда";

break;

case 4:

den = "Четверг";

break;

case 5:

den = "Пятница";

break;

case 6:

den = "Суббота";

break;

case 7:

den = "Воскресенье";

break;

}

BaseFIT.zagolvki[i].Text = den;

BaseFIT.zagolvki[i].BackColor = Color.LightBlue;

BaseFIT.zagolvki[i].ForeColor = Color.Black;

panelrasp.Controls.Add(BaseFIT.zagolvki[i]);

}

int otstuph = BaseFIT.zagolvki[1].Height + interval;

//

// получаем данные по сотрудникам

sql = @"SELECT Расписание.КодРасписания, Расписание.ВремяНачала, Расписание.НомерДняНедели, Сотрудники.ИмяКороткое

FROM Сотрудники INNER JOIN Расписание ON Сотрудники.КодСотрудника = Расписание.КодСотрудника

WHERE (КодЗала=" + kodzala.ToString() + ") and (КодПериода=" + idper.ToString() + ")";

DataTable dtraspsotr = new DataTable();

OleDbDataAdapter daraspsotr = new OleDbDataAdapter(sql, BaseFIT.con);

BaseFIT.con.Open();

daraspsotr.Fill(dtraspsotr);

BaseFIT.con.Close();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// получаем данные для текста

sql = @"SELECT Расписание.ВремяНачала, Расписание.НомерДняНедели, Группы.Группа, Группы.Цвет

FROM Группы INNER JOIN Расписание ON Группы.КодГруппы = Расписание.КодГруппы

WHERE (КодЗала=" + kodzala.ToString() + ") and (КодПериода=" + idper.ToString() + ")";

DataTable dtrasp = new DataTable();

OleDbDataAdapter darasp = new OleDbDataAdapter(sql, BaseFIT.con);

BaseFIT.con.Open();

darasp.Fill(dtrasp);

BaseFIT.con.Close();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//это отдельный запрос для расстановки кодоврасписания

sql = @"SELECT КодРасписания, ВремяНачала, НомерДняНедели, кодзала

FROM Расписание WHERE (КодЗала=" + kodzala.ToString() + ") and (КодПериода=" + idper.ToString() + ")";

DataTable dtrasppoln = new DataTable();

OleDbDataAdapter darasppoln = new OleDbDataAdapter(sql, BaseFIT.con);

BaseFIT.con.Open();

darasppoln.Fill(dtrasppoln);

BaseFIT.con.Close();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i=-1; //начальное занчение массива

foreach (DataRow row in dttime.Rows)

{

i++;

s=row["ВремяНачала"].ToString();

for (int j = 0; j <= 7; j++)

{

foreach (DataRow row2 in dtrasppoln.Rows)

{

if ((Int32.Parse(row2["НомерДняНедели"].ToString()) == j) && (row2["ВремяНачала"].ToString() == s))

{

BaseFIT.rooster[i, j].kodrasp = Int32.Parse(row2["КодРасписания"].ToString());

BaseFIT.rooster[i, j].guppa = "";

BaseFIT.rooster[i, j].sotrudnik= "";

break;

}

}

BaseFIT.rooster[i, j].MyLabel = new Label();

BaseFIT.rooster[i, j].MyLabel.AutoSize = false;

if (j != 0) BaseFIT.rooster[i, j].MyLabel.Size = new Size(globalw, globalh);

else BaseFIT.rooster[i, j].MyLabel.Size = new Size(wtime, globalh);

BaseFIT.rooster[i, j].MyLabel.TextAlign = ContentAlignment.MiddleCenter;

BaseFIT.rooster[i, j].MyLabel.ForeColor = Color.White;

BaseFIT.rooster[i, j].MyLabel.MouseClick += new System.Windows.Forms.MouseEventHandler(this.Mylabel\_Click);

BaseFIT.rooster[i, j].MyLabel.Tag = i.ToString() + "/" + j.ToString();

if (j != 0) BaseFIT.rooster[i, j].MyLabel.Font = new System.Drawing.Font("Arial", 10);// Меняем шрифт, размер, начертание

else BaseFIT.rooster[i, j].MyLabel.Font = new System.Drawing.Font("Arial", 12);

if (j != 0)

BaseFIT.rooster[i, j].MyLabel.Location = new Point(interval + ((globalw + interval) \* j) - (globalw - wtime),otstuph+ interval + (globalh + interval) \* i);//+ Panel1.Size.Height + Panel2.Size.Height)

else BaseFIT.rooster[i, j].MyLabel.Location = new Point(interval + ((globalw + interval) \* j),otstuph+ interval + (globalh + interval) \* i);//+ Panel1.Size.Height + Panel2.Size.Height)

BaseFIT.rooster[i, j].MyLabel.Name = "Label" + i.ToString() + "\_" + j.ToString();

if (j != 0)

{

Boolean flag=false;

foreach (DataRow row1 in dtrasp.Rows)

{

//сначала проверяем группу

if ((Int32.Parse(row1["НомерДняНедели"].ToString()) == j) && (row1["ВремяНачала"].ToString() == s))

{

if (!(row1["Цвет"] is DBNull))

{

BaseFIT.rooster[i, j].MyLabel.BackColor = Color.FromArgb(Int32.Parse(row1["Цвет"].ToString()));

}

else

BaseFIT.rooster[i, j].MyLabel.BackColor = Color.Aqua;

BaseFIT.rooster[i, j].guppa = row1["Группа"].ToString();

flag=true;

break;

}

}

foreach (DataRow row1 in dtraspsotr.Rows)

{

//сначала проверяем группу

if ((Int32.Parse(row1["НомерДняНедели"].ToString()) == j) && (row1["ВремяНачала"].ToString() == s))

{

BaseFIT.rooster[i, j].sotrudnik = row1["ИмяКороткое"].ToString();

flag=true;

break;

}

}

if (!flag)

{BaseFIT.rooster[i, j].MyLabel.BackColor = Color.LightGray;

BaseFIT.rooster[i, j].MyLabel.Text ="-";

}

else

BaseFIT.rooster[i, j].MyLabel.Text = BaseFIT.rooster[i, j].guppa+Environment.NewLine+ BaseFIT.rooster[i, j].sotrudnik;

}

else

{

BaseFIT.rooster[i, j].MyLabel.Text = s;

BaseFIT.rooster[i, j].MyLabel.BackColor = Color.LightBlue;

BaseFIT.rooster[i, j].MyLabel.ForeColor = Color.Black;

}

panelrasp.Controls.Add(BaseFIT.rooster[i, j].MyLabel);

}

}

}

private void button19\_Click\_1(object sender, EventArgs e)

{

proverkaras(Int32.Parse(cbzall.SelectedValue.ToString()),idperiod);

}

private void button9\_Click\_1(object sender, EventArgs e)

{

tabControl1.SelectedIndex = 7;

}

private void button17\_Click\_1(object sender, EventArgs e)

{

ToExcelRasp( "Расписание для зала: "+cbzall.Text+" на период: "+tbnameperiod.Text, 1, 0);

}

private void button10\_Click\_1(object sender, EventArgs e)

{

ToExcel(gridjournal, "Журнал занятий. Зал: " + cbzaljournal.Text + ", на период: " + tbnameperiod2.Text, 1, 5);

}

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

{

foreach (ToolStripButton item in toolStrip2.Items)

{

item.DisplayStyle = System.Windows.Forms.ToolStripItemDisplayStyle.Image;

}

}

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

{

foreach (ToolStripButton item in toolStrip2.Items)

{

item.DisplayStyle = System.Windows.Forms.ToolStripItemDisplayStyle.ImageAndText;

}

}

private void gridworks\_MouseDoubleClick(object sender, MouseEventArgs e)

{

button19\_Click\_1(sender, e);

}

private void button51\_Click(object sender, EventArgs e)

{

BaseFIT.edittable("Заказы", gridabonement, true);

}

private void отчетыToolStripMenuItem1\_Click(object sender, EventArgs e)

{

tabControl1.SelectedIndex = 3;

}

private void button14\_Click(object sender, EventArgs e)

{

modified = false;

formsprcase w = new formsprcase("ПериодДействия", "select КодПериода, НазваниеПериода, НачалоПериода, КонецПериода from ПериодДействия");

w.ShowDialog();

if (modified)

{

tbnameperiod.Text = nameperiod;

}

}

//заменяем функцию щелчка по label в массиве rooster

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

{

//здесь самое интересное

if (Program.guest == true)

{

if (((Label)sender).Text == "-") return;

}

string s1=((Label)sender).Tag.ToString();

int i, j;

char ch = '/';

int ind = s1.IndexOf(ch);

if (ind >= 0)

{

string[] word = s1.Split(ch);

i = Convert.ToInt32(word[0]);

j = Convert.ToInt32(word[1]);

if (j == 0) return;

Program.modified = false;

fraspisanie w = new fraspisanie(BaseFIT.rooster[i, j].kodrasp);

w.ShowDialog();

if (Program.modified) //обновляем кнопку если что

{

string sql=@" SELECT Расписание.КодРасписания, Группы.Группа, Группы.Цвет, Сотрудники.ИмяКороткое

FROM Группы INNER JOIN (Сотрудники INNER JOIN Расписание ON Сотрудники.КодСотрудника = Расписание.КодСотрудника) ON Группы.КодГруппы = Расписание.КодГруппы

WHERE Расписание.КодРасписания=" + BaseFIT.rooster[i, j].kodrasp;

DataSet dsa = new DataSet();

BaseFIT.gettable(sql, dsa);

if (dsa.Tables[0].Rows.Count > 0)

{

BaseFIT.rooster[i, j].guppa = dsa.Tables[0].Rows[0]["Группа"].ToString();

BaseFIT.rooster[i, j].sotrudnik = dsa.Tables[0].Rows[0]["ИмяКороткое"].ToString();

if (!(dsa.Tables[0].Rows[0]["Цвет"] is DBNull)) BaseFIT.rooster[i, j].MyLabel.BackColor = Color.FromArgb(Int32.Parse(dsa.Tables[0].Rows[0]["Цвет"].ToString()));

BaseFIT.rooster[i, j].MyLabel.Text = BaseFIT.rooster[i, j].guppa + Environment.NewLine + BaseFIT.rooster[i, j].sotrudnik;

}

else

{

BaseFIT.rooster[i, j].guppa = "";

BaseFIT.rooster[i, j].sotrudnik = "";

BaseFIT.rooster[i, j].MyLabel.Text = "-";

BaseFIT.rooster[i, j].MyLabel.BackColor = Color.LightGray;

}

}

}

}

//столбец цвет на вкладке справочники группы

private void gridspr\_CellFormatting(object sender, DataGridViewCellFormattingEventArgs e)

{

// Это как раскрашивать ячейки по значениям

if (gridspr[e.ColumnIndex, e.RowIndex].Value != null)

if (gridspr.Columns[e.ColumnIndex].HeaderText == "Цвет")

{

int i = 0;

if (Int32.TryParse(gridspr[e.ColumnIndex, e.RowIndex].Value.ToString(), out i))

{

e.CellStyle.BackColor =Color.FromArgb(i);

e.CellStyle.ForeColor= Color.FromArgb(i);

}

}

}

private void button4\_Click(object sender, EventArgs e)

{

modified = false;

formsprcase w = new formsprcase("ПериодДействия2", "select КодПериода, НазваниеПериода, НачалоПериода, КонецПериода from ПериодДействия");

w.ShowDialog();

if (modified)

{

tbnameperiod2.Text = nameperiod2;

}

}

private void gridjournal\_CellBeginEdit(object sender, DataGridViewCellCancelEventArgs e)

{

this.CellValue = gridjournal.Rows[e.RowIndex].Cells[e.ColumnIndex].Value;

}

private void gridjournal\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

try

{

if ((gridjournal.Columns[e.ColumnIndex].HeaderText == "Количество"))

{

if (this.CellValue == gridjournal.Rows[e.RowIndex].Cells[e.ColumnIndex].Value) return;

int kol = 0;

if (!Int32.TryParse(gridjournal["Количество", e.RowIndex].Value.ToString(), out kol))

{

kol = 0;

MessageBox.Show("Ошибка ввода количества");

gridjournal.Rows[e.RowIndex].Cells[e.ColumnIndex].Value = 0;

return;

}

else

{

if (kol < 0 || kol >= 100)

{

MessageBox.Show("Ошибка ввода количества (>0 и меньше 100)");

return;

}

}

string sql = @"Update ЖурналЗанятий set Количество=" + gridjournal.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString() +

" where кодзанятия=" + gridjournal.Rows[e.RowIndex].Cells[0].Value.ToString();

OleDbCommand oleDbCommand = new OleDbCommand(sql, BaseFIT.con); //Запрос на вставку

BaseFIT.con.Open(); //Открываем соединение с БД

oleDbCommand.ExecuteNonQuery(); //Выполняем запрос.

BaseFIT.con.Close();

}

if ((gridjournal.Columns[e.ColumnIndex].HeaderText == "Кто провел"))

{

if (this.CellValue == gridjournal.Rows[e.RowIndex].Cells[e.ColumnIndex].Value) return;

string sql = @"Update ЖурналЗанятий set КодСотрудника=" + gridjournal.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString() +

" where кодзанятия=" + gridjournal.Rows[e.RowIndex].Cells[0].Value.ToString();

OleDbCommand oleDbCommand = new OleDbCommand(sql, BaseFIT.con); //Запрос на вставку

BaseFIT.con.Open(); //Открываем соединение с БД

oleDbCommand.ExecuteNonQuery(); //Выполняем запрос.

BaseFIT.con.Close();

}

if ((gridjournal.Columns[e.ColumnIndex].HeaderText == "Выполнено"))

{

if (this.CellValue == gridjournal.Rows[e.RowIndex].Cells[e.ColumnIndex].Value) return;

string sql = "";

if (Convert.ToBoolean(gridjournal.Rows[e.RowIndex].Cells[e.ColumnIndex].Value) == true)

{

sql = @"Update ЖурналЗанятий set Выполнено=true where кодзанятия=" + gridjournal.Rows[e.RowIndex].Cells[0].Value.ToString();

OleDbCommand oleDbCommand = new OleDbCommand(sql, BaseFIT.con); //Запрос на вставку

BaseFIT.con.Open(); //Открываем соединение с БД

oleDbCommand.ExecuteNonQuery(); //Выполняем запрос.

BaseFIT.con.Close();

if (gridjournal.CurrentRow.Cells["КодТренераПровел"].Value == DBNull.Value)

{

gridjournal.CurrentRow.Cells["КодТренераПровел"].Value = gridjournal.CurrentRow.Cells["КодТренераПоПлану"].Value;

sql = @"Update ЖурналЗанятий set КодСотрудника=" + gridjournal.CurrentRow.Cells["КодТренераПровел"].Value.ToString() +

" where кодзанятия=" + gridjournal.Rows[e.RowIndex].Cells[0].Value.ToString();

oleDbCommand = new OleDbCommand(sql, BaseFIT.con); //Запрос на вставку

BaseFIT.con.Open(); //Открываем соединение с БД

oleDbCommand.ExecuteNonQuery(); //Выполняем запрос.

BaseFIT.con.Close();

}

}

else

{

sql = @"Update ЖурналЗанятий set Выполнено=false,кодсотрудника=null where кодзанятия=" + gridjournal.Rows[e.RowIndex].Cells[0].Value.ToString();

OleDbCommand oleDbCommand = new OleDbCommand(sql, BaseFIT.con); //Запрос на вставку

BaseFIT.con.Open(); //Открываем соединение с БД

oleDbCommand.ExecuteNonQuery(); //Выполняем запрос.

BaseFIT.con.Close();

// как ячейку в dbnull перевести!!!!!!!!!!!

gridjournal.CurrentRow.Cells["КодТренераПровел"].Value = DBNull.Value; ;

}

}

}

catch

{

MessageBox.Show("Ошибка ввода");

}

finally

{

if (BaseFIT.con != null && BaseFIT.con.State != ConnectionState.Closed)

{

BaseFIT.con.Close();

}

}

}

private void button6\_Click\_1(object sender, EventArgs e)

{

tabControl1.SelectedIndex = 2;

}

private void button8\_Click(object sender, EventArgs e)

{

// кнопка фильтр в журнале занятий

if (dsjournal == null)

{ MessageBox.Show("Журнал не сформирован"); return; }

string filtr = "";

if (chbsotrvipolnil.Checked)

if (cbsotrvipolnil.SelectedIndex>-1)

{

string skod = cbsotrvipolnil.SelectedValue.ToString();

filtr = filtr + "(КодТренераПровел="+skod+") and ";

}

if (chbsotrplan.Checked)

if (cbsotrplan.SelectedIndex > -1)

{

string skod = cbsotrplan.SelectedValue.ToString();

filtr = filtr + "(КодТренераПоПлану=" + skod + ") and ";

}

if (chbdennedeli.Checked)

if (cbdennedeli.SelectedIndex > -1)

{

string skod = (cbdennedeli.SelectedIndex+1).ToString();

filtr = filtr + "(НомерДняНедели=" + skod + ") and ";

}

if (chbnapr.Checked)

if (cbnapr.SelectedIndex > -1)

{

string skod = cbnapr.SelectedValue.ToString();

filtr = filtr + "(КодНаправления=" + skod + ") and ";

}

if (chbgruppa.Checked)

if (tbnamegruppa.Text!="")

{

string skod =idgruppa.ToString();

filtr = filtr + "(КодГруппы=" + skod + ") and ";

}

if (rbvipolneno.Checked) filtr = filtr + "(Выполнено=true) and ";

else if (rbnevipolneno.Checked) filtr = filtr + "(Выполнено=false) and ";

if (rbperiod2.Checked)

{

if (dtpend2.Value.Date < dtpstart2.Value.Date)

{

MessageBox.Show("Конец периода должен быть больше или равен начала периода",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

return;

}

filtr = filtr + @"(((ДатаЗанятия)>=#" + dtpstart2.Value.ToString("MM'/'dd'/'yyyy") + @"# And

(ДатаЗанятия)<=#" + dtpend2.Value.ToString("MM'/'dd'/'yyyy") + "#)) and ";

}

if (rbzadatu.Checked)

{

filtr = filtr + @"(ДатаЗанятия=#" + dtpdata.Value.ToString("MM'/'dd'/'yyyy") + @"#) And ";

}

if (filtr != "")

{

filtr = filtr.Substring(0, filtr.Length - 6);

MessageBox.Show(filtr);

dvjournal.RowFilter = filtr;

}

else dvjournal.RowFilter = "";

}

//выбор группы для фильтра по журналу занятий

private void button13\_Click(object sender, EventArgs e)

{

modified = false;

formsprcase w = new formsprcase("Группы", @"SELECT Группы.КодГруппы, Группы.Группа, Направления.Направление

FROM Направления INNER JOIN Группы ON Направления.КодНаправления = Группы.КодНаправления");

w.ShowDialog();

if (modified)

{

tbnamegruppa.Text = namegruppa;

}

}

private void button7\_Click(object sender, EventArgs e)

{

dvjournal.RowFilter = "";

}

//далее настройка меню переход

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

{

tabControl1.SelectedIndex = 6;

}

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

{

tabControl1.SelectedIndex = 0;

}

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

{

tabControl1.SelectedIndex = 2;

}

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

{

tabControl1.SelectedIndex = 4;

}

private void СотрудникиToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

tabControl1.SelectedIndex = 5;

}

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

{

tabControl1.SelectedIndex =3;

}

private void ЗаставкаToolStripMenuItem1\_Click(object sender, EventArgs e)

{

tabControl1.SelectedIndex = 1;

}

}

}

ClassBaseFIT.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data;

using System.IO;

using System.Data.OleDb;

namespace FitnessClub

{

class ClassBaseFIT

{

public struct superlabel

{

public Label MyLabel;

public int kodrasp;

public string guppa;

public string sotrudnik;

}

public OleDbConnection con = new OleDbConnection();

public string connection;

OleDbDataAdapter daspr;

DataSet dsspr;

public string fiouser;

public int koduser;

public string loginuser;

public string roleuser;

public string doljnuser;

public string tekparol;

public superlabel[,] rooster=new superlabel[50,8];// массив супер label

public Label [] zagolvki = new Label[8];// массив супер label

//инициализация

public ClassBaseFIT()

{

string pathfile = Directory.GetCurrentDirectory() + "\\Path.inf";

if (File.Exists(pathfile))

{

try

{

FileStream fs = new FileStream(pathfile, FileMode.Open);

StreamReader sr = new StreamReader(fs, Encoding.UTF8);

connection = sr.ReadLine();

sr.Close();

}

catch (Exception ex)

{

connection = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=|DataDirectory|\\BaseFit.mdb;Persist Security Info=True";

}

}

else

{

connection = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=|DataDirectory|\\BaseFit.mdb;Persist Security Info=True";

}

con.ConnectionString = connection;

}

public Boolean editspravochniki(string roluser)

{

if (((Program.BaseFIT.roleuser == "Администратор")

|| (Program.BaseFIT.roleuser == "Менеджер")

|| (Program.BaseFIT.roleuser == "Системный администратор"))) return true;

else return false;

}

public void createraspisanie(int kodzala,int kodperioda)

{

string sql = "select времятекст from времяначала order by времятекст";

DataTable dttime = new DataTable();

con.Open();

OleDbDataAdapter da = new OleDbDataAdapter(sql, con);

da.Fill(dttime);

con.Close();

OleDbCommand oleDbCommand = new OleDbCommand("insert into Расписание " +

@" (КодЗала,

КодПериода, ВремяНачала,

НомерДняНедели) VALUES

( @КодЗала,

@КодПериода, @ВремяНачала,

@НомерДняНедели)", con);

//добавляем параметры

OleDbParameter oleDbParameter1 = new OleDbParameter("@КодЗала", OleDbType.Integer);

oleDbParameter1.Value = kodzala; oleDbCommand.Parameters.Add(oleDbParameter1);

OleDbParameter oleDbParameter2 = new OleDbParameter("@КодПериода", OleDbType.Integer);

oleDbParameter2.Value = kodperioda; oleDbCommand.Parameters.Add(oleDbParameter2);

string s = "";

//Добавляем параметр

try

{

con.Open(); //Открываем соединение с БД

foreach (DataRow row in dttime.Rows)

{

s = row["времятекст"].ToString();

for

(int i = 1; i <= 7; i++)

{

oleDbCommand.CommandText = @"insert into Расписание " +

@" (КодЗала,

КодПериода, ВремяНачала,

НомерДняНедели) VALUES

( @КодЗала,

@КодПериода, '" + s + "'," + i.ToString() + ")";

oleDbCommand.ExecuteNonQuery(); //Выполняем запрос.

//Закрываем соединение

}

}

con.Close();

MessageBox.Show("Расписание создано");

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

finally

{

if (con != null && con.State != ConnectionState.Closed)

{

con.Close();

}

}

}

//получаем данные по сотруднику

public void Login( String s, String p, ref Int32 count)

{

string sql = @"

SELECT

Сотрудники.Логин,

Сотрудники.Пароль,

Сотрудники.ФИО,

Роли.Роль,

Должности.Должность,

Сотрудники.КодСотрудника

FROM Роли INNER JOIN

(Должности INNER JOIN Сотрудники ON Должности.КодДолжности = Сотрудники.КодДолжности) ON Роли.КодРоли = Сотрудники.КодРоли

WHERE (Сотрудники.Логин = '" + s + "') AND (Сотрудники.Пароль = '" + p + "')";

try

{

con.Open();

DataSet ds = new DataSet();

OleDbDataAdapter da = new OleDbDataAdapter(sql, con);

DataTable dt = new DataTable();

da.Fill(dt);

con.Close();

con.Open();

count = dt.Rows.Count;

if (count > 0)

{

string passwordDB = dt.Rows[0].ItemArray[1].ToString();//доступ для конкретной строки

if (p == passwordDB.Trim())

{

count = 1;

loginuser = dt.Rows[0].ItemArray[0].ToString();

tekparol = dt.Rows[0].ItemArray[1].ToString();

fiouser = dt.Rows[0].ItemArray[2].ToString();

roleuser = dt.Rows[0].ItemArray[3].ToString();

koduser = Int32.Parse(dt.Rows[0].ItemArray[5].ToString());

doljnuser = dt.Rows[0].ItemArray[4].ToString();

koduser = Convert.ToInt32(dt.Rows[0].ItemArray[5]);

}

else

{

count = 0;

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

finally

{

if (con != null && con.State != ConnectionState.Closed)

{

con.Close();

}

}

}

//универсальный метод добавления данных в grid

public void gettable(string sql, DataSet ds)

{

try

{

con.Open();

OleDbDataAdapter da = new OleDbDataAdapter(sql, con);

da.Fill(ds);

con.Close();

}

catch

{

MessageBox.Show("Ошибка получения данных из таблицы");

}

finally

{

if (con != null && con.State != ConnectionState.Closed)

{

con.Close();

}

}

}

//универсальный метод добавления данных в grid

public void gettablesotrudnik( ref OleDbDataAdapter da)

{

string sql = @"SELECT

Сотрудники.КодСотрудника,

Сотрудники.ФИО,

Сотрудники.ИмяКороткое,

Сотрудники.Телефон,

Сотрудники.Образование,

Должности.Должность

FROM Должности INNER JOIN Сотрудники ON Должности.КодДолжности = Сотрудники.КодДолжности;

";

con.Open();

da = new OleDbDataAdapter(sql, con);

con.Close();

}

//получение таблицы Клиент

public void gettableklient(ref OleDbDataAdapter da)

{

string sql = @"SELECT

КодКлиента, Клиент,

Телефон, Email, Примечания

FROM Клиенты";

con.Open();

da = new OleDbDataAdapter(sql, con);

con.Close();

}

//поиск в таблице

public void poisk(DataGridView grid,string s)

{

if (s.Trim() == "")

{

MessageBox.Show("Нет запроса");

return;

}

for (int i = 0; i < grid.RowCount; i++)

{

for (int j = 0; j < grid.ColumnCount; j++)

{

grid.Rows[i].Cells[j].Selected = false;

if (grid.Rows[i].Cells[j].Value != null)

if (grid.Rows[i].Cells[j].Value.ToString().ToLower().Contains(s.ToLower()))

{

grid.Rows[i].Cells[j].Selected = true;

break;

}

}

}

}

//универсальный метод получения значения

public string getname(string sql)

{

con.Open();

OleDbDataAdapter da = new OleDbDataAdapter(sql, con);

DataSet ds = new DataSet();

da.Fill(ds);

con.Close();

string s = "";

if (ds.Tables[0].Rows.Count==1)

s = ds.Tables[0].Rows[0][0].ToString();

return s;

}

//универсальный метод добавления данных в grid поставщика

public void gettablevidabonementa(ref OleDbDataAdapter da)

{

string sql = @"SELECT ВидыАбонементов.КодВидаАбонемента,

НазванияВидовАбонемента.НазваниеВидаАбонемента, Залы.Зал, ВидыАбонементов.Цена,

ВидыАбонементов.КоличествоПосещений

FROM Залы INNER JOIN

(НазванияВидовАбонемента INNER JOIN ВидыАбонементов

ON НазванияВидовАбонемента.КодНазванияВидаАбонемента = ВидыАбонементов.КодНазванияВидаАбонемента)

ON Залы.КодЗала = ВидыАбонементов.КодЗала order by Залы.Зал, ВидыАбонементов.КоличествоПосещений";

con.Open();

da = new OleDbDataAdapter(sql, con);

con.Close();

}

//универсальный метод добавления данных в grid периоддействия

public void gettableperiod(ref OleDbDataAdapter da)

{

string sql = @"SELECT КодПериода,

НазваниеПериода as Название,

НачалоПериода as Начало,

КонецПериода as Конец

FROM ПериодДействия order by НачалоПериода Desc;

";

con.Open();

da = new OleDbDataAdapter(sql, con);

con.Close();

}

//универсальный метод добавления данных в grid направлений групп

public void gettablenapr(ref OleDbDataAdapter da)

{

string sql = @"SELECT КодНаправления,

Направление

FROM Направления;

";

con.Open();

da = new OleDbDataAdapter(sql, con);

con.Close();

}

//универсальный метод добавления данных в grid групп

public void gettablegruppa(ref OleDbDataAdapter da)

{

string sql = @"SELECT Группы.КодГруппы, Группы.Группа, Направления.Направление, Группы.Цвет

FROM Направления INNER JOIN Группы ON Направления.КодНаправления = Группы.КодНаправления";

con.Open();

da = new OleDbDataAdapter(sql, con);

con.Close();

}

//универсальный метод добавления данных в grid зал

public void gettablezal(ref OleDbDataAdapter da)

{

string sql = @"SELECT

Залы.КодЗала,

Залы.Зал,

Залы.Площадь,

Залы.Вместимость,

Залы.Адрес

FROM Залы;

";

con.Open();

da = new OleDbDataAdapter(sql, con);

con.Close();

}

//универсальный метод добавления данных в grid абонементы

public void gettableabonement(ref OleDbDataAdapter da)

{

string sql = @"SELECT Абонементы.КодАбонемента, Клиенты.Клиент, Клиенты.Телефон, НазванияВидовАбонемента.НазваниеВидаАбонемента as [Абонемент],

Залы.Зал, Абонементы.ИтогоКОплате as [К оплате], Абонементы.Оплачено, Абонементы.ДатаНачала as [C], Абонементы.ДатаКонца as По, Сотрудники.ФИО as Менеджер

FROM Залы INNER JOIN (Сотрудники INNER JOIN (Клиенты INNER JOIN (НазванияВидовАбонемента INNER JOIN (ВидыАбонементов INNER JOIN Абонементы

ON ВидыАбонементов.КодВидаАбонемента = Абонементы.КодВидаАбонемента)

ON НазванияВидовАбонемента.КодНазванияВидаАбонемента = ВидыАбонементов.КодНазванияВидаАбонемента)

ON Клиенты.КодКлиента = Абонементы.КодКлиента) ON Сотрудники.КодСотрудника = Абонементы.КодСотрудника) ON Залы.КодЗала = ВидыАбонементов.КодЗала";

con.Open();

da = new OleDbDataAdapter(sql, con);

con.Close();

}

//вставка записи

public void insertrecord(string nametable, DataGridView grid, Boolean updategrid)

{

Program.modified = false;

Program.insert = true;

selectrecord(nametable, 0, "");

//обновляем таблицу если изменяли

if (Program.modified && updategrid)

{

selecttable(nametable, grid);

grid.CurrentCell = grid.Rows[grid.RowCount - 1].Cells[1];

}

}

//выбор записей

public void selectrecord(string nametable, int id,string name)

{

switch (nametable)

{

case "НазванияВидовАбонемента":

FormSpr w4 = new FormSpr("НазванияВидовАбонемента", id, "КодНазванияВидаАбонемента", "НазваниеВидаАбонемента", name);

w4.ShowDialog();

break;

case "Должности":

FormSpr w5 = new FormSpr("Должности", id, "КодДолжности", "Должность", name);

w5.ShowDialog();

break;

case "Абонементы":

fabonement w6 = new fabonement(id);

w6.ShowDialog();

break;

case "Сотрудники":

fsotrudnik w12 = new fsotrudnik(id);

w12.ShowDialog();

break;

case "Клиенты":

fklient w13 = new fklient(id);

w13.ShowDialog();

break;

case "Направления":

fnapr w13n = new fnapr(id);

w13n.ShowDialog();

break;

case "Группы":

fgrupp w13nn = new fgrupp(id);

w13nn.ShowDialog();

break;

case "Залы":

fobject w14 = new fobject(id);

w14.ShowDialog();

break;

case "ПериодДействия":

fperiod w15 = new fperiod(id);

w15.ShowDialog();

break;

case "ВидыАбонементов":

fvidabonementa w16 = new fvidabonementa(id);

w16.ShowDialog();

break;

default: // все иное

MessageBox.Show("Таблица " + nametable + " не найдена");

return;

} //конец case

}

//удаление записей

public void deleterecord(string nametable, DataGridView grid)

{

try

{

int nrows = indexselectedgrid(grid);

if (nrows == -1)

{

MessageBox.Show("Не выбрана запись");

return;

}

int index = 0;

if (!Int32.TryParse(grid[0, nrows].Value.ToString(), out index))

{

MessageBox.Show("Ошибка удаления ");

return;

}

string pole = grid.Columns[0].HeaderText;

OleDbCommand command1 = new OleDbCommand();

switch (nametable)

{

case "Группы":

command1 = new OleDbCommand(@" SELECT COUNT(\*) from Расписание where КодГруппы = " + index.ToString(), con);

break;

case "НазванияВидовАбонемента":

command1 = new OleDbCommand(@" SELECT COUNT(\*) from ВидыАбонементов where КодНазванияВидаАбонемента = " + index.ToString(), con);

break;

case "Направления":

command1 = new OleDbCommand(@" SELECT COUNT(\*) from Группы where Коднаправления = " + index.ToString(), con);

break;

case "Должности":

command1 = new OleDbCommand(@" SELECT COUNT(\*) from Сотрудники where коддолжности = " + index.ToString(), con);

break;

case "Абонементы":

command1 = new OleDbCommand(@" SELECT COUNT(\*) from Абонементы where кодабонемента = " + index.ToString()+

" and оплачено=true", con);

break;

case "Сотрудники":

command1 = new OleDbCommand(@" SELECT COUNT(\*) from ЖурналЗанятий where (КодСотрудника = " + index.ToString() +

") ", con);

break;

case "Клиенты":

command1 = new OleDbCommand(@" SELECT COUNT(\*) from Абонементы where КодКлиента = " + index.ToString(), con);

break;

case "ПериодДействия":

command1 = new OleDbCommand(@" SELECT COUNT(\*) from Расписание where КодПериода = " + index.ToString(), con);

break;

case "ВидыАбонементов":

command1 = new OleDbCommand(@" SELECT COUNT(\*) from Абонементы where КодВидаАбонемента = " + index.ToString(), con);

break;

case "Залы":

command1 = new OleDbCommand(@" SELECT COUNT(\*) from Расписание where КодЗала= " + index.ToString(), con);

break;

default:

MessageBox.Show("Таблица отсутствует в списке для обработки");

return;

}

//дополнительная проверка

if ((nametable == "Залы"))

{

command1 = new OleDbCommand(@" SELECT COUNT(\*) from ВидыАбонементов where КодЗала= " + index.ToString(), con);

con.Open(); //Открываем соединение с БД

object result = command1.ExecuteScalar();

int a = Convert.ToInt32(result);//Выполняем запрос.

con.Close();

if (a > 0)

{

MessageBox.Show("Ошибка удаления, на зал есть виды абонементов.",

"Ошибка удаления данных", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

return;

}

}

{

con.Open(); //Открываем соединение с БД

object result = command1.ExecuteScalar();

int a = Convert.ToInt32(result);//Выполняем запрос.

con.Close();

if (a > 0)

{

if (nametable == "Абонементы")

MessageBox.Show("Ошибка удаления, абонемент оплачен",

"Ошибка удаления данных", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

else MessageBox.Show("Ошибка удаления, данная запись используется в других таблицах.",

"Ошибка удаления данных", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

return;

}

}

//дополнительная проверка

if ((nametable == "Сотрудники"))

{

command1 = new OleDbCommand(@" SELECT COUNT(\*) from Расписание where КодСотрудника= " + index.ToString(), con);

con.Open(); //Открываем соединение с БД

object result = command1.ExecuteScalar();

int a = Convert.ToInt32(result);//Выполняем запрос.

con.Close();

if (a > 0)

{

MessageBox.Show("Ошибка удаления, в Расписании есть записи связанные с данным сотрудником.",

"Ошибка удаления данных", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

return;

}

}

if ((nametable == "Сотрудники"))

{

command1 = new OleDbCommand(@" SELECT COUNT(\*) from ЖурналЗанятий where КодСотрудника= " + index.ToString(), con);

con.Open(); //Открываем соединение с БД

object result = command1.ExecuteScalar();

int a = Convert.ToInt32(result);//Выполняем запрос.

con.Close();

if (a > 0)

{

MessageBox.Show("Ошибка удаления, в Журнале занятий есть записи связанные с данным сотрудником.",

"Ошибка удаления данных", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

return;

}

}

string s = "Вы действительно хотите удалить ";

if (nametable == "Абонементы") s = "Вы действительно хотите удалить абонемент клиента ";

var resultt = MessageBox.Show(s + grid[1, nrows].Value.ToString() + "?", "Выбор",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

// If the no button was pressed ...

if (resultt == DialogResult.Yes)

{

command1 = new OleDbCommand(@"delete FROM " + nametable + " where " + pole + "=" + index.ToString(), con);

con.Open();

command1.ExecuteNonQuery();

con.Close();

MessageBox.Show("Удаление прошло успешно ");

grid.Rows.RemoveAt(nrows);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка удаления " + ex.ToString(),

"Ошибка удаления данных", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

return;

}

finally

{

if (con != null && con.State != ConnectionState.Closed)

{

con.Close();

}

}

}

//проверка выбора строки в таблице

public int indexselectedgrid(DataGridView grid)

{

if (grid.Rows.Count == 0)

{

return -1;

}

if (grid.CurrentRow != null)

return grid.CurrentRow.Index;

else return -1;

}

//редактирование записи

public void edittable(string nametable, DataGridView grid, Boolean updategrid)

{

int nrows = indexselectedgrid(grid);

if (nrows == -1)

{

MessageBox.Show("Не выбрана строка для редактирования");

return;

}

int id = Int32.Parse(grid[0, nrows].Value.ToString());

Program.insert = false;

Program.modified = false;

selectrecord(nametable, id, grid[1, nrows].Value.ToString());

if (Program.modified && updategrid)

{

selecttable(nametable, grid);

grid.CurrentCell = grid.Rows[nrows].Cells[1];

}

}

//выборка из таблицы

public void selecttable(string nametable, DataGridView grid)

{

try

{

switch (nametable)

{

case "Сотрудники":

gettablesotrudnik(ref daspr);

dsspr = new DataSet();

daspr.Fill(dsspr);

grid.DataSource = dsspr.Tables[0];

break;

case "Залы":

gettablezal(ref daspr);

dsspr = new DataSet();

daspr.Fill(dsspr);

grid.DataSource = dsspr.Tables[0];

grid.Columns[0].Visible = false;

break;

case "ВидыАбонементов":

gettablevidabonementa(ref daspr);

dsspr = new DataSet();

daspr.Fill(dsspr);

grid.DataSource = dsspr.Tables[0];

grid.Columns[0].Visible = false;

break;

case "ПериодДействия":

gettableperiod(ref daspr);

dsspr = new DataSet();

daspr.Fill(dsspr);

grid.DataSource = dsspr.Tables[0];

grid.Columns[0].Visible = false;

break;

case "Клиенты":

gettableklient(ref daspr);

dsspr = new DataSet();

daspr.Fill(dsspr);

grid.DataSource = dsspr.Tables[0];

break;

case "Направления":

gettablenapr(ref daspr);

dsspr = new DataSet();

daspr.Fill(dsspr);

grid.DataSource = dsspr.Tables[0];

grid.Columns[0].Visible = false;

break;

case "Группы":

gettablegruppa(ref daspr);

dsspr = new DataSet();

daspr.Fill(dsspr);

grid.DataSource = dsspr.Tables[0];

grid.Columns[0].Visible = false;

break;

case "Абонементы":

gettableabonement(ref daspr);

dsspr = new DataSet();

daspr.Fill(dsspr);

grid.DataSource = dsspr.Tables[0];

break;

default:

string sql = "Select \* from " + nametable; //универсальный метод добавления данных в grid

dsspr = new DataSet();

gettable(sql, dsspr);

grid.DataSource = dsspr.Tables[0];

grid.Columns[0].Visible = false;

break;

}//switch

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

# Приложение Б. Руководство пользователя