

Содержание

Введение.....	3
1 Структурный системный анализ	5
1.1 Описание предметной области	7
1.2 Разработка функциональной модели информационной системы.....	8
2 Реализация и разработка проекта базы данных	11
2.1 Инфологическое проектирование	11
2.2 Логическая модель данных	15
2.3 Физическая структура информационной системы	16
3 Разработка интерфейса	17
3.1 Разработка формы меню.....	17
Заключение	68
Список информационных источников.....	69

					ПД.09.02.07.XXXX.ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Кураедов Д.В			Разработка информационной системы для ведения отчётов спортклуба «SportClub»	Лит.	Лист	Листов
Провер.							2	70
Реценз.						ЙОТК И-41		
Н. Контр.								
Утверд.								

Тема «Разработка информационной системы для ведения отчётов спортклуба «SportClub»»

Введение

Для успешного и эффективного функционирования практически любой системы необходимы ведение контроля, анализа и внедрение автоматизации для отдельных процессов или системы в целом. Автоматизация позволяет повысить производительность и качество системы, оптимизировать процессы управления, снизить затраты.

Целью данной дипломной работы является проектирование информационной системы спортклуба, которая позволила бы значительно улучшить качество обслуживания клиентов и упростить процессы оформления абонементов, а также достичь максимального результата работы спортклуба.

Под информационной системой понимают любой объект, который одновременно рассматривается и как единое целое, и как объединенная в интересах достижения поставленных целей совокупность разнородных элементов. Системы значительно отличаются между собой как по составу, так и по главным целям.

В информатике понятие «система» широко распространено и имеет множество смысловых значений. Чаще всего оно используется применительно к набору технических средств и программ. Системой может называться аппаратная часть компьютера. Системой может также считаться множество программ для решения конкретных прикладных задач, дополненных процедурами ведения документации и управления расчетами. Добавление к понятию «система» слова «информационная» отражает цель ее создания и функционирования. Информационные системы обеспечивают сбор, хранение, обработку, поиск, выдачу информации, необходимой в процессе принятия решений задач из любой области. Они помогают анализировать проблемы и создавать новые продукты.

Информационная система – взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Информационная система представляет собой хранилище информации, снабженное процедурами ввода, поиска, размещения и выдачи информации. Наличие таких процедур – главная особенность информационных систем, отличающих их от простых скоплений информационных материалов.

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		3

Для описания предметной области и проектируемой системы будем использовать диаграммы UML.

Unified Modeling Language (UML) — унифицированный язык моделирования. Расшифруем: *modeling* подразумевает создание модели, описывающей объект. *Unified* (универсальный, единый) — подходит для широкого класса проектируемых программных систем, различных областей приложений, типов организаций, уровней компетентности, размеров проектов. UML описывает объект в едином заданном синтаксисе, поэтому где бы вы не нарисовали диаграмму, ее правила будут понятны для всех, кто знаком с этим графическим языком — даже в другой стране.

Одна из задач UML — служить средством коммуникации внутри команды и при общении с заказчиком. Давайте рассмотрим возможные варианты использования диаграмм.

- Проектирование. UML-диаграммы помогут при моделировании архитектуры больших проектов, в которой можно собрать как крупные, так и более мелкие детали и нарисовать каркас (схему) приложения. По нему впоследствии будет строиться код.

- Реверс-инжиниринг — создание UML-модели из существующего кода приложения, обратное построение. Может применяться, например, на проектах поддержки, где есть написанный код, но документация неполная или отсутствует.

- Из моделей можно извлекать текстовую информацию и генерировать относительно удобочитаемые тексты — документировать.

Текст и графика будут дополнять друг друга.

Как и любой другой язык, UML имеет собственные правила оформления моделей и синтаксис. С помощью графической нотации UML можно визуализировать систему, объединить все компоненты в единую структуру, уточнять и улучшать модель в процессе работы. На общем уровне графическая нотация UML содержит 4 основных типа элементов:

- фигуры;
- линии;
- значки;
- надписи.

UML-нотация является де-факто отраслевым стандартом в области разработки программного обеспечения, ИТ-инфраструктуры и бизнес-систем.

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		4

1. Структурный системный анализ.

Спортклуб называется “SportClub”

В современном обществе информация стала полноценным ресурсом производства, важным элементом социальной и политической жизни общества. Качество информации определяет качество управления.

В последнее время все больше предприятий сталкиваются с проблемой улучшения управляемости компании: улучшение контроля и ускорение бизнес-процессов, улучшение возможности их отслеживания, оптимизация рабочего времени, экономия трудозатрат, повышение производительности труда и так далее. Единственным способом реализации подобных задач является внедрение информационной системы.

Применение современных информационных технологий имеет важное значение для оптимизации внутренних процессов организации, оперативного доведения информации до исполнителей, улучшения взаимодействия подразделений и отдельных исполнителей в процессе работы с документами, контроля исполнения документов и поручений, поиска информации и определения стадии исполнения документов и их местонахождения, то есть, в конечном счете, способствует более оперативному и качественному решению вопросов, которым посвящены документы. Главное при этом - улучшение взаимодействия всех подразделений организации, повышение управляемости, а также достижение более высокой оперативности в работе.

Актуальность темы определяется тем, что информационные системы составляют в настоящее время основу компьютерного обеспечения информационных процессов, входящих практически во все сферы человеческой деятельности автоматизировать процесс ведения документации и отчетности.

О своевременности и актуальности рассматриваемой проблемы говорит тот факт, что большую часть своего времени, администратор спортклуба тратит на оформление различной документации и отчетов.

					ПД.09.02.07.XXXX.ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Выше изложенное в целом определило **цель исследования**: повышение эффективности работы администратора спортклуба за счет разработки и внедрения информационной системы.

Данная информационная система предназначена для хранения информации о ценах абонементов у клиентов, об абонементах, о экспертах, сотрудничающих с турагентством и, с возможностью внесения данных, выборки и изменения данных, вывода информации в необходимом формате.

Объект исследования: работа администратором спортклуба.

Предмет исследования: информационная система, автоматизирующая работу администратора спортклуба.

В соответствии с поставленной целью в проекте определены следующие задачи исследования:

- На основе теоретического анализа литературы и Internet-источников произвести анализ предметной области спортклуба.
- Провести функционально-ориентированное проектирование информационной системы.
- Разработать инфологическую модель информационной системы.
- Спроектировать логическую структуру информационной системы.
- Разработать физическую структуру информационной системы.
- Разработать запросы и отчеты к информационной системе.
- Разработать интерфейс БД.
- Создать руководство пользователя.

1.1 Описание предметной области.

При разработке информационной системы было проведено обследование деятельности спортклуба по следующим источникам:

- Администратором спортклуба были предоставлены необходимые нормативные документы по правилам оформления абонеента и прочее;
- журнал, содержащий условия заключения договора с различными экспертами;
- литература и Internet-источники, описывающие работу спортклуба.

Таким образом, в результате обследования предметной области были определены следующие **входные данные**:

- информация о клиентах,
- информация о экспертах,
- информация об абонеентах,
- информация о платежах, информация об условиях оформления.

К **выходным данным** относятся отчеты об оплатах, сведения о экспертах.

Для создания информационной модели спортклуба необходимо осуществить формальное описание его работы. Изучение руководящих документов является первым шагом в изучении процессов работы спортклуба. Затем, на основе эталонных знаний, исследуется реальная работа администратора спортклуба и сравнивается с указанием руководящих документов. Многократное изучение руководящих документов, периодическое наблюдение за реальными действиями администратора и комментарии этого работника позволяют получить знания о функционировании работы спортклуба, разработать модели.

Целью решения данной задачи является сведение к минимуму работы администратора с бумажными носителями, что ускорит процесс обработки поступающей информации, сократив время ожидания для приёма заявок на оформление абонеента от клиентов, исключит возможную путаницу информации.

					ПД.09.02.07.XXXX.ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1.2 Разработка функциональной модели информационной системы

Проектирование информационной системы начинается с этапа построения бизнес-процессов.

Для описания бизнес-процессов, подлежащих учету в информационной системе «Спортклуб», используется функциональное моделирование.

Любое действие может быть декомпозировано, т.е. разложено на более мелкие, которые, в свою очередь могут быть декомпозированы, и т.д. до уровня разумной достаточности.

Диаграмма вариантов использования в UML — диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному разработчику и пользователю совместно обсуждать проектируемую или существующую систему.

При моделировании системы с помощью диаграммы прецедентов системный аналитик стремится:

- чётко отделить систему от её окружения;
- определить действующих лиц (акторов), их взаимодействие с системой и ожидаемую функциональность системы;
- определить в глоссарии предметной области понятия, относящиеся к детальному описанию функциональности системы (то есть прецедентов).

Работа над диаграммой может начаться с текстового описания, полученного при работе с заказчиком. При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов опускаются (для них составляется другой документ).

На диаграмме использования изображаются:

- акторы — группы лиц или систем, взаимодействующих с нашей системой;
- варианты использования (прецеденты) — сервисы, которые наша система предоставляет акторам;
- комментарии;
- отношения между элементами диаграммы.

На мой взгляд, наиболее правильный порядок построения диаграммы следующий:

- выделить группы действующих лиц (работающих с системой по-разному, часто из-за различных прав доступа);
- идентифицировать как можно больше вариантов использования (процессов, которые могут выполнять пользователи). При этом не следует делить процессы слишком мелко, нужно выбирать лишь те, которые дадут пользователю значимый результат. Например, кассир может «продать товар» (это будет являться прецедентом), однако «ввод штрих-кода товара для получения цены» самостоятельным прецедентом не является;
- дополнить прецеденты словесным описанием (сценарием):
 - a.** для каждого прецедента создать разделы: «главная последовательность» и «альтернативные последовательности»;
 - b.** при составлении сценария нужно упорно задавать заказчику вопросы «что происходит?», «что дальше?», «что еще может происходить?» и записывать ответы на них.

Сценарии являются очень важной частью диаграмм использования, хотя их формат и не регламентирован. Ряд авторов предлагает использовать псевдокод для представления сценария и даже сразу строить диаграммы деятельности или взаимодействия, но на мой взгляд, наиболее предпочтительным вариантом на этапе построения use-case диаграмм является текстовый, описывающий систему с точки зрения пользователя (т.к. именно этот формат будет наиболее понятен заказчику, с которым вам предстоит согласовывать техническое задание).

Изображение диаграммы прецедентов представлено на рисунке 1

					ПД.09.02.07.XXXX.ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

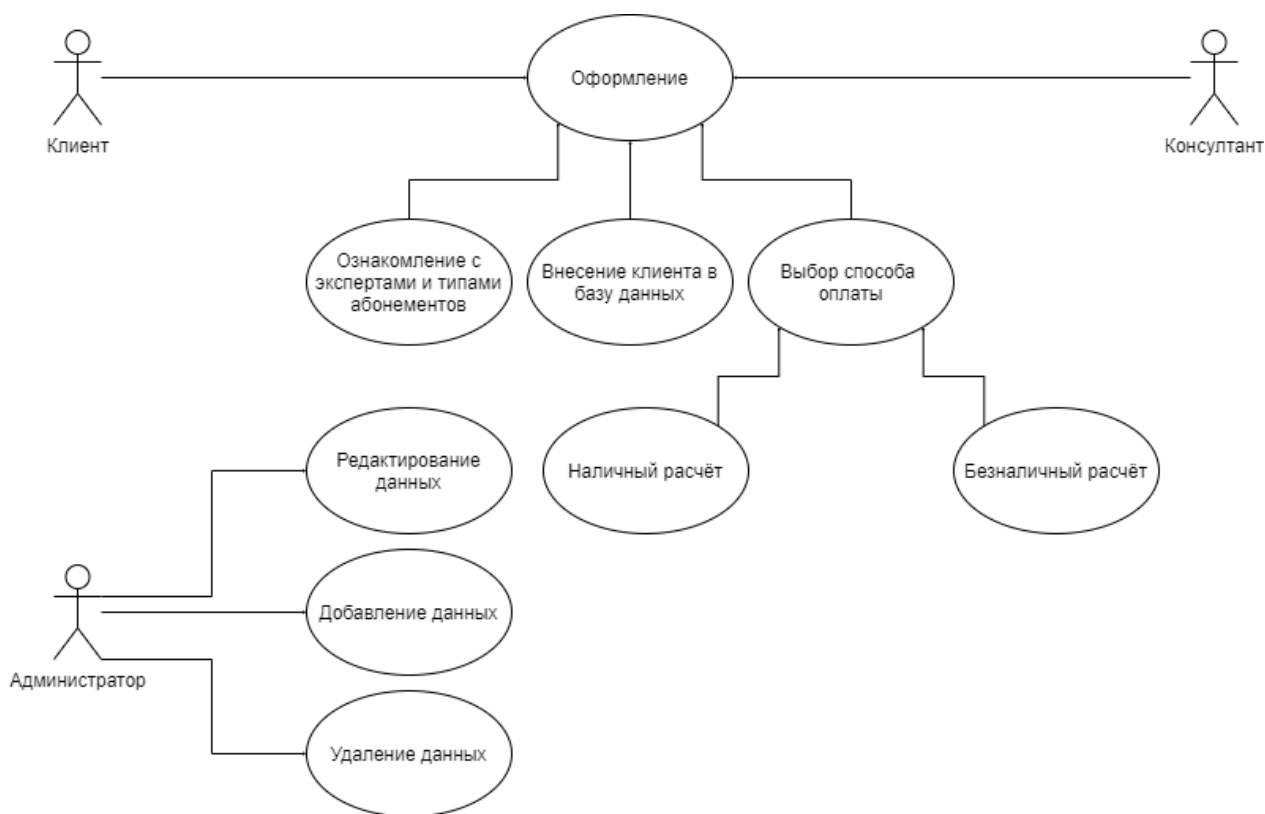


Рисунок 1. Диаграмма прецедентов

Диаграмма последовательности (англ. *sequence diagram*) — UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) информационной системы в рамках прецедента.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни» (англ. *lifeline*), отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), стрелки, показывающие обмен сигналами или сообщениями между объектами

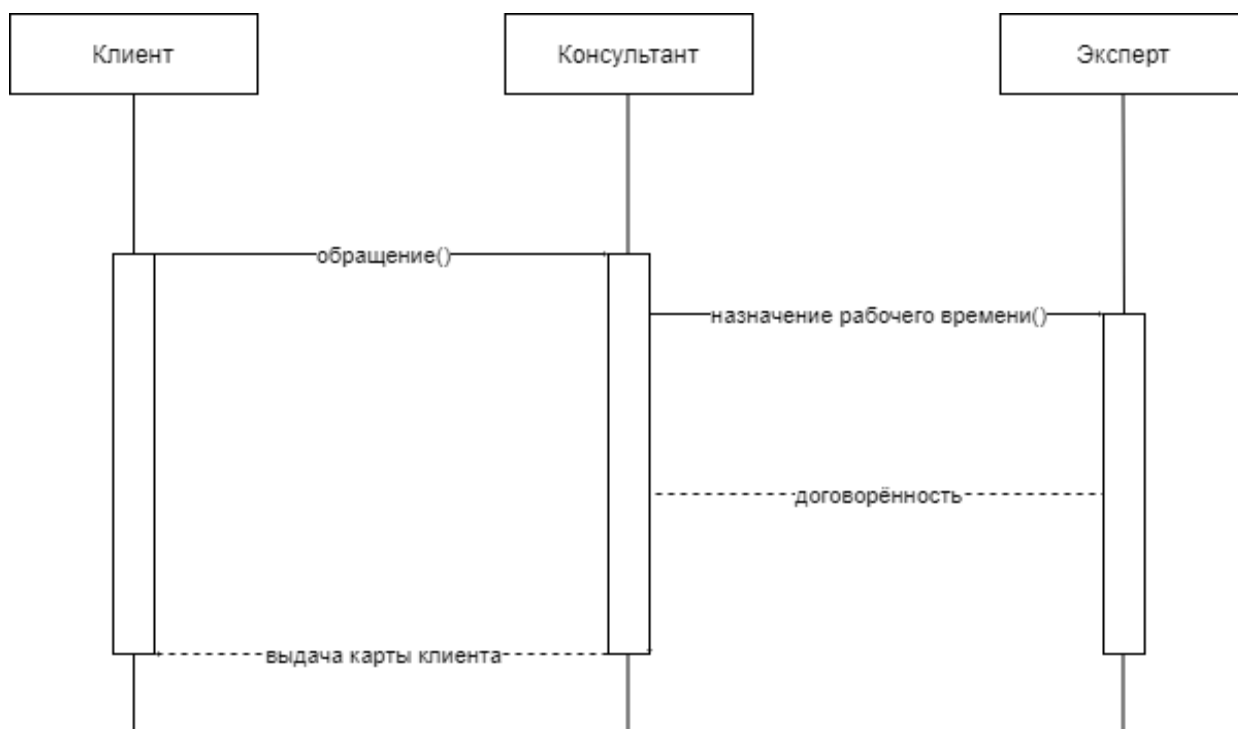


Рисунок 2. Диаграмма последовательности

2. Разработка и реализация проекта базы данных

2.1 Инфологическое проектирование

ER-модель (от англ. *Entity-Relationship model*, модель «сущность — связь») — модель данных, позволяющая описывать концептуальные схемы предметной области.

ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время проектирования баз данных происходит преобразование схемы, созданной на основе ER-модели, в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма «сущность-связь» (англ. *Entity-Relationship diagram*, *ERD*, *ER-диаграмма*).

Понятия «ER-модель» и «ER-диаграмма» часто не различают, хотя для визуализации ER-моделей могут быть использованы и другие графические нотации, либо визуализация может вообще не применяться (например, использоваться текстовое описание).

Модель была предложена в 1976 году Питером Ченом, им же предложена и самая популярная графическая нотация для модели.

- Концептуальная модель данных — схема наивысшего уровня с минимальным количеством подробностей. Достоинство этого подхода заключается в возможности отобразить общую структуру модели и всю архитектуру системы. Менее масштабные системы могут обойтись и без этой модели. В этом случае можно сразу переходить к логической модели.
- Логическая модель данных содержит более подробную информацию, нежели концептуальная модель. На этом уровне определяются более подробные операционные и транзакционные сущности. Логическая модель не зависит от технологии, в которой она будет применяться.
- Физическая модель данных: на основе каждой логической модели данных можно составить одну или две физических модели. В последних должно присутствовать достаточно технических подробностей для составления и внедрения самой базы данных.

Обращаем ваше внимание на тот факт, что похожие уровни масштаба и детализации встречаются и в других видах схем (например, в диаграммах DFD), однако данная классификация отличается от трехсхемного подхода в разработке ПО, где деление информации осуществляется по несколько иному принципу. Правда, иногда разработчики применяют ER-диаграммы с дополнительными иерархиями, если дизайн базы данных требует больше информационных уровней. К примеру, разработчик может добавить новые группы по принципу расширения вверх (суперклассы) и вниз (подклассы).

• **Только реляционные данные.** Следует четко понимать, что цель ER-диаграмм — показать связи и отношения между элементами, поэтому они отображают только реляционную структуру.

• **Только для структурированных данных.** Данные должны быть

					ПД.09.02.07.XXXX.ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

четко разбиты на поля, столбцы и строки, иначе пользы от ER-диаграммы будет мало. Это касается и частично структурированных данных, так как только некоторые из них будут пригодны для работы.

•**Сложность интеграции с существующей базой данных.** Применение ER-моделей для интеграции с существующей базой данных — непростая задача по причине различия в архитектурах.

Области применения диаграмм

•Проектирование баз данных.

ER-диаграммы применяются для моделирования и проектирования реляционных баз данных, причем как в плане логических и бизнес-правил (логические модели данных), так и в плане внедрения конкретных технологий (физические модели данных). В сфере разработки программного обеспечения ER-диаграмма, как правило, служит первым шагом в определении требований проекта по созданию информационных систем. На дальнейших этапах работы ER-диаграммы также применяются для моделирования конкретных баз данных. Реляционная база данных сопровождается соответствующей реляционной таблицей и при необходимости может быть представлена в этом формате.

•Отладка баз данных.

ER-диаграммы применяются для анализа уже имеющихся баз данных с целью выявить и устранить ошибки в логике или развертывании. Диаграмма позволяет выявить, где именно закрались ошибки.

•Информационные системы для бизнеса.

ER-схемы используются для проектирования и анализа реляционных баз данных, применяемых в бизнес-процессах. Реляционные базы данных могут пригодиться в любом бизнес-процессе, где задействованы данные, разбитые на поля, включая сущности, действия и взаимосвязи. Базы данных помогают оптимизировать процессы, извлекать данные и повышать качество результатов.

•Реорганизация бизнес-процессов (BPR).

ER-диаграммы помогают анализировать базы данных, применяемые при реорганизации бизнес-процессов и моделировании новых баз данных.

•Образование.

Базы данных — широко распространенный в наши дни способ хранения реляционной информации, применяемой в целях образования и для последующего извлечения данных, поэтому ER-диаграммы играют не последнюю роль в планировании подобных структур данных.

•Исследовательская деятельность.

Поскольку исследовательская работа во многом опирается на четко структурированные данные, ER-диаграммы играют ключевую роль в построении оптимальных баз данных для анализа информации.

Диаграммы «сущность-связь» (или ERD) — неотъемлемая составляющая процесса моделирования любых систем, включая простые и сложные базы данных, однако применяемые в них фигуры и способы нотации могут запросто ввести в заблуждение любого.

Это руководство поможет вам стать настоящим экспертом по нотации ER-диаграмм и уверенно взяться за моделирование собственных баз данных!

Концептуальные модели данных дают общее представление о том, что должно входить в состав модели. Концептуальные ER-диаграммы можно брать за основу логических моделей данных. Их также можно использовать для создания отношений общности между разными ER-моделями, положив их в основу интеграции.

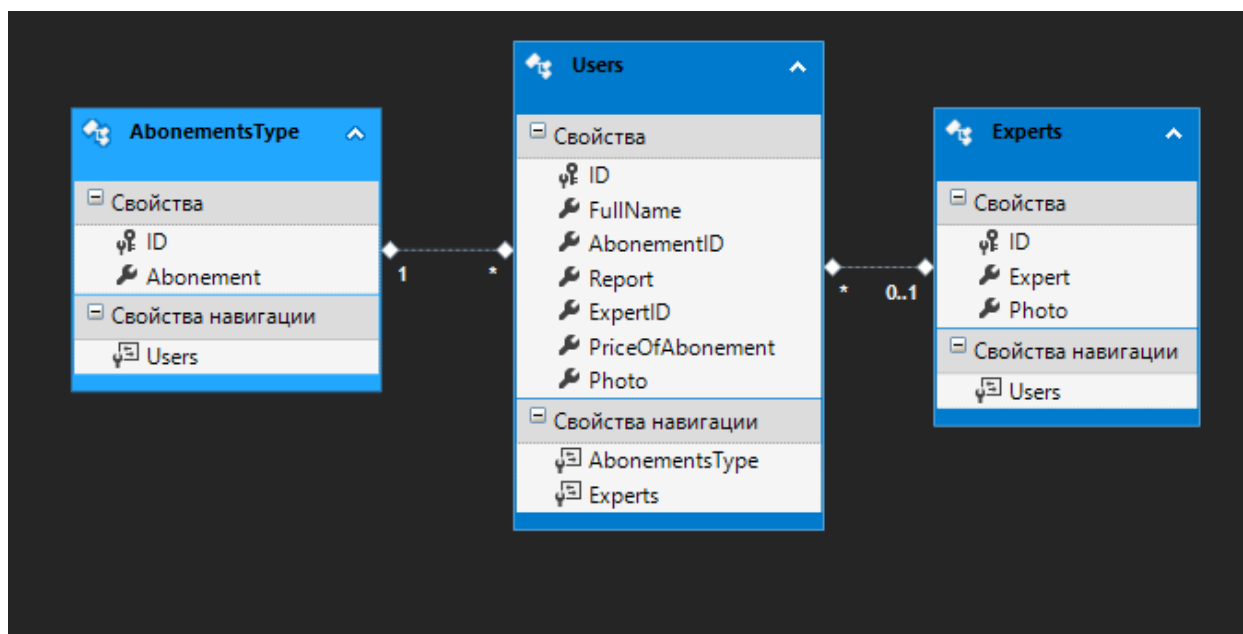


Рисунок 3. ER-диаграмма

В разрабатываемой информационной системе, чтобы не допустить избыточность данных были спроектированы следующие сущности в соответствии с определенными входными данными:

Сущность «Users» содержит информацию о клиентах

Сущность «Experts» содержит информацию о экспертах

Сущность «AbonementsType» содержит информацию об абонементных.

2.2 Логическая модель данных.

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

В данной информационной системе, разработаны связи один ко многим (1:M), которые указаны в таблице 1

Родительская Структурный системный анализ	Дочерняя таблица	Тип связи
Products	Clients	1:M
Products	Images	1:M
Products	ProductSales	1:M
Products	Manufacturers	1:M

Таблица 1. Классификация связей информационной системы (пример)

Родительская таблица	Дочерняя таблица	Тип связи
AbonementsType	Users	1:1
Experts	Users	1:1

Таблица 2. Классификация связей информационной системы

2.3 Физическая структура информационной системы

Для создания информационной системы используются средства Microsoft SQL Server Management Studio 18

Структура таблицы: поля, их типы, и размер представлены в таблице 2.

Имя поля	Тип данных	Размер
ID	int	
FullName	nvarchar	50
AbonementID	int	
Report	nvarchar	MAX
ExpertID	int	
Photo	nvarchar	50
PriceOfAbonement	int	

Таблица 3. «Users»

Имя поля	Тип данных	Размер
ID	int	
Abonement	nvarchar	50

Таблица 4. «AbonementsType»

Имя поля	Тип данных	Размер
ID	int	
Expert	nvarchar	50
Photo	nvarchar	50

Таблица 5. «Experts»

3. Разработка интерфейса

3.1. Разработка формы-меню

Главная страница

Команда	Назначение
Эксперты	Открывается окно с выводом всех экспертов
Абонементы	Открывается окно с выводом всех видов абонементов
Клиенты	Открывается окно с выводом всех клиентов
Выход	Выход из приложения

Таблица 6. Основное меню

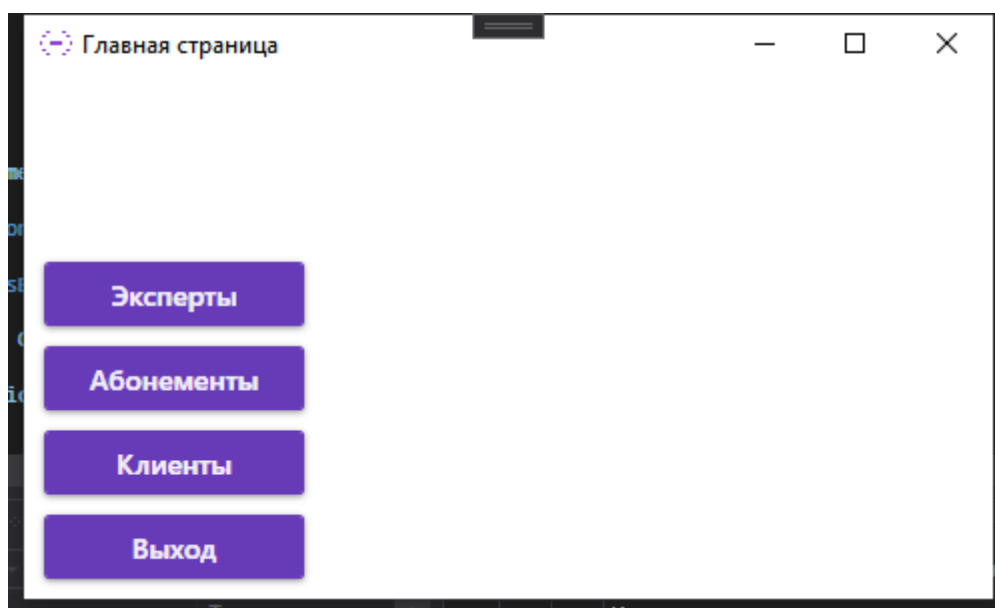


Рисунок 4. Главная страница

Xaml:

```
<Window x:Class="SportClub.StartWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:SportClub"
        mc:Ignorable="d"
        Title="Главная страница"
        WindowStartupLocation="CenterScreen"
        Height="300" Width="500">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="150"/>
            <ColumnDefinition />
        </Grid.ColumnDefinitions>

        <StackPanel Orientation="Vertical" VerticalAlignment="Bottom"
            Grid.Column="0" Margin="5">

            <Button Content="Эксперты" Name="ExpertsButton" Click="ExpertsButton_Click" Margin="5"/>

            <Button Content="Абонементы" Name="AbonementsButton"
            Click="AbonementsButton_Click" Margin="5"/>

            <Button Content="Клиенты" Name="UsersButton" Click="UsersButton_Click" Margin="5"/>

            <Button Content="Выход" Name="ExitButton" Click="ExitButton_Click" Margin="5"/>

        </StackPanel>

    </Grid>
</Window>
```

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		18

Cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
```

```
namespace SportClub
```

```
{
```

```
    public partial class StartWindow : Window
```

```
    {
```

```
        public StartWindow()
```

```
        {
```

```
            InitializeComponent();
```

```
        }
```

```
        private void ExitButton_Click(object sender, RoutedEventArgs e)
```

```
        {
```

```
            Application.Current.Shutdown();
```

```
        }
```

```
        private void UsersButton_Click(object sender, RoutedEventArgs e)
```

```
        {
```

```
            var usersWin = new MainWindow();
```

```
            usersWin.Show();
```

```
            Close();
```

```
        }
```

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

```

e) private void AbonementsButton_Click(object sender, RoutedEventArgs
    {
        var abonementsWin = new AbonementsTypeWindow();
        abonementsWin.Show();

        Close();
    }

private void ExpertsButton_Click(object sender, RoutedEventArgs e)
{
    var expertsWin = new ExpertsWindow();
    expertsWin.Show();

    Close();
}
}
}

```

Окно с экспертами

Команда	Назначение
Добавить эксперта	Открытие окна для ввода данных нового эксперта
Главное окно	Происходит возврат на главную страницу
Выход	Выход из приложения
Назад	Переход к предыдущей странице
Вперёд	Переход к следующей странице

Таблица 7. Окно с выводом экспертов

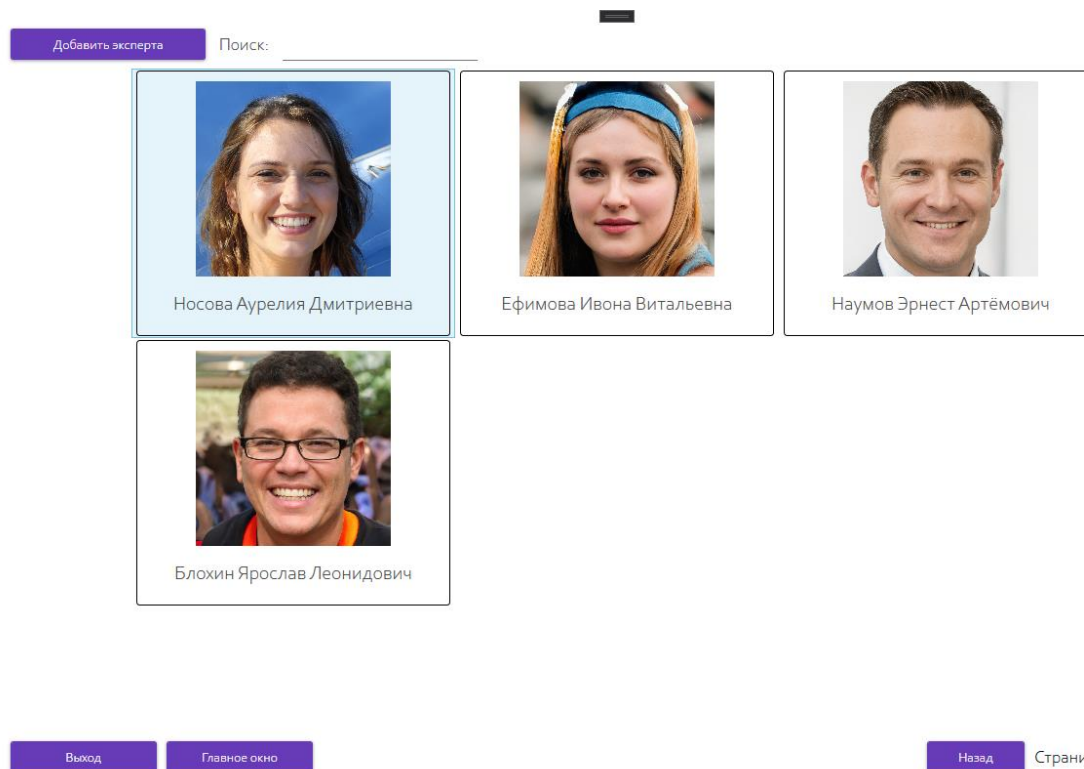


Рисунок 5. Окно с выводом экспертов

Xaml:

```
<Window x:Class="SportClub.ExpertsWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:SportClub"
        mc:Ignorable="d"
        Title="Эксперты" WindowStartupLocation="CenterScreen"
        ResizeMode="NoResize"
        WindowStyle="None"
        WindowState="Maximized"
        FontFamily="Corbel Light"
        FontSize="18">
    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="50"/>
            <RowDefinition />
            <RowDefinition Height="50"/>
        </Grid.RowDefinitions>

        <StackPanel Grid.Row="0" Orientation="Horizontal" Margin="3">
            <Button Content="Добавить эксперта" x:Name="AddButton"
Click="AddWindow_Click" Width="200" Margin="5"/>
            <Label Content="Поиск:" VerticalContentAlignment="Center" Mar-
gin="5"/>
            <TextBox Width="200" x:Name="SearchFilterTextBox"
KeyUp="SearchFilterTextBox_KeyUp" VerticalContentAlignment="Center" Mar-
gin="5"/>
        </StackPanel>

        <StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlign-
ment="Left" Margin="3">
            <Button x:Name="ExitButton" Content="Выход" HorizontalAlign-
ment="Left" Click="ExitButton_Click" Width="150" Margin="5"/>
        </StackPanel>
    </Grid>
</Window>
```

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

```
<Button x:Name="StartWinButton" Content="Главное окно" HorizontalAlignment="Left" Click="StartWinButton_Click" Width="150" Margin="5"/>
```

```
</StackPanel>
```

```
<StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Right" Margin="3">
```

```
<Button x:Name="PrevPage" Content="Назад" Click="PrevPage_Click" Width="100" Margin="5"/>
```

```
<TextBlock Text="{Binding CurrentPage, StringFormat=Страница {0}}" VerticalAlignment="Center" Margin="5"/>
```

```
<Button x:Name="NextPage" Content="Вперёд" Click="NextPage_Click" Width="100" Margin="5"/>
```

```
</StackPanel>
```

```
<ListView
```

```
Grid.Row="1"
```

```
ItemsSource="{Binding ExpertsList}"
```

```
x:Name="ExpertsListView"
```

```
ScrollViewer.HorizontalScrollBarVisibility="Disabled" Cursor="Hand">
```

```
<ListView.ItemContainerStyle>
```

```
<Style TargetType="ListViewItem">
```

```
<Setter Property="HorizontalContentAlignment" Value="Left"/>
```

```
</Style>
```

```
</ListView.ItemContainerStyle>
```

```
<ListView.ContextMenu>
```

```
<ContextMenu>
```

```
<MenuItem Header="Редактировать" x:Name="EditItem" Click="EditItem_Click"/>
```

```
<MenuItem Header="Удалить" x:Name="DeleteItem" Click="DeleteItem_Click"/>
```

```
</ContextMenu>
```

```
</ListView.ContextMenu>
```

```
<ListView.ItemsPanel>
```

					ПД.09.02.07.XXXX.ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <ItemsPanelTemplate>
            <WrapPanel Orientation="Horizontal" MaxWidth="1100" HorizontalAlignment="Center" />
        </ItemsPanelTemplate>
    </ListView.ItemsPanel>

    <ListView.ItemTemplate>
        <DataTemplate>
            <Border BorderThickness="1" BorderBrush="Black" CornerRadius="3">

                <Grid Margin="10">
                    <StackPanel Width="300" Height="250">
                        <Image Source="{Binding Path=PhotoView}" HorizontalAlignment="Center" Width="250" Height="200" Margin="0 0 0 15"/>
                        <TextBlock FontSize="20" HorizontalAlignment="Center" Text="{Binding Expert}"/>
                    </StackPanel>
                </Grid>
            </Border>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
</Grid>
</Window>

```

Cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
```

```
namespace SportClub
```

```
{
```

```
    public partial class Experts
```

```
    {
```

```
        public Uri PhotoView
```

```
        {
```

```
            get
```

```
            {
```

```
                var PhotoName = Environment.CurrentDirectory + Photo ?? "";
```

```
                return System.IO.File.Exists(PhotoName) ? new Uri(PhotoName) :
```

```
new Uri("pack://application:,,,/img/NoPhoto.png");
```

```
            }
```

```
        }
```

```
    }
```

```
    public partial class ExpertsWindow : Window, INotifyPropertyChanged
```

```
    {
```

```
        public event PropertyChangedEventHandler PropertyChanged;
```

```
        private IEnumerable<Experts> _ExpertsList;
```

```
        public IEnumerable<Experts> ExpertsList
```

					ПД.09.02.07.XXXX.ПЗ	Лист
						25
Изм.	Лист	№ докум.	Подпись	Дата		


```

    {
        get
        {
            var Result = _ExpertsList;

            if (SearchFilter != "")
                Result = Result.Where(ai => ai.Expert.IndexOf(SearchFilter,
StringComparison.OrdinalIgnoreCase) >= 0);

            return Result.Skip((CurrentPage - 1) * 6).Take(6);
        }
        set
        {
            _ExpertsList = value;
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs("Expert-
sList"));
            }
        }
    }

    public ExpertsWindow()
    {
        InitializeComponent();
        DataContext = this;
        ExpertsList = Core.DB.Experts.ToArray();
    }

    private int _CurrentPage;
    public int CurrentPage
    {
        get
        {
            return _CurrentPage;
        }
        set
    }

```

```

    {
        if (value > 0)
        {
            if ((_ExpertsList.Count() % 6) == 0)
            {
                if (value <= _ExpertsList.Count() / 6)
                {
                    _CurrentPage = value;
                    Invalidate();
                }
            }
            else
            {
                if (value <= (_ExpertsList.Count() / 6) + 1)
                {
                    _CurrentPage = value;
                    Invalidate();
                }
            }
        }
    }
}

```

```
private int _ExpertsListValue = 0;
```

```
public int ExpertsListValue
```

```

{
    get
    {
        return _ExpertsListValue;
    }
    set
    {
        _ExpertsListValue = value;
        if (PropertyChanged != null)
        {
            PropertyChanged(this, new PropertyChangedEventArgs("Expert-
sList"));

```

```

        }
    }
}

private string _SearchFilter = "";
public string SearchFilter
{
    get
    {
        return _SearchFilter;
    }
    set
    {
        _SearchFilter = value;
        if (PropertyChanged != null)
        {
            PropertyChanged(this, new PropertyChangedEventArgs("ExpertsList"));
        }
    }
}

private void Invalidate()
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("ExpertsList"));
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("CurrentPage"));
}

private void AddWindow_Click(object sender, RoutedEventArgs e)
{
    var addWin = new AddEditExpertWindow(new Experts());
    if (addWin.ShowDialog() == true)
    {
        ExpertsList = Core.DB.Experts.ToArray();
    }
}

```

```

    }

    private void EditItem_Click(object sender, RoutedEventArgs e)
    {
        var editWin = new AddEditExpertWindow(ExpertsListView.SelectedItem as Experts);
        if (editWin.ShowDialog() == true)
        {
            ExpertsList = Core.DB.Experts.ToArray();
        }
    }

    private void DeleteItem_Click(object sender, RoutedEventArgs e)
    {
        var deleteExpert = ExpertsListView.SelectedItem as Experts;
        try
        {
            Core.DB.Experts.Remove(deleteExpert);
            Core.DB.SaveChanges();

            MessageBox.Show($"Удалено");

            ExpertsList = Core.DB.Experts.ToArray();

            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs("ExpertsList"));
            }
        }
        catch { }
    }

    private void ExitButton_Click(object sender, RoutedEventArgs e)
    {
        Application.Current.Shutdown();
    }

```

e)

```
private void SearchFilterTextBox_KeyUp(object sender, KeyEventArgs
{
    SearchFilter = SearchFilterTextBox.Text;
    Invalidate();
}
private void PrevPage_Click(object sender, RoutedEventArgs e)
{
    CurrentPage--;
}

private void NextPage_Click(object sender, RoutedEventArgs e)
{
    CurrentPage++;
}

private void StartWinButton_Click(object sender, RoutedEventArgs e)
{
    var startWin = new StartWindow();
    startWin.Show();

    Close();
}
}
```

Окно с добавлением и редактированием данных эксперта

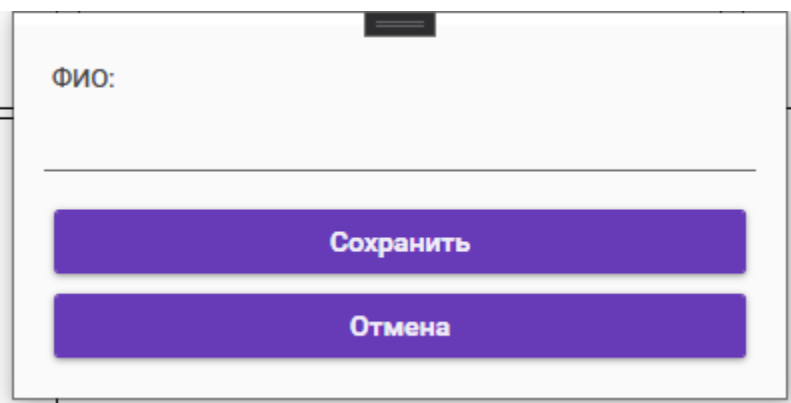


Рисунок 6. Окно добавления данных эксперта

Команда	Назначение
Сохранить	Происходит сохранение данных
Назад	Происходит закрытие окна

Таблица 8. Окно редактирования данных эксперта

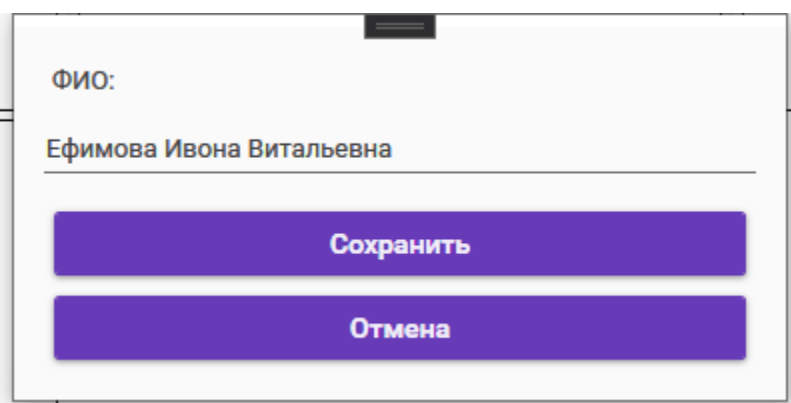


Рисунок 7. Окно редактирования данных эксперта

Команда	Назначение
Сохранить	Происходит сохранение данных
Назад	Происходит закрытие окна

Таблица 9. Окно добавления данных эксперта

Xaml:

```
<Window x:Class="SportClub.AddEditExpertWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:SportClub"
        mc:Ignorable="d"
        xmlns:materialDesign="http://materialdesigninx-
aml.net/winfx/xaml/themes"
        TextElement.Foreground="{DynamicResource MaterialDesignBody}"
        TextElement.FontWeight="Regular"
        TextElement.FontSize="13"
        TextOptions.TextFormattingMode="Ideal"
        TextOptions.TextRenderingMode="Auto"
        Background="{DynamicResource MaterialDesignPaper}"
        FontFamily="{DynamicResource MaterialDesignFont}"

        Height="200" MinHeight="200"
        Width="400" MinWidth="400"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None">
    <Grid Margin="10">

        <StackPanel Orientation="Vertical">
            <Label Content="ФИО:" Margin="5"/>
            <TextBox Text="{Binding AllExperts.Expert}" Margin="5"/>
        </StackPanel>
        <StackPanel Margin="5" VerticalAlignment="Bottom">
            <Button x:Name="SaveButton" Content="Сохранить" Click="Save-
Button_Click" Margin="5"/>
            <Button x:Name="CloseButton" Content="Отмена" Click="CloseBut-
ton_Click" Margin="5"/>
        </StackPanel>
    </Grid>
</Window>
```

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		32

Cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
```

namespace SportClub

```
{
    public partial class AddEditExpertWindow : Window
    {
        public Experts AllExperts { get; set; }
        public AddEditExpertWindow(Experts experts)
        {
            InitializeComponent();
            DataContext = this;
            AllExperts = experts;
        }

        private void SaveButton_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                if (AllExperts.Expert == null)
                    throw new Exception("Не введено ФИО эксперта");

                if (AllExperts.ID == 0)
                    Core.DB.Experts.Add(AllExperts);
            }
            catch { }
        }
    }
}
```



```

        Core.DB.SaveChanges();

        DialogResult = true;

        MessageBox.Show($"Сохранено");
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка: {ex.Message}");
    }
}

private void CloseButton_Click(object sender, RoutedEventArgs e)
{
    Close();
}
}
}

```

Окно с абонементом

Команда	Назначение
Добавить абонемент	Открытие окна для ввода данных нового абонемента
Главное окно	Происходит возврат на главную страницу
Выход	Выход из приложения
Назад	Переход к предыдущей странице
Вперёд	Переход к следующей странице

Таблица 10. Окно с выводом абонементов

Рисунок 8. Окно с выводом абонементов

Xaml:

```
<Window x:Class="SportClub.AbonementsTypeWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:local="clr-namespace:SportClub"
  mc:Ignorable="d"
  WindowStartupLocation="CenterScreen"
  ResizeMode="NoResize"
  WindowStyle="None"
  WindowState="Maximized"
  FontFamily="Corbel Light"
  FontSize="18">
  <Grid Margin="10">
    <Grid.RowDefinitions>
      <RowDefinition Height="50"/>
      <RowDefinition />
      <RowDefinition Height="50"/>
    </Grid.RowDefinitions>

    <StackPanel Grid.Row="0" Orientation="Horizontal" Margin="3">
      <Button Content="Добавить абонемент" x:Name="AddButton"
Click="AddWindow_Click" Width="200" Margin="5"/>
      <Label Content="Поиск:" VerticalContentAlignment="Center" Mar-
gin="5"/>
      <TextBox Width="200" x:Name="SearchFilterTextBox"
KeyUp="SearchFilterTextBox_KeyUp" VerticalContentAlignment="Center" Mar-
gin="5"/>
    </StackPanel>

    <StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlign-
ment="Left" Margin="3">
      <Button x:Name="ExitButton" Content="Выход" HorizontalAlign-
ment="Left" Click="ExitButton_Click" Width="150" Margin="5"/>
    </StackPanel>
  </Grid>
</Window>
```

```
<Button x:Name="StartWinButton" Content="Главное окно" HorizontalAlignment="Left" Click="StartWinButton_Click" Width="150" Margin="5"/>
```

```
</StackPanel>
```

```
<StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Right" Margin="3">
```

```
<Button x:Name="PrevPage" Content="Назад" Click="PrevPage_Click" Width="100" Margin="5"/>
```

```
<TextBlock Text="{Binding CurrentPage, StringFormat=Страница {0}}" VerticalAlignment="Center" Margin="5"/>
```

```
<Button x:Name="NextPage" Content="Вперёд" Click="NextPage_Click" Width="100" Margin="5"/>
```

```
</StackPanel>
```

```
<ListView
```

```
Grid.Row="1"
```

```
ItemsSource="{Binding AbonementsTypeList}"
```

```
x:Name="AbonementsTypeListView"
```

```
ScrollViewer.HorizontalScrollBarVisibility="Disabled" Cursor="Hand">
```

```
<ListView.ItemContainerStyle>
```

```
<Style TargetType="ListViewItem">
```

```
<Setter Property="HorizontalContentAlignment" Value="Left"/>
```

```
</Style>
```

```
</ListView.ItemContainerStyle>
```

```
<ListView.ContextMenu>
```

```
<ContextMenu>
```

```
<MenuItem Header="Редактировать" x:Name="EditItem" Click="EditItem_Click"/>
```

```
<MenuItem Header="Удалить" x:Name="DeleteItem" Click="DeleteItem_Click"/>
```

```
</ContextMenu>
```

```
</ListView.ContextMenu>
```

```
<ListView.ItemsPanel>
```

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		37

```

        <ItemsPanelTemplate>
            <WrapPanel Orientation="Horizontal" MaxWidth="1100" HorizontalAlignment="Center" />
        </ItemsPanelTemplate>
    </ListView.ItemsPanel>

    <ListView.ItemTemplate>
        <DataTemplate>
            <Border BorderThickness="1" BorderBrush="Black" CornerRadius="3">

                <Grid Margin="10">
                    <StackPanel Width="300" Height="100">
                        <TextBlock
                            FontSize="20"
                            HorizontalAlignment="Center"
                            Text="{Binding Abonement}"/>
                    </StackPanel>
                </Grid>
            </Border>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
</Grid>
</Window>

```

Cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
```

```
namespace SportClub
{
    public partial class AbonementsTypeWindow : Window, INotifyProperty-
Changed
    {
        public event PropertyChangedEventHandler PropertyChanged;

        private IEnumerable<AbonementsType> _AbonementsTypeList;

        public IEnumerable<AbonementsType> AbonementsTypeList
        {
            get
            {
                var Result = _AbonementsTypeList;

                if (SearchFilter != "")
                    Result = Result.Where(ai => ai.Abonement.IndexOf(SearchFilter,
StringComparison.OrdinalIgnoreCase) >= 0);

                return Result.Skip((CurrentPage - 1) * 12).Take(12);
            }
        }
    }
}
```

					ПД.09.02.07.XXXX.ПЗ	Лист
						39
Изм.	Лист	№ докум.	Подпись	Дата		

```

        set
        {
            _AbonementsTypeList = value;
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs("AbonementsTypeList"));
            }
        }
    }

    public AbonementsTypeWindow()
    {
        InitializeComponent();
        DataContext = this;
        AbonementsTypeList = Core.DB.AbonementsType.ToArray();
    }

    private int _CurrentPage;
    public int CurrentPage
    {
        get
        {
            return _CurrentPage;
        }
        set
        {
            if (value > 0)
            {
                if ((_AbonementsTypeList.Count() % 12) == 0)
                {
                    if (value <= _AbonementsTypeList.Count() / 12)
                    {
                        _CurrentPage = value;
                        Invalidate();
                    }
                }
            }
        }
    }

```

```

        else
        {
            if (value <= (_AbonementsTypeList.Count() / 12) + 1)
            {
                _CurrentPage = value;
                Invalidate();
            }
        }
    }
}

private int _AbonementsTypeListValue = 0;
public int AbonementsTypeListValue
{
    get
    {
        return _AbonementsTypeListValue;
    }
    set
    {
        _AbonementsTypeListValue = value;
        if (PropertyChanged != null)
        {
            PropertyChanged(this, new PropertyChangedEventArgs("AbonementsTypeList"));
        }
    }
}

private string _SearchFilter = "";
public string SearchFilter
{
    get
    {
        return _SearchFilter;
    }
}

```



```

        set
        {
            _SearchFilter = value;
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs("AbonementsTypeList"));
            }
        }

    private void Invalidate()
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("AbonementsTypeList"));
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("CurrentPage"));
    }

    private void AddWindow_Click(object sender, RoutedEventArgs e)
    {
        var addWin = new AddEditATWindow(new AbonementsType());
        if (addWin.ShowDialog() == true)
        {
            AbonementsTypeList = Core.DB.AbonementsType.ToArray();
        }
    }

    private void EditItem_Click(object sender, RoutedEventArgs e)
    {
        var editWin = new AddEditATWindow(AbonementsTypeListView.SelectedItem as AbonementsType);
        if (editWin.ShowDialog() == true)
        {
            AbonementsTypeList = Core.DB.AbonementsType.ToArray();
        }
    }

```

```

private void DeleteItem_Click(object sender, RoutedEventArgs e)
{
    var deleteAbonement = AbonementsTypeListView.SelectedItem as
AbonementsType;
    try
    {
        Core.DB.AbonementsType.Remove(deleteAbonement);
        Core.DB.SaveChanges();

        MessageBox.Show($"Удалено");

        AbonementsTypeList = Core.DB.AbonementsType.ToArray();

        if (PropertyChanged != null)
        {
            PropertyChanged(this, new PropertyChangedEventArgs("AbonementsTypeList"));
        }
    }
    catch { }
}

private void ExitButton_Click(object sender, RoutedEventArgs e)
{
    Application.Current.Shutdown();
}

private void SearchFilterTextBox_KeyUp(object sender, KeyEventArgs
e)
{
    SearchFilter = SearchFilterTextBox.Text;
    Invalidate();
}

private void PrevPage_Click(object sender, RoutedEventArgs e)
{
    CurrentPage--;
}

```

```

    }

    private void NextPage_Click(object sender, RoutedEventArgs e)
    {
        CurrentPage++;
    }

    private void StartWinButton_Click(object sender, RoutedEventArgs e)
    {
        var startWin = new StartWindow();
        startWin.Show();

        Close();
    }
}

```

					ПД.09.02.07.XXXX.ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

Окно с добавлением и редактированием данных абонемента

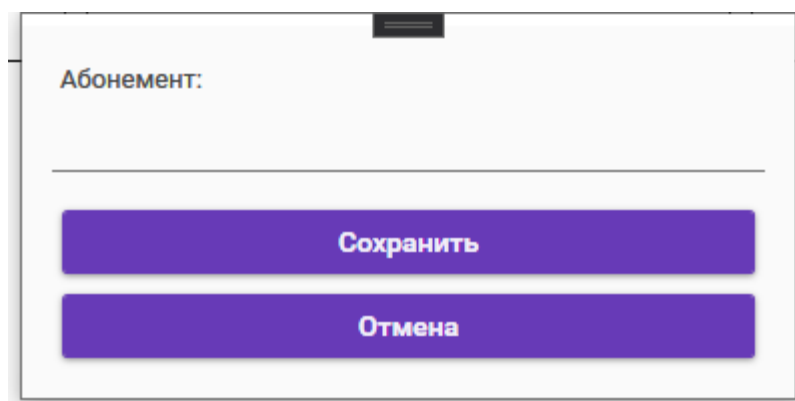


Рисунок 9. Окно добавления данных абонемента

Команда	Назначение
Сохранить	Происходит сохранение данных
Назад	Происходит закрытие окна

Таблица 11. Окно редактирования данных абонемента

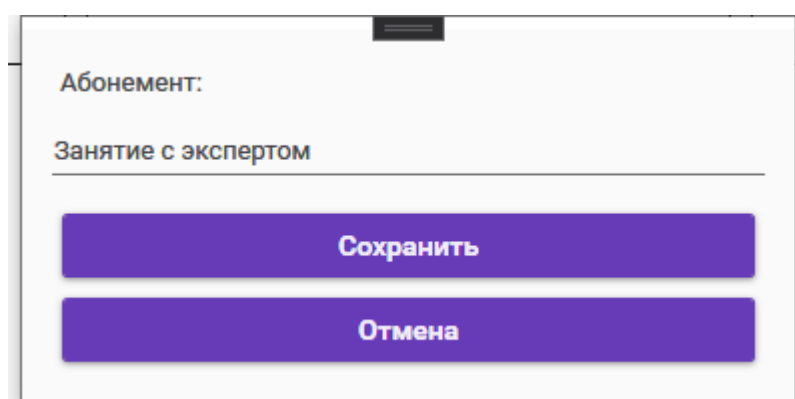


Рисунок 10. Окно редактирования данных абонемента

Команда	Назначение
Сохранить	Происходит сохранение данных
Назад	Происходит закрытие окна

Таблица 12. Окно добавления данных абонемента

Xaml:

```
<Window x:Class="SportClub.AddEditATWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:SportClub"
    mc:Ignorable="d"
    xmlns:materialDesign="http://materialdesigninx-
aml.net/winfx/xaml/themes"
    TextElement.Foreground="{DynamicResource MaterialDesignBody}"
    TextElement.FontWeight="Regular"
    TextElement.FontSize="13"
    TextOptions.TextFormattingMode="Ideal"
    TextOptions.TextRenderingMode="Auto"
    Background="{DynamicResource MaterialDesignPaper}"
    FontFamily="{DynamicResource MaterialDesignFont}"

    Height="200" MinHeight="200"
    Width="400" MinWidth="400"
    WindowStartupLocation="CenterScreen"
    WindowStyle="None">
<Grid Margin="10">
    <StackPanel Orientation="Vertical">
        <Label Content="Абонемент:" Margin="5"/>
        <TextBox Text="{Binding AllAbonementsType.Abonement}" Mar-
gin="5"/>
    </StackPanel>
    <StackPanel Margin="5" VerticalAlignment="Bottom">
        <Button x:Name="SaveButton" Content="Сохранить" Click="Save-
Button_Click" Margin="5"/>
        <Button x:Name="CloseButton" Content="Отмена" Click="CloseBut-
ton_Click" Margin="5"/>
    </StackPanel>
</Grid>
</Window>
```

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		46

```

Cs:
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace SportClub
{
    public partial class AddEditATWindow : Window
    {
        public AbonementsType AllAbonementsType { get; set; }
        public AddEditATWindow(AbonementsType abonementsType)
        {
            InitializeComponent();
            DataContext = this;
            AllAbonementsType = abonementsType;
        }

        private void SaveButton_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                if (AllAbonementsType.Abonement == null)
                    throw new Exception("Не выбран тип абонемента");

                if (AllAbonementsType.ID == 0)
                    Core.DB.AbonementsType.Add(AllAbonementsType);
            }
            catch { }
        }
    }
}

```

```

        Core.DB.SaveChanges();

        DialogResult = true;

        MessageBox.Show($"Сохранено");
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка: {ex.Message}");
    }
}

private void CloseButton_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}
}
}

```

					ПД.09.02.07.XXXX.ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

Окно с клиентами

Команда	Назначение
Добавить клиента	Открытие окна для ввода данных нового клиента
Главное окно	Происходит возврат на главную страницу
Выход	Выход из приложения
Назад	Переход к предыдущей странице
Вперёд	Переход к следующей странице


Таблица 13. Окно с выводом клиентов

Добавить клиента


Все типы абонементов

Сортировка по цене: ☒ по возрастанию ☐ по убыванию


Поиск:




Коновалов Адольф Максимович
Тренажорный зал
Носова Аурелия Дмитриевна
Цена: 1000




Цветкова Милолика Кимовна
Занятие с экспертом
Наумов Эрнест Артёмович
Цена: 1500




Цветкова Наталья Владимировна
Бокс
Наумов Эрнест Артёмович
Цена: 1500



Рыбаков Сергей Ростиславович
Йога
Ефимова Илона Витальевна
Цена: 2000



Носова Евгения Сергеевна
Йога
Ефимова Илона Витальевна
Цена: 2000



Желткевич Диана Сергеевна
Тренажорный зал
Наумов Эрнест Артёмович
Цена: 3000

Выход

Главное окно

Назад

Страница 0

Вперёд

Рисунок 11. Окно с выводом клиентов

Xaml:

```
<Window x:Class="SportClub.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:SportClub"
        mc:Ignorable="d"
        Title="Клиенты"
        WindowStartupLocation="CenterScreen"
        ResizeMode="NoResize"
        WindowStyle="None"
        WindowState="Maximized"
        FontFamily="Corbel Light"
        FontSize="18">
    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="50"/>
            <RowDefinition />
            <RowDefinition Height="50"/>
        </Grid.RowDefinitions>

        <StackPanel Grid.Row="0" Orientation="Horizontal" Margin="3">
            <Button Content="Добавить клиента" x:Name="AddButton"
Click="AddWindow_Click" Width="200" Margin="5"/>
            <ComboBox
                SelectedIndex="0"
                VerticalAlignment="Center"
                Width="250"
                x:Name="AbonementsTypeListComboBox"
                ItemsSource="{Binding AbonementsTypeList}"
                SelectionChanged="AbonementsTypeList_SelectionChanged"
                Margin="5">
                <ComboBox.ItemTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding Abonement}"/>
                    </DataTemplate>
                </ComboBox.ItemTemplate>
            </ComboBox>
        </StackPanel>
    </Grid>
</Window>
```

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		50

```

        </DataTemplate>
    </ComboBox.ItemTemplate>
</ComboBox>
    <Label Content="Сортировка по цене:" VerticalContentAlign-
ment="Center" Margin="5"/>
    <RadioButton GroupName="SportClub" Tag="1" Content="по воз-
растанию" IsChecked="True" Checked="RadioButton_Checked" VerticalCon-
tentAlignment="Center" VerticalAlignment="Center" Margin="5"/>
    <RadioButton Content="по убыванию" VerticalContentAlign-
ment="Center" VerticalAlignment="Center" Tag="2" GroupName="SportClub"
Checked="RadioButton_Checked" Margin="5"/>
    <Label Content="Поиск:" VerticalContentAlignment="Center" Mar-
gin="5"/>
    <TextBox Width="200" x:Name="SearchFilterTextBox"
KeyUp="SearchFilterTextBox_KeyUp" VerticalContentAlignment="Center" Mar-
gin="5"/>

</StackPanel>

    <StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlign-
ment="Left" Margin="3">
        <Button x:Name="ExitButton" Content="Выход" HorizontalAlign-
ment="Left" Click="ExitButton_Click" Width="150" Margin="5"/>
        <Button x:Name="StartWinButton" Content="Главное окно" Hori-
zontalAlignment="Left" Click="StartWinButton_Click" Width="150" Mar-
gin="5"/>
    </StackPanel>

    <StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlign-
ment="Right" Margin="3">
        <Button x:Name="PrevPage" Content="Назад"
Click="PrevPage_Click" Width="100" Margin="5"/>
        <TextBlock Text="{Binding CurrentPage, StringFormat=Страница
{0}}" VerticalAlignment="Center" Margin="5"/>
        <Button x:Name="NextPage" Content="Вперёд"
Click="NextPage_Click" Width="100" Margin="5"/>
    </StackPanel>

```

```

<ListView
  Grid.Row="1"
  ItemsSource="{ Binding UsersList}"
  x:Name="UsersListView"
  ScrollViewer.HorizontalScrollBarVisibility="Disabled"      Cur-
sor="Hand">

  <ListView.ItemContainerStyle>
    <Style TargetType="ListViewItem">
      <Setter Property="HorizontalContentAlignment" Value="Left"/>
    </Style>
  </ListView.ItemContainerStyle>

  <ListView.ContextMenu>
    <ContextMenu>
      <MenuItem Header="Редактировать" x:Name="EditItem"
Click="EditItem_Click"/>
      <MenuItem Header="Отчёт" x:Name="ReportItem" Click="Re-
portItem_Click"/>
      <MenuItem Header="Удалить" x:Name="DeleteItem"
Click="DeleteItem_Click"/>
    </ContextMenu>
  </ListView.ContextMenu>

  <ListView.ItemsPanel>
    <ItemsPanelTemplate>
      <WrapPanel Orientation="Horizontal" MaxWidth="1100" Hori-
zontalAlignment="Center" />
    </ItemsPanelTemplate>
  </ListView.ItemsPanel>

  <ListView.ItemTemplate>
    <DataTemplate>
      <Border BorderThickness="1" BorderBrush="Black" CornerRa-
dius="3">

        <Grid Margin="10">
          <StackPanel Width="300" Height="300">

```

```

<Image
    Source="{ Binding Path=PhotoView }"
    HorizontalAlignment="Center"
    Width="250"
    Height="200"/>
<TextBlock
    FontSize="20"
    HorizontalAlignment="Center"
    Text="{ Binding FullName }"/>
<TextBlock
    FontSize="20"
    HorizontalAlignment="Center"
    Text="{ Binding AbonementsType.Abonement }"/>
<TextBlock
    FontSize="20"

    HorizontalAlignment="Center"    Text="{ Binding
    Experts.Expert }"/>
<TextBlock
    FontSize="20"
    HorizontalAlignment="Center"
    Text="{ Binding PriceOfAbonement, StringFormat={ }Цена: {0} }"/>
</StackPanel>
</Grid>
</Border>
</DataTemplate>
</ListView.ItemTemplate>
</ListView>
</Grid>
</Window>

```

Cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
```

```
namespace SportClub
```

```
{
```

```
    public partial class Users
```

```
    {
```

```
        public Uri PhotoView
```

```
        {
```

```
            get
```

```
            {
```

```
                var PhotoName = Environment.CurrentDirectory + Photo ?? "";
```

```
                return System.IO.File.Exists(PhotoName) ? new Uri(PhotoName) :
```

```
new Uri("pack://application:,,,/img/NoPhoto.png");
```

```
            }
```

```
        }
```

```
    }
```

```
    public partial class MainWindow : Window, INotifyPropertyChanged
```

```
    {
```

```
        public event PropertyChangedEventHandler PropertyChanged;
```

					ПД.09.02.07.XXXX.ПЗ	Лист
						54
Изм.	Лист	№ докум.	Подпись	Дата		

```

private IEnumerable<Users> _UsersList;

public IEnumerable<Users> UsersList
{
    get
    {
        var Result = _UsersList;

        if (_AbonementsTypeListValue > 0)
            Result = Result.Where(ai => ai.AbonementID == _Abonement-
sTypeListValue);

        if (SearchFilter != "")
            Result = Result.Where(ai => ai.FullName.IndexOf(SearchFilter,
StringComparison.OrdinalIgnoreCase) >= 0);

        if (SortList) Result = Result.OrderBy(ai => ai.PriceOfAbonement);
        else Result = Result.OrderByDescending(ai => ai.PriceOfAbo-
nement);

        return Result.Skip((CurrentPage - 1) * 6).Take(6);
    }
    set
    {
        _UsersList = value;
        if(PropertyChanged != null)
        {
            PropertyChanged(this, new PropertyChangedEventArgs("Users-
List"));
        }
    }
}

public List<AbonementsType> AbonementsTypeList { get; set; }

public MainWindow()
{

```

```

InitializeComponent();
DataContext = this;
UsersList = Core.DB.Users.ToArray();
AbonementsTypeList = Core.DB.AbonementsType.ToList();
AbonementsTypeList.Insert(0, new AbonementsType { Abonement =
"Все типы абонементов" });
}

```

```

private int _CurrentPage;
public int CurrentPage
{
    get
    {
        return _CurrentPage;
    }
    set
    {
        if (value > 0)
        {
            if ((_UsersList.Count() % 6) == 0)
            {
                if (value <= _UsersList.Count() / 6)
                {
                    _CurrentPage = value;
                    Invalidate();
                }
            }
            else
            {
                if (value <= (_UsersList.Count() / 6) + 1 )
                {
                    _CurrentPage = value;
                    Invalidate();
                }
            }
        }
    }
}

```

```

    }

    private int _AbonementsTypeListValue = 0;
    public int AbonementsTypeListValue
    {
        get
        {
            return _AbonementsTypeListValue;
        }
        set
        {
            _AbonementsTypeListValue = value;
            if(PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs("Users-
List"));
            }
        }
    }

    private bool _SortList = true;
    public bool SortList
    {
        get
        {
            return _SortList;
        }
        set
        {
            _SortList = value;
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs("Users-
List"));
            }
        }
    }

```



```

private string _SearchFilter = "";
public string SearchFilter
{
    get
    {
        return _SearchFilter;
    }
    set
    {
        _SearchFilter = value;
        if (PropertyChanged != null)
        {
            PropertyChanged(this, new PropertyChangedEventArgs("Users-
List"));
        }
    }
}

private void Invalidate()
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("UsersList"));
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("CurrentPage"));
}

private void AddWindow_Click(object sender, RoutedEventArgs e)
{
    var addWin = new AddEditWindow(new Users());
    if(addWin.ShowDialog() == true)
    {
        UsersList = Core.DB.Users.ToArray();
    }
}

private void EditItem_Click(object sender, RoutedEventArgs e)

```

```

        {
            var editWin = new AddEditWindow(UsersListView.SelectedItem as
Users);
            if(editWin.ShowDialog() == true)
            {
                UsersList = Core.DB.Users.ToArray();
            }
        }

private void DeleteItem_Click(object sender, RoutedEventArgs e)
{
    var deleteUser = UsersListView.SelectedItem as Users;
    try
    {
        Core.DB.Users.Remove(deleteUser);
        Core.DB.SaveChanges();

        MessageBox.Show($"Удалено");

        UsersList = Core.DB.Users.ToArray();

        if (PropertyChanged != null)
        {
            PropertyChanged(this, new PropertyChangedEventArgs("ProductList"));
        }
    }
    catch { }
}

private void ExitButton_Click(object sender, RoutedEventArgs e)
{
    Application.Current.Shutdown();
}

private void RadioButton_Checked(object sender, RoutedEventArgs e)
{
    SortList = (sender as RadioButton).Tag.ToString() == "1";
}

```

```

    }
    private void SearchFilterTextBox_KeyUp(object sender, KeyEventArgs e)
    {
        SearchFilter = SearchFilterTextBox.Text;
        Invalidate();
    }
    private void PrevPage_Click(object sender, RoutedEventArgs e)
    {
        CurrentPage--;
    }
    private void NextPage_Click(object sender, RoutedEventArgs e)
    {
        CurrentPage++;
    }
    private void AbonementsTypeList_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        AbonementsTypeListValue = (AbonementsTypeListComboBox.SelectedItem as AbonementsType).ID;
    }
    private void ReportItem_Click(object sender, RoutedEventArgs e)
    {
        var reportWin = new ReportWindow(UsersListView.SelectedItem as Users);
        if (reportWin.ShowDialog() == true)
        {
            UsersList = Core.DB.Users.ToArray();
        }
    }
    private void StartWinButton_Click(object sender, RoutedEventArgs e)
    {
        var startWin = new StartWindow();
        startWin.Show();
        Close();
    }
}
}

```

Окно с добавлением и редактированием данных клиента

Рисунок 12. Окно добавления данных клиента

Команда	Назначение
Сохранить	Происходит сохранение данных
Назад	Происходит закрытие окна

Таблица 14. Окно добавления данных клиента

ФИО

Цветкова Милолика Кимовна

Абонемент:

Занятие с экспертом

Эксперт:

Наумов Эрнест Артёмович

Отчёт о выполненном занятии:

Цена абонемента:

1500

Сохранить

Отмена

Рисунок 13. Окно редактирования данных клиента

Команда	Назначение
Сохранить	Происходит сохранение данных
Назад	Происходит закрытие окна

Таблица 15. Окно редактирования данных клиента

Xaml:

```
<Window x:Class="SportClub.AddEditWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:SportClub"
        mc:Ignorable="d"

        xmlns:materialDesign="http://materialdesigninx-
aml.net/winfx/xaml/themes"

        TextElement.Foreground="{DynamicResource MaterialDesignBody}"
        TextElement.FontWeight="Regular"
        TextElement.FontSize="13"
        TextOptions.TextFormattingMode="Ideal"
        TextOptions.TextRenderingMode="Auto"
        Background="{DynamicResource MaterialDesignPaper}"
        FontFamily="{DynamicResource MaterialDesignFont}"

        WindowStartupLocation="CenterScreen"
        WindowStyle="None">
<Grid Margin="10">
    <StackPanel>
        <Label Content="ФИО"/>
        <TextBox
            Text="{Binding AllUsers.FullName}"/>
        <Label
            Content="Абонемент:"/>
        <ComboBox
            ItemsSource="{Binding AbonementsTypeSelect}"
            SelectedItem="{Binding AllUsers.AbonementsType}">
            <ComboBox.ItemTemplate>
                <DataTemplate>
                    <TextBlock Text="{Binding Abonement}"/>
                </DataTemplate>
            </ComboBox.ItemTemplate>
        </ComboBox>
    </StackPanel>
</Grid>
```

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		63

```

</ComboBox>
<Label Content="Эксперт:"/>
<ComboBox
    ItemsSource="{Binding ExpertsSelect}"
    SelectedItem="{Binding AllUsers.Experts}">
    <ComboBox.ItemTemplate>
        <DataTemplate>
            <TextBlock Text="{Binding Expert}"/>
        </DataTemplate>
    </ComboBox.ItemTemplate>
</ComboBox>
<Label
    Content="Отчёт о выполненном занятии:"/>
<TextBox
    Text="{Binding AllUsers.Report}" Height="100"/>
<Label
    Content="Цена абонемента:"/>
<TextBox
    Text="{Binding AllUsers.PriceOfAbonement}"/>
</StackPanel>
<StackPanel
    Margin="5"
    VerticalAlignment="Bottom">
    <Button
        x:Name="SaveButton"
        Content="Сохранить"
        Click="SaveButton_Click"
        Margin="5"/>
    <Button
        x:Name="CloseButton"
        Content="Отмена"
        Click="CloseButton_Click"
        Margin="5"/>
</StackPanel>
</Grid>
</Window>

```

Cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
```

```
namespace SportClub
```

```
{
    public partial class AddEditWindow : Window
    {
        public Users AllUsers { get; set; }

        public List<AbonementsType> AbonementsTypeSelect { get; set; }

        public List<Experts> ExpertsSelect { get; set; }

        public AddEditWindow(Users users)
        {
            InitializeComponent();
            DataContext = this;
            AllUsers = users;
            AbonementsTypeSelect = Core.DB.AbonementsType.ToList();
            ExpertsSelect = Core.DB.Experts.ToList();
        }

        private void SaveButton_Click(object sender, RoutedEventArgs e)
        {
            try
```

					ПД.09.02.07.XXXX.ПЗ	Лист
						65
Изм.	Лист	№ докум.	Подпись	Дата		


```

    {
        if (AllUsers.AbonementsType == null)
            throw new Exception("Не выбран тип абонемента");

        if (AllUsers.Experts == null)
            throw new Exception("Не выбран эксперт");

        if (AllUsers.ID == 0)
            Core.DB.Users.Add(AllUsers);

        Core.DB.SaveChanges();

        DialogResult = true;

        MessageBox.Show($"Сохранено");
    }
    catch(Exception ex)
    {
        MessageBox.Show($"Ошибка: {ex.Message}");
    }
}

private void CloseButton_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}
}
}

```

App.xaml :

```
<Application x:Class="SportClub.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:SportClub"
    StartupUri="StartWindow.xaml">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Light.xaml" />
                <ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Defaults.xaml" />
                <ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Primary/MaterialDesignColor.DeepPurple.xaml" />
                <ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Accent/MaterialDesignColor.Lime.xaml" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</Application>
```

					ПД.09.02.07.XXXX.ПЗ	Лист
						67
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения курсового проекта средствами MySQL Workbench была разработана информационная система «Спортклуб «SportClub»», предназначенная для автоматизации работы администратора спортклуба.

Концептуальная модель данных представлена диаграммой «сущность-связь». На ее основании разработана логическая структура базы данных, в ходе реализации которой задействованы механизмы обеспечения целостности данных.

Создание главной формы доступа к данным позволяет пользователям легко просматривать, обновлять или анализировать данные.

Разработанное приложение просто в применении и может быть использовано в любом спортклубе.

Формы отображают информацию из таблиц в удобном виде, а также служат для ввода данных в таблицы.

Сопоставление результатов проекта с поставленными задачами позволяет заключить следующее:

- На основе теоретического анализа литературы и Internet-источников произведен анализ предметной области спортклуба.
- Проведено функционально-ориентированное проектирование информационной системы.
- Разработана инфологическая модель данных.
- Спроектирована логическая структура информационной системы.
- Разработана физическая структура информационной системы.
- Разработаны запросы, отчеты к информационной системе.
- Разработан интерфейс БД.

Список информационных источников

1. Бурков А.В. Проектирование информационных систем [Текст]/ Бурков А.В.- Йошкар-Ола, МарГУ 2009 г. – 312с.
2. ВикрамВасвани. Полный справочник по MySQL. Москва, 2006.
3. Гвоздева Т.В., Баллод Б.А. Проектирование информационных систем DJVU [Текст]/ Гвоздева Т.В., Баллод Б.А. -Ростов н/д.: Феникс, 2009. – 512 с.
4. Горохова Т.Н. Разработка и эксплуатация информационных систем Учебное пособие для студентов среднего профессионального образования. [Текст]/ Горохова Т.Н. – Санкт-Петербург: Санкт-Петербургский колледж управления и экономики "Александровский лицей", 2010. – 84с.
5. Громаков Е.И. Проектирование интегрированных компьютерных систем управления: Учебно-методическое пособие PDF [Текст]/ Громаков Е.И – Томск: Изд-во Томского политехнического университета, 2012 г.– 168 с
6. Джон КоггЗолл. PHP 5. Полное руководство. Диалектика, 2006
7. Казаченко, Колобашкина Т.В. и др. Безопасность жизнедеятельности. Промышленная и экологическая безопасность. Методические указания к дипломному проектированию, СПб-ГУАП, 2001
8. Коваленко В.В. Проектирование информационных систем Учеб.пособие [Текст]/ Коваленко В.В.-Рязань, Рязанский государственный радиотехнический университет, 2006г. – 184 с.
9. Люк Веллинг, Лора Томсон. Разработка Web – приложений с помощью PHP и MySQL.Вильямс, 2005
10. Мазупкевич А. PHP. Настольная книга программиста. Новое издание, 2003
11. Определение эффективности инвестиций. Методические указания к выполнению дипломного проекта. СПб, 2002
12. Прогнозирование элементов бизнес-плана проектов. Методические указания к выполнению дипломных проектов. СПб 2002.
13. Роберт Дж. Мюлер. Базы данных и UML. Справочник в электронном виде, 2001
14. Смирнов Н.В. Проектирование информационных систем [Текст]/ Смирнов Н.В. – Санкт -Петербург, БГТУ "ВОЕНМЕХ", 2008 г. – 146 с.
15. Справочное руководство пользователя по пакету RationalRose.
16. Трофимов С.А. Case – технологии. Практическая работа в RationalRose. Бином 2001, 272с.
17. УндиБоггс, Майкл Боггс. UML и RationalRose. Лори, 2004

					ПД.09.02.07.XXXX.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		69

18. Виды диаграмм [Электронный ресурс] – Режим доступа: <http://www.intuit.ru/studies/courses/1007/229/lecture/5954?page=2> – 15.11.2016г.
19. <http://www.exponenta.ru/soft/others/mvs/stud2/23.asp> – 20.11.2016г. – Компьютерный лабораторный практикум "Моделирование"[Электронный ресурс]
20. Режим доступа: http://citforum.ru/database/advanced_intro/31.shtml – 16.11.2016 г. – Кузнецов С. Базы данных. Вводный курс [Электронный ресурс]
21. Репозитории преподавателя github.com/kolei