

Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
федерального государственного бюджетного образовательного учреждения высшего
образования
**«Владимирский государственный университет
Имени Александра Григорьевича и Николая Григорьевича Столетовых»
(МИВлГУ)**

Факультет _____ ИТР
Кафедра _____ ПИН

КУРСОВАЯ РАБОТА

По _____ Распределенные системы обработки данных
Тема _____ Распределенная ИС «АРМ администратора гостиницы»

(оценка)

Руководитель
Привезенцев Д.Г.
(фамилия, инициалы)

(подпись) (дата)

Члены комиссии

(подпись) (Ф.И.О.)

Студент ПИН-120
(группа)

Андронов И.А.
(фамилия, инициалы)

(подпись) (Ф.И.О.)

(подпись) (дата)

Муром 2023

Место для задания

Целью данной курсовой работы является разработка распределенной ИС для автоматизации рабочего места администратора гостиницы по технологии ASP.NET Core MVC. Программа реализована на языке C# в среде разработки «Microsoft Visual Studio 2022».

The purpose of this course work is to develop a distributed IS for automating the workplace of a hotel administrator using technology ASP.NET Core MVC. The program is implemented in C# in the development environment "Microsoft Visual Studio 2022".

Содержание

Введение	3
1 Анализ технического задания	4
2 Разработка моделей данных	8
3 Проектирование работы системы	11
<u>3.1 Разработка моделей данных.....</u>	<u>15</u>
4 Разработка и реализация системы	18
4.1 Выбор инструментов реализации	18
4.2 Разработка системы.....	19
5 Тестирование	21
6 Руководство по работе с приложением.....	24
6.1 Руководство пользователя.....	24
6.2 Руководство программиста	34
Заключение.....	46
Приложение	47

					МИВУ.09.03.04-01.000 ПЗ								
Изм	Лист	№ докум.	Подп.	Дата	Распределенная ИС АРМ администратора гостиницы					Лит.		Лист	Листов
Разраб.		Андронов И.А.										5	54
Провер.		Привезенцев Д.Г.								МИВУ ПИН - 120			
Н.контр.													
Утв.													

Введение

В современных условиях одно из приоритетных направлений российской экономики - выработка основных теоретических и методологических позиций по применению менеджмента на предприятиях и в различных сферах деятельности, в том числе и администрировании гостиниц.

Актуальность данной темы заключается в том, что эффективное управление в современных условиях рынка - необходимое условие повышения эффективности бизнеса, создания, развития и реализации конкурентных преимуществ предприятия.

Перед современным руководителем встают стратегические вопросы выбора направления развития бизнеса и определения ключевых конкурентных преимуществ компании. Выработка такого видения и управления компетенцией организации является ключевой, предпринимательской, функцией менеджмента компании.

Много внимания руководство вынуждено уделять вопросу оптимизации структур компании. Вместе с тем, для современного руководителя одинаково важен вопрос создания системы и технологии управления, которая обеспечивает эффективную операционную деятельность компании.

Совершенствование управления невозможно без изучения, систематизации и обобщения опыта развития общественных отношений как в экономике в целом, так и в области управления гостиницами. В данной курсовой работе производится разработка распределенной информационной системы автоматизированного рабочего места администратора гостиницы.

					МИВУ.09.03.04-01.000 ПЗ	Лист
						3
Изм	Лист	№ докум.	Подп.	Дата		

1 Анализ технического задания

1.1 Описание предметной области

Деятельность администратора гостиницы связана с переработкой и хранением большого количества информации. Огромный документооборот затрудняет работу, снижает эффективность доступа к нужной информации.

Из-за этого и возникает необходимость автоматизации процесса регистрации и работы с клиентами. Автоматизация позволит сократить время, необходимое для регистрации клиентов и поиска нужной информации.

Одной из основных задач администратора гостиницы является бронирование номеров, а также прием, регистрация, размещение и выписка посетителей. В процессе оформления, специалистом заносится основная информация о клиенте, предоставленном номере и услугах, сроках и конечной цене проживания в номере.

Целью работы является разработка и создание автоматизированного рабочего места администратора гостиницы для облегчения внесения и учета данных, упрощения введения документооборота и сокращения затрат. Выполнение данных требований позволит снизить затраты на создание программного продукта и создать качественную систему, позволяющую повысить экономическую эффективность.

1.2 Формирование требований к системе.

На основании произведенного анализа предметной области, сформированы требования к разрабатываемому приложению:

- Хранение информации о всех типах номеров, информации о клиентах (после выбытия включительно) и услугах;
- Возможность добавления новых услуг или редактирование существующих;
- Возможность оплаты клиентом нескольких номеров.

Дополнительные требования к разрабатываемой системе:

					МИВУ.09.03.04-01.000 ПЗ	Лист
						4
Изм	Лист	№ докум.	Подп.	Дата		

- Авторизация пользователей (использование Microsoft Identity);
- Интерфейс и доступный функционал зависит от роли пользователя (2 роли);
- Адаптивный интерфейс;
- Валидация данных на стороне клиента (на предмет корректности формата данных)
- и на стороне сервера (на предмет соответствия данных логике работы);
- Формирование отчетов в виде файлов Excel таблиц и pdf-файлов;
- Отправка уведомлений пользователям в виде e-mail писем.

1.3 Обоснование выбора средств разработки.

В ходе разработки приложения для курсового проекта выбор среды разработки пал на Visual Studio 2022. Для разработки серверной части выбран язык программирования C#. В качестве системы управления базами данных было решено использовать SQLite. Выбор SQLite обусловлен его простотой и гибкостью, возможностью встраивания прямо в приложение, что идеально подходит для меньших проектов и упрощает развертывание и поддержку. SQLite способен эффективно обрабатывать как структурированные, так и неструктурированные данные.

Исходя из вышеперечисленного, программный продукт будет иметь следующие системные требования:

- ЦП x64 с тактовой частотой 1 ГГц;
- Объем оперативной памяти 256 Мб;
- Операционная система Windows 8/8.1/10;
- Свободное место на жестком диске: 500 Мб.

1.3.1 Описание среды разработки Visual Studio 2022

Microsoft Visual Studio – полнофункциональная интегрированная среда

					МИВУ.09.03.04-01.000 ПЗ	Лист
Изм	Лист	№ докум.	Подп.	Дата		5

разработки, или IDE, с поддержкой популярных языков программирования, среди которых C, C++, VB.NET, C#, F#, JavaScript, Python.

Функциональность Visual Studio охватывает все этапы разработки программного обеспечения, предоставляя современные инструменты для написания кода, проектирования графических интерфейсов, сборки, отладки и тестирования приложений. Возможности Visual Studio могут быть дополнены путем подключения необходимых расширений.

Редактор кода Visual Studio поддерживает подсветку синтаксиса, вставку фрагментов кода, отображения структуры и связанных функций. Существенно ускорить работу помогает технология IntelliSense – автозавершение кода по мере ввода.

1.3.2 Выбор и обоснование языка разработки.

C# («Си Шарп») – один из наиболее быстро растущих, востребованных и при этом «удобных» языков программирования. Это модификация фундаментального языка C от компании Microsoft, призванная создать наиболее универсальное средство для разработки программного обеспечения для большого количества устройств и операционных систем.

C# – это объектно-ориентированный язык программирования. Он был создан в период с 1998 по 2002 год командой инженеров Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота.

Язык входит в семью C-подобных языков. Синтаксис приближен к Java и C++. Его особенности:

- статистическая типизация,
- поддерживается полиморфизм,
- поддерживается перегрузка операторов,
- доступна делегация, атрибуты, события, обобщенные типы и анонимные функции.

Разработка Microsoft много особенностей унаследовала у Delphi, Smalltalk и

					МИВУ.09.03.04-01.000 ПЗ	Лист
Изм	Лист	№ докум.	Подп.	Дата		6

Java. При этом создатели нового языка исключили из своего детища многие практики и спецификации, считающиеся «проблемными».

Язык C# практически универсален. Можно использовать его для создания любого ПО: продвинутых бизнес-приложений, видеоигр, функциональных веб-приложений, приложений для Windows, macOS, мобильных программ для iOS и Android.

Среди основных преимуществ C# выделяют:

1. C# популярен за счет своей «простоты». Простоты для современных программистов и больших команд разработчиков, чтобы те могли в сжатые сроки создавать функциональные и производительные приложения. Этому способствуют нетипичные конструкции языка и специфичный синтаксис, помогающий максимально органично реализовать намеченные функции.

2. Популярность языка – еще одно значимое преимущество. Большое количество поклонников C# способствуют его развитию. Также это благоприятно влияет на рост числа вакансий, связанных с разработкой на языке Microsoft. Программисты, хорошо знакомые с C#, востребованы в индустрии, несмотря на их большое и постоянно увеличивающееся количество.

3. Понятный синтаксис C# заметно упрощает не только разработку как таковую, но и другие важные аспекты совместной работы, например, чтение чужого кода. Это упрощает процесс рефакторинга и исправления ошибок при работе над приложениями в больших командах.

4. Также нельзя не упомянуть низкий порог вхождения. C# – популярная и достаточно простая в освоении технология. Уже через полгода можно поднатореть в разработке и начать делать полноценные программы.

					МИВУ.09.03.04-01.000 ПЗ	Лист
						7
Изм	Лист	№ докум.	Подп.	Дата		

2 Разработка моделей данных

Технология ASP.NET Core MVC, выбранная в качестве основы для реализации распределённой информационной системы автоматизированного рабочего места администратора гостиницы, обеспечивает использование подхода Model-View-Controller (MVC). Это надёжный и устойчивый архитектурный шаблон, который способствует организации приложения на три основные части: модель (Model), представление (View) и контроллер (Controller).

Модель (Model) — это ядро приложения, отвечающее за бизнес-логику и управление данными. Она включает в себя классы, свойства, методы и другие элементы, необходимые для работы с информацией, включая взаимодействие с базами данных, веб-службами и прочими источниками данных.

Контроллер (Controller), в свою очередь, является своего рода "дирижёром", управляющим потоком данных между моделью и представлением. Используя модель, контроллер получает необходимую информацию, а затем передает её в представление для дальнейшего отображения пользователю.

Представление (View) формирует визуализацию данных, предоставленных контроллером. Это "лицо" приложения, с которым взаимодействует пользователь.

Таким образом, использование ASP.NET Core MVC обеспечивает четкое разделение ответственности между компонентами системы, что упрощает разработку и поддержку приложения.

Для моего приложения были выделены следующие основные модели данных:

Модель «Апартаменты» содержит:

- уникальный идентификатор апартамента;
- внешний ключ, указывающий на тариф, который предусмотрен в данном номере;
- Порядковый номер апартамента;
- Площадь апартамента;

– статус апартамента, «Зарезервирован», «Не зарезервирован». Данное свойство используется исключить во внутренних алгоритмах, не доступных для обычных пользователей;

- Список удобств апартамента;
- Список включенных в апартаменты услуг;

Модель «Тариф» содержит:

- Уникальный идентификатор тарифа;
- Название тарифа;
- Описание тарифа;
- Цена тарифа;

Модель «Клиент» содержит:

- Уникальный идентификатор клиент;
- имя клиента;
- фамилия клиента;
- отчество клиента;
- телефон сотрудника;
- электронная почта сотрудника;

Модель «Бронирование» содержит:

- уникальный идентификатор бронирования;
- начальная дата бронирования;
- конечная дата бронирования;
- внешний ключ, указывающий на апартаменты, прикрепленные к данному бронированию;
- внешний ключ, указывающий на клиента, прикрепленного к данному бронированию;
- статус бронирования, «Активен», «Не активен»;

					МИВУ.09.03.04-01.000 ПЗ	Лист
						9
Изм	Лист	№ докум.	Подп.	Дата		

– список дополнительных услуг, которые выбрал клиент при бронировании апартаментов.

Модель «Услуги» содержит:

- уникальный идентификатор услуги;
- название услуги;
- описание услуги;
- цена услуги;

Модель «Удобства» содержит:

- уникальный идентификатор удобства;
- название удобства;

Модель «Архив заказов» содержит:

- уникальный идентификатор удобства;
- название удобства;

					МИВУ.09.03.04-01.000 ПЗ	Лист
						10
Изм	Лист	№ докум.	Подп.	Дата		

3 Проектирование работы системы

Проектирование работы системы является важным этапом в разработке программного обеспечения. Это процесс создания плана, описывающего структуру, функциональность и взаимодействие компонентов системы. Разрабатываемая система будет иметь 7 моделей данных: клиент, апартаменты, бронирование, тариф, услуги, удобства, архив бронирований.

Ниже приведена диаграмма классов (рисунок 1), которая описывает свойства, присущие к каждой модели:

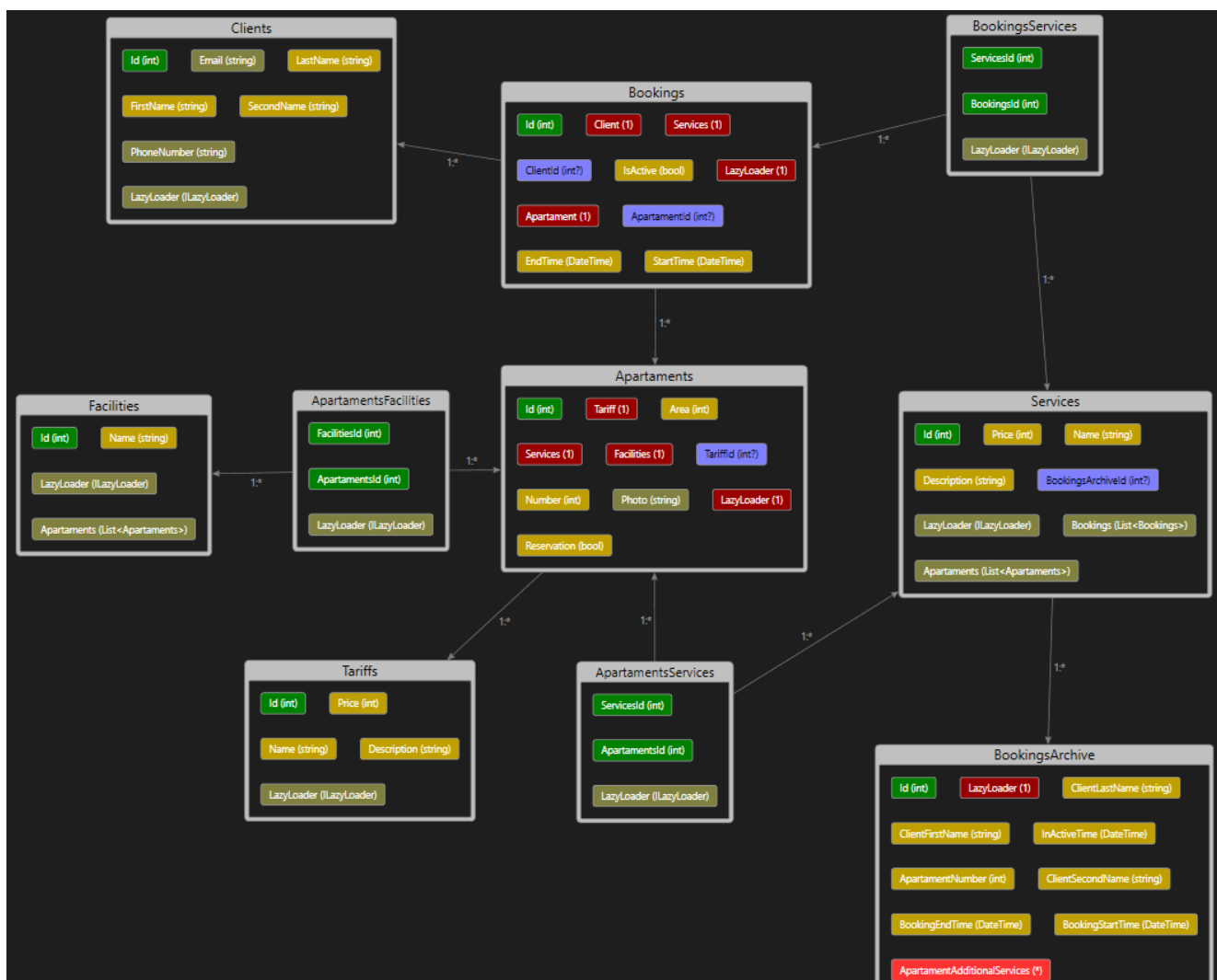


Рисунок 1 – Диаграмма классов

Для описания функциональности системы с точки зрения ее пользователей, используется диаграмма прецедентов. Она позволяет идентифицировать актеров (пользователей) системы и действия (прецеденты), которые они могут выполнять в этой системе. В системе предусмотрены 3 роли пользователей: администратор, менеджер, клиент. Для каждой роли приведена диаграмма вариантов использования (рисунок 2-6).

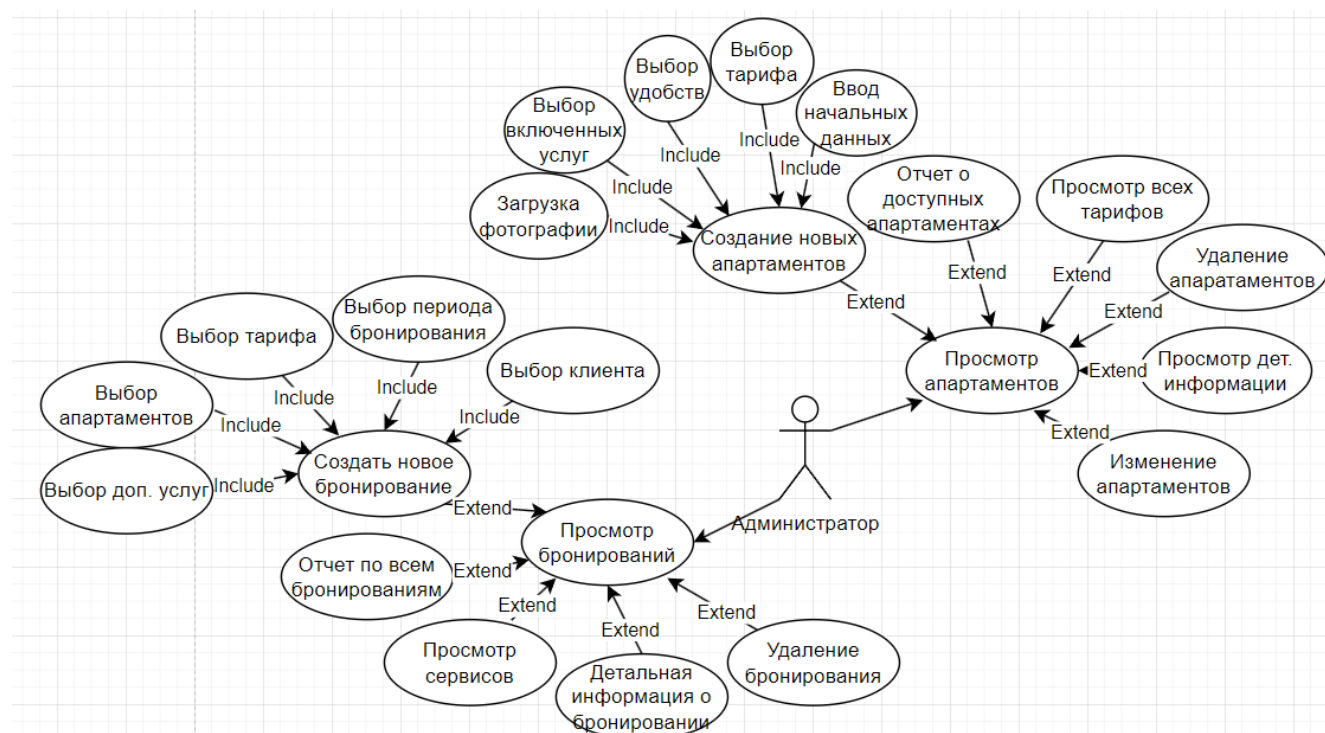


Рисунок 2 – Диаграмма вариантов использования для роли «Администратор».

Часть 1.

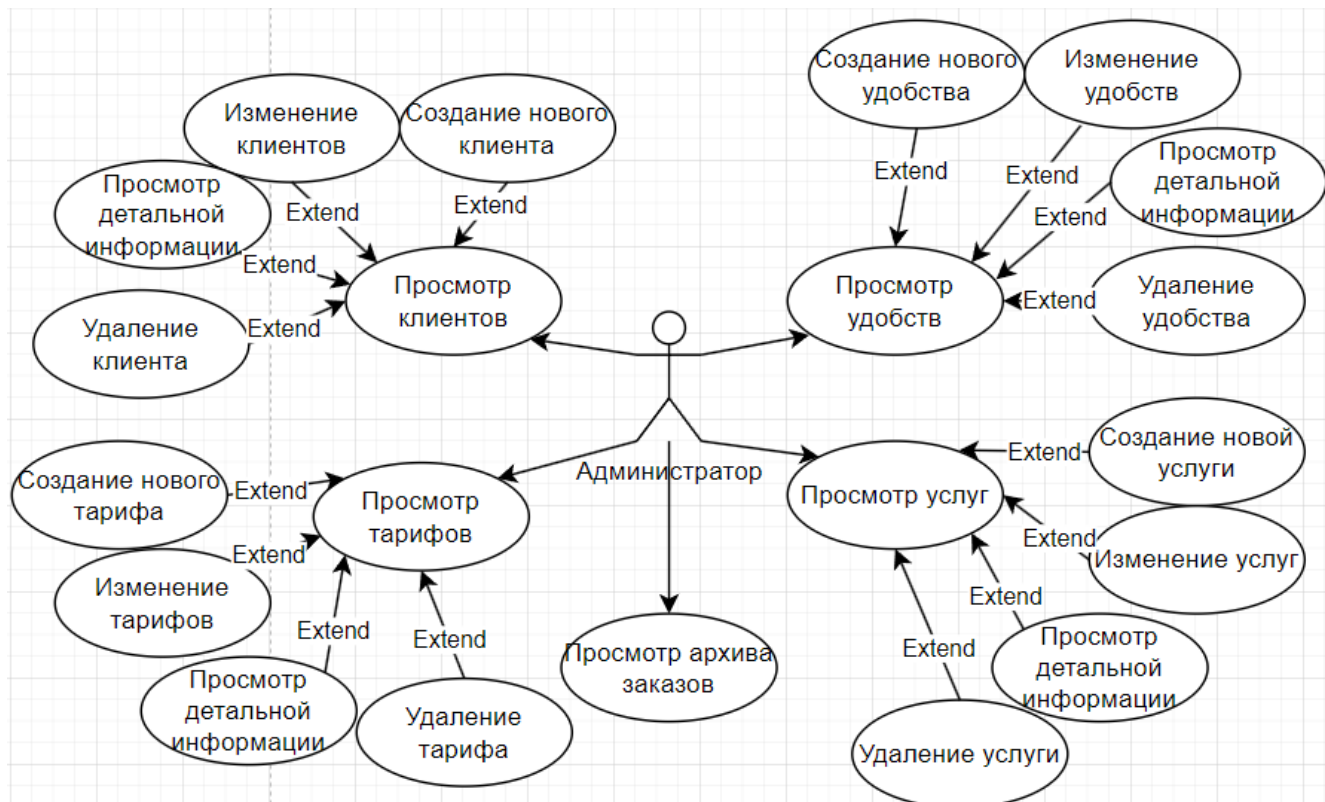


Рисунок 3– Диаграмма вариантов использования для роли «Администратор».

Часть 2.



Рисунок 4 – Диаграмма вариантов использования для роли «Менеджер». Часть 1.



Рисунок 5 – Диаграмма вариантов использования для роли «Менеджер». Часть 2.

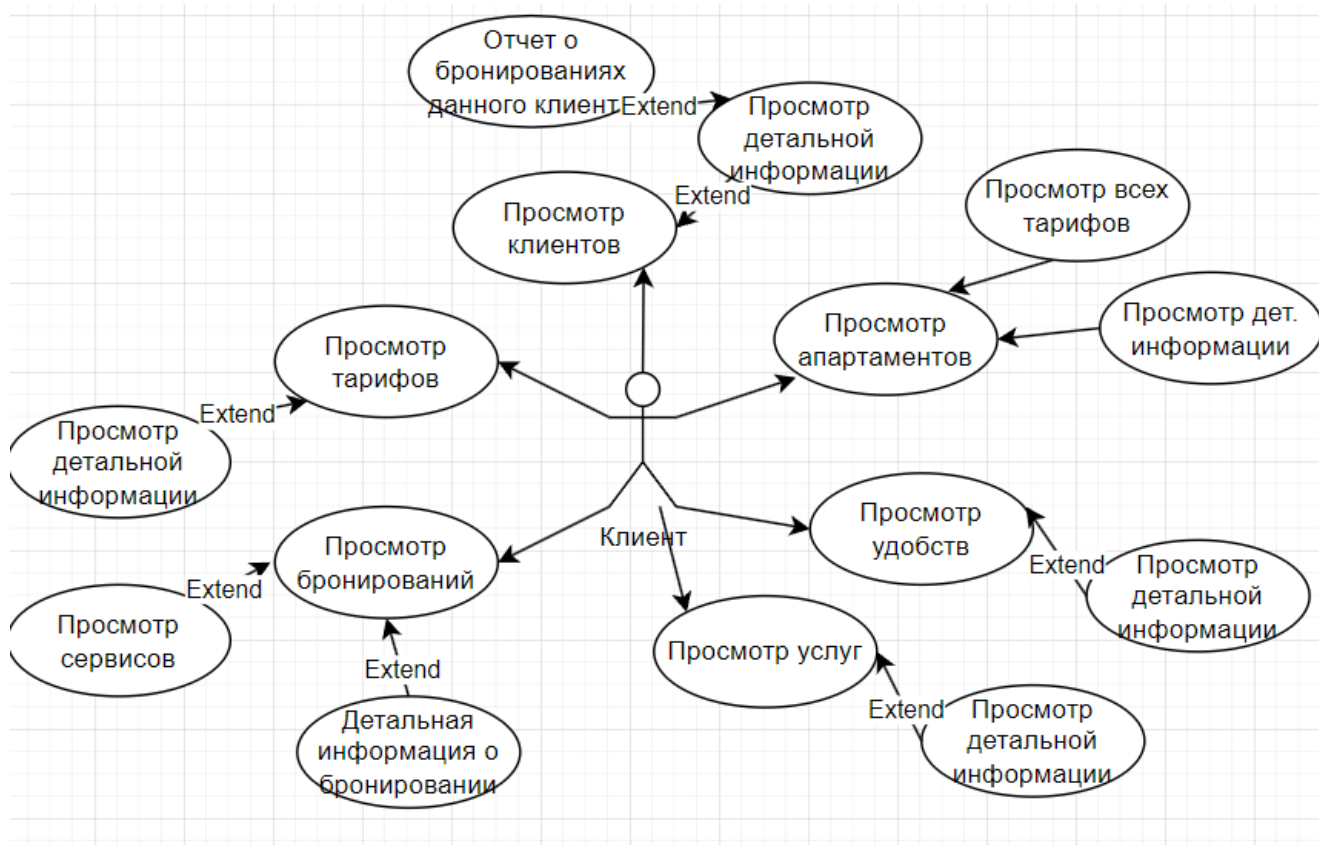


Рисунок 6 – Диаграмма вариантов использования для роли «Клиент».

3.1 Разработка моделей данных

Для минимизации возможных ошибок в процессе разработки базы данных и взаимодействующего с ней приложения возникает необходимость визуализации и структуризации данных. Для организации этих процессов было принято решение разработать концептуальную, логическую и физическую модели данных.

3.1.1 Концептуальная модель базы

В предметной области существуют 7 сущностей: клиент, апартаменты, бронирование, архив бронирований, услуги и удобства и тарифы. Клиент содержит в себе такие атрибуты как: ФИО, электронная почта и номер телефона. Номер включает в себя фото номера, список удобств, тариф, площадь, скрытое логическое свойство «Резервация» и включенные услуги. В атрибуты тарифа входят: название тарифа, цена тарифа, описание тарифа. В атрибуты удобства входит название. Услуги включают в себя такие атрибуты, как: название услуги, описание услуги и цена услуги. И последняя сущность представляет из себя архив истякших бронирований. Её атрибуты: номер апартаментов, который был привязан к бронированию, ФИО клиента, прикрепленного к бронированию, список дополнительных услуг, а так же время начала, конца и удаления бронирования.

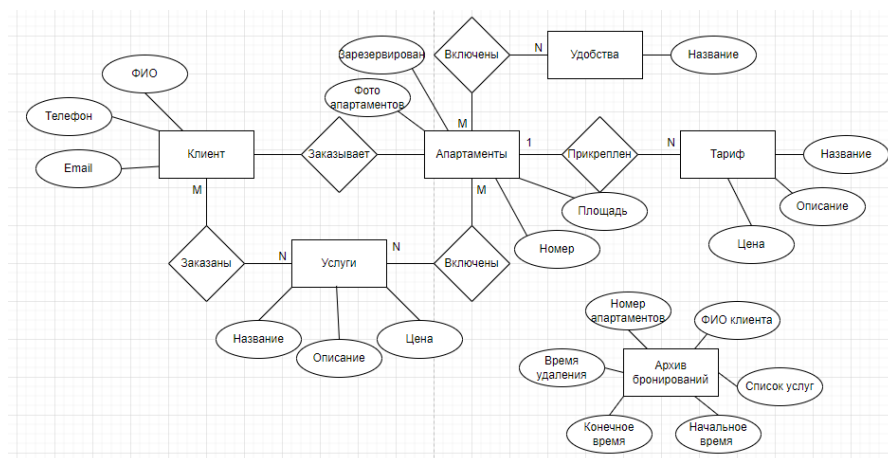


Рисунок 7 – Концептуальная модель.

3.1.2 Логическая модель

Дальнейшее проектирование внутреннего устройства базы данных после нормализации таблиц следует начать с логического представления возможных таблиц и содержащихся в них данных. Отличным инструментом для этого будет являться логическая модель данных, на которой можно отразить все таблицы, их атрибуты, а также связи между ними в том представлении, в каком они будут храниться в базе данных.

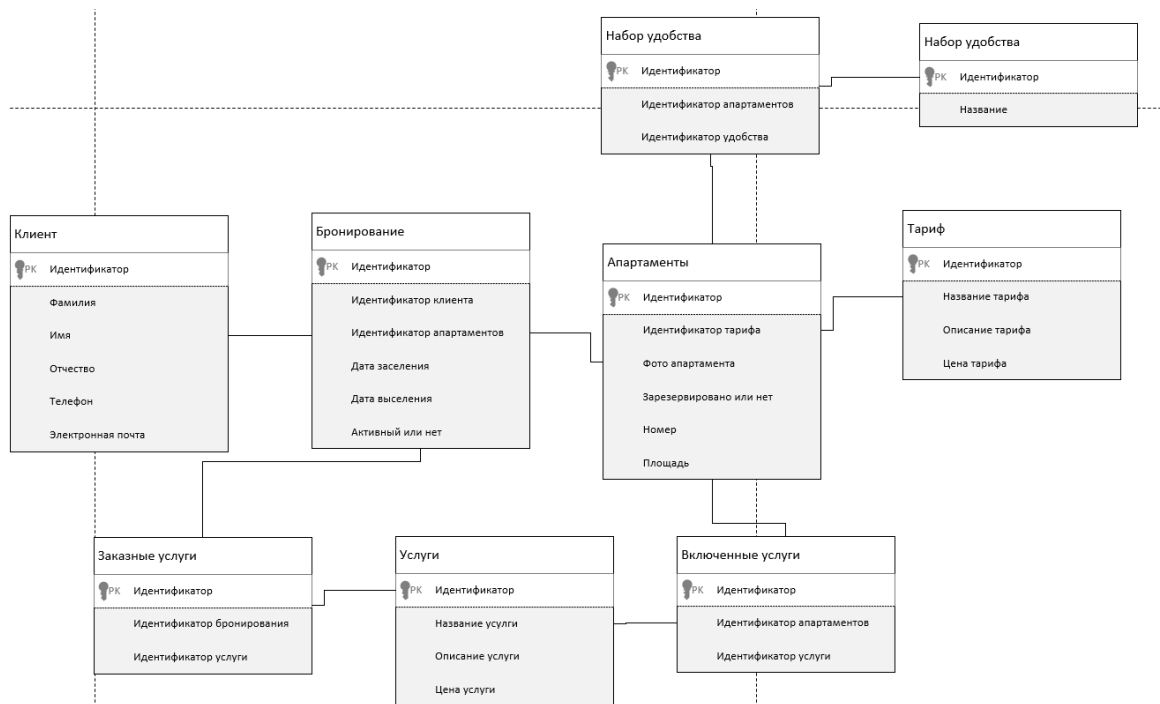


Рисунок 8. Логическая модель.

3.1.3 Физическая модель

После разработки логической модели данных возникла необходимость создания физической модели данных, демонстрирующей таблицы, содержащиеся в них данные и их типы, связи в самой базе данных. Именно по физической модели данных в дальнейшем будет вестись разработка.

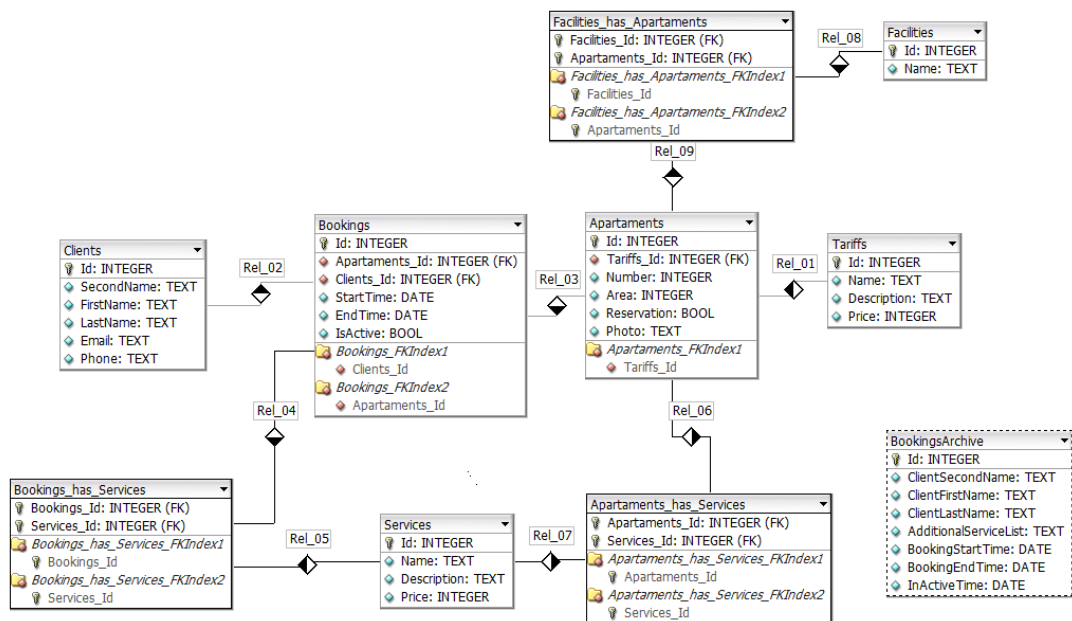


Рисунок 9. Физическая модель

4 Разработка и реализация системы

4.1 Выбор инструментов реализации

Распределенная система построена на основе платформы ASP.NET Core, используя инструменты, такие как SQLite, NuGet, Git, в среде разработки Visual Studio 2022.

Во-первых, платформа ASP.NET MVC представляет собой мощный инструмент с богатой библиотекой для разработки веб-приложений различных уровней сложности. Поддерживая множество языков программирования, включая C#, VB.NET и F#, она гарантирует высокую производительность и безопасность.

Во-вторых, SQLite - это компактная, самодостаточная, сервер-независимая и конфигурация-независимая система управления базами данных. Она обеспечивает полнофункциональную поддержку транзакций SQL, включая представления, триггеры и хранимые процедуры. Помимо этого, SQLite отличается высокой надежностью и производительностью, а также возможностью легкого встраивания в приложения любого размера.

В-третьих, NuGet служит инструментом управления зависимостями и упрощает процесс внедрения сторонних библиотек в проект. Это существенно упрощает процесс разработки и обновления приложения.

Наконец, Git служит мощной системой контроля версий, обеспечивая эффективное управление кодом и сотрудничество в команде. Он позволяет легко отслеживать изменения, возвращаться к предыдущим версиям и решать конфликты при слиянии кода.

Среда разработки Visual Studio 2022 предоставляет все необходимое для разработки приложений на платформе .NET: удобный набор инструментов для разработки, тестирования и отладки приложений, интегрированный менеджер NuGet и поддержку Git.

В общем и целом, выбор платформы ASP.NET Core, инструментов SQLite, NuGet и Git, в совокупности с средой разработки Visual Studio 2022, гарантирует

					МИВУ.09.03.04-01.000 ПЗ	Лист
						18
Изм	Лист	№ докум.	Подп.	Дата		

высокую производительность, безопасность и эффективность в процессе разработки приложений. Это надежные и проверенные инструменты, которые поддерживают создание приложений любого уровня сложности.

4.2 Разработка системы

На этапе разработки распределенной информационной системы для автоматизированного рабочего места администратора гостиницы выполнены следующие действия:

1. Определены модели данных:

- Модели данных представляют основные сущности и таблицы базы данных в проекте. В разрабатываемой системе определены следующие модели: HotelContext – контекст для доступа к данным, Apartments – сущность апартаментов, то есть номера для проживания, Bookings– сущность, отвечающая за бронирование апартаментов, Clients – сущность клиента, Facilities – сущность удобства, Services– сущность услуги, Tariffs – сущность тарифа, включенного в апартаменты, BookingsArchive – сущность закончившихся бронирований.

- Указаны данные атрибутов у моделей Data Annotations для задания правил валидации, атрибутов Display для определения отображаемых имен полей и других атрибутов для настройки свойств моделей.

2. Разработаны контроллеры:

- Контроллеры обрабатывают запросы от пользователей и взаимодействуют с моделями и представлениями.

- Определены действия (методы) контроллера для обработки различных типов запросов, таких как GET, POST. Большая часть контроллеров содержит методы отображения, удаления, обновления данных. Стоит выделить контроллер с моделью бронирования, который, не считая стандартных операций с данными, содержит методы взаимодействия с содержащими списками и взаимодействия не только с сущностью «Бронирование», но и «Апартаменты», статус который меняется от того, содержится ли она в заказе или нет.

					МИВУ.09.03.04-01.000 ПЗ	Лист
						19
Изм	Лист	№ докум.	Подп.	Дата		

3. Созданы представления:

- Представления отображают данные пользователю и обрабатывают пользовательский ввод. Каждому контроллеру соответствует как минимум 3 View-представления: Index - отображает все записи текущей модели, Details – показывает подробную информацию выбранной пользователем записи, Delete – страница с подтверждением удаления выбранной записи, которая содержит подробную информацию о удаляемой записи.

- Определены HTML-шаблонов, форм и элементов управления для взаимодействия с пользователем, стилизация и адаптивность интерфейса системы реализована с помощью инструментария Bootstrap 5.3.

4. Настройка маршрутизации:

- Определены маршрутов, которые связывают URL-адреса с соответствующими действиями контроллера и представлениями.

5. Настройка доступа к данным:

- Для доступа к базе данных и выполнения операций CRUD (создание, чтение, обновление, удаление) над моделями данных, используется Entity Framework.

- Выполнена настройка подключения к базе данных, определение схемы базы данных и выполнение миграций при необходимости.

6. Добавлена аутентификации и авторизации на базе инструмента Identity ASP.NET Core.

- Identity ASP.NET Core - это мощный инструмент для обеспечения безопасности приложения и управления пользователями. Он облегчает процесс создания и управления системой аутентификации и авторизации. Для системы авторизации, регистрации определены 4 роли: клиент, рабочий, прораб, администратор, каждая роль имеет свой набор действий, к которым имеет доступ, определяемых в контроллере и view-представлении.

					МИВУ.09.03.04-01.000 ПЗ	Лист
						20
Изм	Лист	№ докум.	Подп.	Дата		

5 Тестирование

Целью проведения тестирования является подтверждение реализации требуемой функциональности системы. В процессе тестирования, результат каждого теста фиксируется. Считается, что тест прошел успешно в случае, если результат совпадает с ожидаемым результатом. Если результат отличается от ожидаемого, он фиксируется в протоколе тестирования. Так как все CRUD операции в моделях схожи, протестируем одну выборочную сущность на CRUD операции, и нетипичные алгоритмы создания сущностей, таких как Apartments, и Bookings. В таблице 1 приведена методика тестирования разработанного программного продукта.

Таблица 1 - Тестирование

Выполненное действие	Полученный результат
1. Пользователь зашел на сайт и нажал на кнопку «Клиенты» в навигационной панели.	Открытие страницы авторизации.
2. Пользователь авторизовался как «Администратор».	Открытие страницы Index сущности «Clients»
3. Пользователь выбрал нужного ему клиента и нажал на кнопку «Изменить», изменил данные и нажал на кнопку «Сохранить».	Открытие страницы изменения текущего пользователя. Данные выбранного клиента изменились. Открытие Index страницы клиентов.
4. Пользователь нажал выбрал нужного ему клиента и нажал на кнопку «Детальная информация»	Открытие Details страницы с подробными данными о текущем клиенте.
5. Пользователь вышел из детальной информации, выбрал клиента и нажал на кнопку «Удалить». После этого	Открытие Delete страницы с текущим пользователем. Удаление текущего экземпляра клиента из базы данных. Удаление бронирований с участием

пользователь еще раз нажал на «Удалить».	текущего клиента. Перенаправление на страницу Index клиентов.
6. Пользователь нажал на кнопку «Создать нового клиента», ввел корректные данные и нажал на кнопку «Создать».	Открытие Create страницы контроллера ClientsController, регистрация новой записи Clients в базе данных.
7. Пользователь перешел на страницу «Апартаменты» и нажала на кнопку «Создать новые апартаменты».	Открытие начального этапа многоступенчатого контроллера для создания новых апартаментов.
8. Пользователь ввел порядковый номер и площадь апартаментов и нажал «Далее»	Открытие второго этапа контроллера – выбор тарифа.
9. Пользователь выбрал один из представленных тарифов и нажал «Выбрать».	Открытие третьего этапа контроллера, выбор удобств.
10. Пользователь выбрал конкретные удобства, активируя чекбоксы, и нажал «Далее».	Открытие четвертого этапа контроллера, выбор услуг.
11. Пользователь выбрал конкретные услуги и нажал «Далее»	Открытие последнего этапа контроллера – загрузка фотографии
12. Пользователь загрузил/не загрузил фотографию и нажал «Загрузить фотографию».	Формирование новой сущности апартаментов из данных, которые были введены/выбраны на протяжении всех этапов контроллера. Вывод страницы Index апартаментов.
13. Пользователь нажал на кнопку «Бронирования» на панели	Открытие начального этапа многоступенчатого контроллера для создания новой брони.

навигации, а затем на «Создать новую бронь».	
14. Пользователь выбрал на конкретного клиента и нажал на «Выбрать»	Переход на следующий этап контроллера к выбору начальной и конечной дат бронирования.
15. Пользователь выбрал нужные даты и нажал на кнопку «Далее».	Переход к третьему этапу контроллера – выбору тарифа.
16. Пользователь выбрал один из представленных тарифов и нажал на кнопку «Выбрать»	Переход к предпоследнему этапу – выбору апартаментов. Пользователю представляется выбор апартаментов, которые удовлетворяют условиям, которые пользователь ввел/выбрал на предыдущих этапах, а именно: тариф, даты.
17. Пользователь выбрал нужные ему апартаменты и нажал на кнопку «Выбрать».	Открытие последнего этапа контроллера – выбор дополнительных услуг.
18. Пользователь выбрал/не выбрал услуги и нажал на кнопку «Далее»	Регистрация новой записи таблицы «Bookings» на основе входных данных, которые ввел/выбрал пользователь. Переадресация на Index контроллера Bookings.

6 Руководства по работе с приложением

6.1 Руководство пользователя

Данное руководство содержит описывает функциональные возможности пользователи системы, в ней существует 3 роли: «администратор», «клиент», «рабочий». Руководство описывает действия роли «администратор», который имеет все возможные действия в системы. В зависимости от вашей роли в системе, вы имеет разные возможности, но это не влияет на полноту описания руководства. Так же чтобы избавиться от дублирования в данном руководстве будут приведены только уникальные случаи.

Страница «Клиенты»:

Чтобы попасть на страницу «Клиенты», нужно использовать навигационное меню сверху (Рисунок 10).

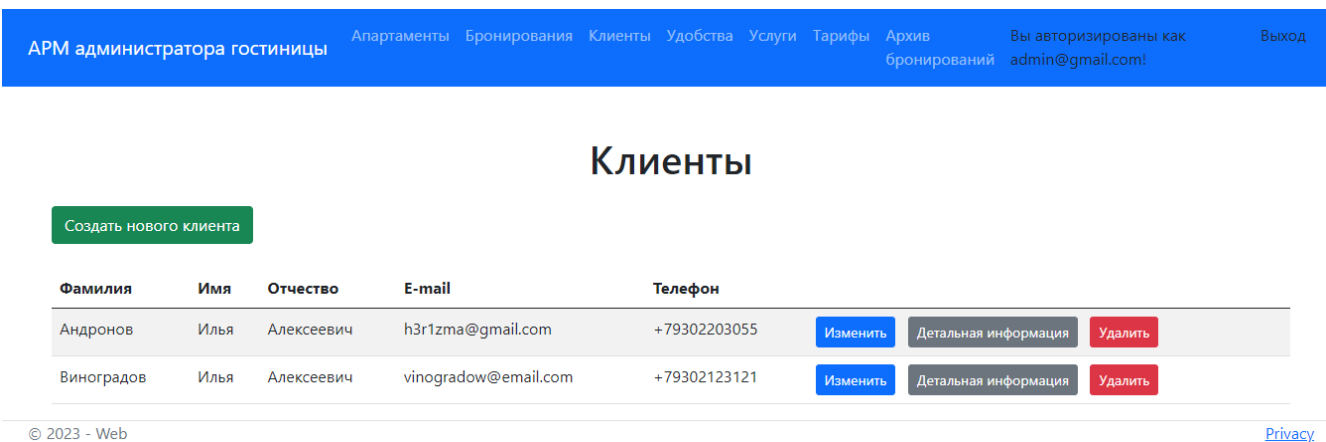


Рисунок 10 – Страница отображения всех клиентов

1. Создание нового клиента. Нажатие кнопки «Создать нового клиента» перенаправляет на страницу создания нового клиента (рисунок 11).

Создание клиента

Фамилия	<input type="text"/>
Имя	<input type="text"/>
Отчество	<input type="text"/>
Email	<input type="text"/>
Телефон	<input type="text"/>
<input type="button" value="Создать"/>	

[Вернуться к списку клиентов](#)

Рисунок 11 – Страница создания клиента.

Заполнив все нужные поля удовлетворяющими данными и нажав на кнопку «Создать», новый клиент отобразится на странице списка клиентов (рисунок 10).

2. Чтобы узнать подробную информацию о клиенте, нужно нажать кнопку «Детальная информация» у соответствующего клиента, впоследствии откроется страница с информацией. Существует два типа такой страницы. Если у данного клиента есть активные бронирования, то они отобразятся под карточкой детальной информации (Рисунок 12) или отобразиться надпись «Бронирований нет» (Рисунок 13).

APM администратора гостиницы	Апартаменты Бронирования Клиенты Удобства Услуги Тарифы Архив бронирований	Вы авторизованы как admin@gmail.com!	Выход		
Детальная информация					
Фамилия:	Виноградов				
Имя:	Илья				
Отчество:	Алексеевич				
E-mail:	vinogradow@email.com				
Телефонный номер:	+79302123121				
<input type="button" value="Изменить"/>					
<input type="button" value="Вернуться к списку клиентов"/>					
Бронирования клиента Виноградов Илья Алексеевич					
<input type="button" value="Отчет по моему заказу"/>					
Номер апарт.	Тариф	Начальная дата	Конечная дата	Дополнительные услуги	Общая цена
11	Room Only	19.05.2023 0:00:00	21.05.2023 0:00:00	Массаж	141912
					<input type="button" value="Удалить"/>

Рисунок 12 – Страница с детальной информацией клиента с активными бронированиями.

					МИВУ.09.03.04-01.000 ПЗ	Лист
Изм	Лист	№ докум.	Подп.	Дата		25

АРМ администратора гостиницы		Апартаменты	Бронирования	Клиенты	Удобства	Услуги	Тарифы	Архив бронирований	Вы авторизированы как admin@gmail.com!	Выход
------------------------------	--	-------------	--------------	---------	----------	--------	--------	--------------------	--	-------

Детальная информация

Фамилия:	Андронов
Имя:	Илья
Отчество:	Алексеевич
E-mail:	h3r1zma@gmail.com
Телефонный номер:	+79302203055

Изменить

Вернуться к списку клиентов

Бронирований нет

Рисунок 13 – Страница с детальной информацией клиента без бронирований.

3. Редактирование информации о клиенте. На рисунке 10, нажав на кнопку «Изменить» у соответствующего клиента, откроется новая страница с заполненными полями. Для обновления данных следует поменять значения полей на странице и нажать кнопку «Сохранить» (рисунок 14), после чего произойдет переход на страницу отображения всех клиентов, где отобразиться новая информация (рисунок 10).

АРМ администратора гостиницы		Апартаменты	Бронирования	Клиенты	Удобства	Услуги	Тарифы	Архив бронирований	Вы авторизированы как admin@gmail.com!	Выход
------------------------------	--	-------------	--------------	---------	----------	--------	--------	--------------------	--	-------

Изменение клиента

Фамилия	<input type="text" value="Андронов"/>
Имя	<input type="text" value="Илья"/>
Отчество	<input type="text" value="Алексеевич"/>
Электронная почта	<input type="text" value="h3r1zma@gmail.com"/>
Телефонный номер	<input type="text" value="+79302203055"/>

Сохранить

Вернуться к списку клиентов

Рисунок 14 – Страница редактирования клиента

4. Удаление клиента. Чтобы удалить клиента, следует нажать на кнопку «Удалить» около удаляемого клиента на рисунке 10, далее вы попадаете на страницу подтверждения удаления (рисунок 15). Кнопка «Удалить» удалит выбранного клиента и вернет на страницу отображения клиента, кнопка «Вернуться к списку» просто откроет страницу со всеми клиентами.

АРМ администратора гостиницы
Апартаменты
Бронирования
Клиенты
Удобства
Услуги
Тарифы
Архив бронирований
Вы авторизованы как admin@gmail.com!
Выход

Удаление клиента

Вы точно хотите это удалить?

Фамилия
Андронов

Имя
Илья

Отчество
Алексеевич

Электронная почта
h3r1zma@gmail.com

Телефонный номер
+79302203055

Удалить

Вернуться к списку клиентов

Рисунок 15 – Страница подтверждения удаления заказа

Добавление нового апартаменты:

1. Для того, чтобы создать новые апартаменты, нужно перейти на страницу «Апартаменты», которая находится на навигационной панели и нажать на кнопку «Создать новые апартаменты» (Рисунок 16).

Апартаменты

Создать новые апартаменты

Начальная дата

Конечная дата

Отчет по доступным апартаментам

Все тарифы

#	Картинка	Площадь (кв.м)	Цена (руб./день)	Тариф			
1		1	3050	Только комната	Изменить	Детальная информация	Удалить
2		2	3100	Только комната	Изменить	Детальная информация	Удалить
3		30	4750	Только комната	Изменить	Детальная информация	Удалить
11		1212	63600	Только комната	Изменить	Детальная информация	Удалить
11		1231	70706	Только комната	Изменить	Детальная информация	Удалить
12		12	3850	Только комната	Изменить	Детальная информация	Удалить

Рисунок 16 – Страница отображения апартаментов.

- Первый этап создания апартаментов. Нажатие кнопки «Создать новые апартаменты» перенаправляет на страницу ввода начальной информации (рисунок 17).

Ввод начальной информации

Введите порядковый номер

Введите площадь(м²)

Далее

Рисунок 17 – Страница 1 этапа создания апартаментов.

- Выбор тарифа. После того, как вы ввели корректные данные, вас перенаправит на следующий этап, а именно выбор тарифа. На данной странице, вы

можете выбрать тариф, создать новый или изменить уже существующий (Рисунок 18).

Рисунок 18 – Страница 2 этапа создания апартаментов.

4. Выбор удобств. После того, как вы выбрали нужный вам тариф, вас перенаправят на страницу выбора удобств. Здесь есть несколько вариантов действий, или вы можете создать новый тариф, а так же изменить уже существующий, или просто выбрать из списка, поставив галочку в чекбоксе и нажать «Далее» (Рисунок 19). Так же вы можете оставить чекбоксы пустыми и просто нажать «Далее». В таком случае, добавить удобства можно будет ли при изменении записи.

Рисунок 19 – Страница 3 этапа создания апартаментов.

5. Выбор услуг. После того, как вы выбрали удобства апартаментам, вас перенаправит на следующий, четвертый этап создания апартаментов, а именно – выбор услуг. Набор действий схож с предыдущим этапом, поэтому после того, как

вы выберите нужные услуги, нажмите кнопку «Далее» для перехода на заключительный этап (Рисунок 20).

АРМ администратора гостиницы
Апартаменты
Бронирования
Клиенты
Удобства
Услуги
Тарифы
Архив бронирований
Вы авторизованы как admin@gmail.com!
Выход

Выбор услуг

Создать новую услугу

Название	Описание	Цена руб./день	
<input type="checkbox"/> Ежедневный оздоровляющий массаж	Будет массаж	500	Изменить
<input type="checkbox"/> Массаж	фыв	12312	Изменить

Далее

Рисунок 20 – Страница 4 этапа создания апартаментов.

6. Выбор фотографии. После того как вы выбрали услуги, вас перенаправят на заключительный этап – загрузка фотографии. У вас есть два варианта, либо ничего не загружать, просто нажав на кнопку «Загрузить фото», либо загрузить фотографию с устройства и только после этого нажать на кнопку «Загрузить фото» (Рисунок 21). После этого вас перенаправят на главную страницу, со списком всех существующих апартаментов (Рисунок 16).


АРМ администратора гостиницы
Апартаменты
Бронирования
Клиенты
Удобства
Услуги
Тарифы
Архив бронирований
Вы авторизованы как admin@gmail.com!
Выход

Загрузка фотографии

Выберите фото

Выберите файл

TestPhotoNew.jpg



Загрузить фото

Рисунок 21 – Страница 5 этапа создания апартаментов.

Добавление новой брони:

1. Что сделать новую бронь, вам нужно перейти на страницу «Бронирования», которая находится сверху, в навигационной панели и нажать на кнопку «Создать новую бронь» (Рисунок 22).

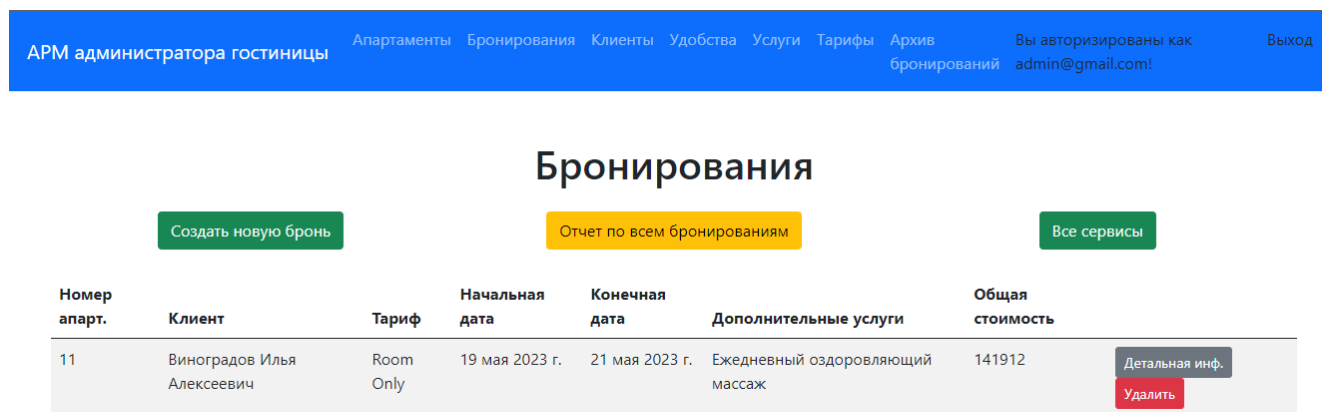


Рисунок 22 – Страница отображения бронирований.

2. Выбор клиента. Первый этап создания бронирования встречает нас с выбора клиента (Рисунок 23). На данной странице пользователь может как создать нового клиента, так и изменить уже существующего. После выполнения всех второстепенных действий, нажимаем на кнопку «Выбрать» у нужного клиента.

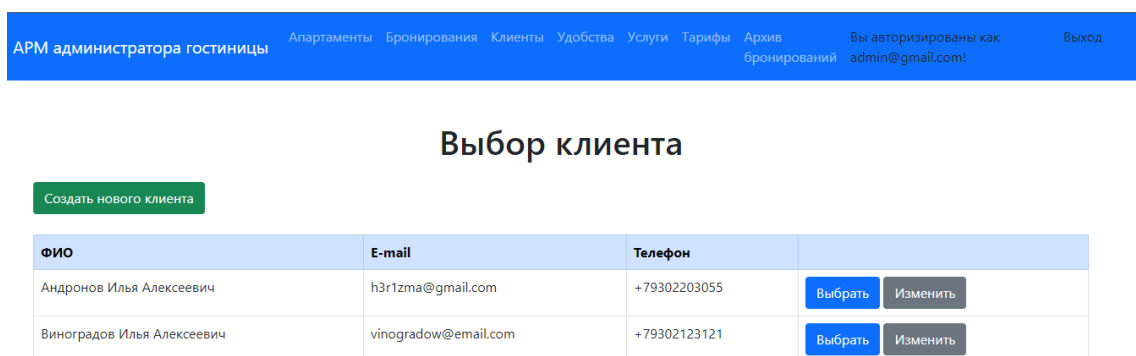


Рисунок 23 – Страница 1 этапа создания бронирования.

3. Выбор дат. После того, как вы выбрали клиента, вас переадресовали на этап с выбором начальной и конечной датами брони (Рисунок 24). От значений,

выбранных на данном и следующем этапах будут предложены апартаменты, удовлетворяющие значениям на данном и следующем этапах.

АРМ администратора гостиницы

АпартаментыБронированияКлиентыУдобстваУслугиТарифыАрхив бронирований

Вы авторизованы как admin@gmail.com!

Выход

Выбор дат

Начальная дата

17.05.2023

Конечная дата

21.05.2023

Далее

Рисунок 24 – Страница 2 этапа создания бронирования.

4. Выбор тарифа. После того, как вы выбрали даты, вас переадресовывает на страницу с выбором тарифа (Рисунок 25), где выбирается тариф, благодаря которому в следующем будут подобраны апартаменты.

АРМ администратора гостиницы

АпартаментыБронированияКлиентыУдобстваУслугиТарифыАрхив бронирований

Вы авторизованы как admin@gmail.com!

Выход

Выбор тарифа

Создать новый тариф

Название	Описание	Цена (руб/день)	
Room Only	Только комната	3000	<div>ВыбратьИзменить</div>
Half Board	Комната с завтраком	1000	<div>ВыбратьИзменить</div>

Рисунок 25 – Страница 3 этапа создания бронирования.

5. Выбор апартаментов. Предпоследним этапом является выбор апартаментов. На основе того, какие даты вы выбрали и какой тариф, вам будут предложены свободные номера на данные даты с выбранным тарифом (Рисунок 26). После того, как вы выберете нужные апартаменты, нажмите около них на кнопку «Выбрать».

Выбор апартаментов

Создать новые апартаменты

Площадь м^2	Удобства	Включенные услуги	Цена руб/день	
1	FacilityTest2 FacilityTest3		3050	Выбрать Изменить
2	12312312		3100	Выбрать Изменить
30	12312312 FacilityTest2 FacilityTest3	Ежедневный оздоровляющий массаж	4750	Выбрать Изменить
12	12312312	Ежедневный оздоровляющий массаж	3850	Выбрать Изменить
35	12312312	Массаж	10906	Выбрать Изменить
1231	FacilityTest3	Ежедневный оздоровляющий массаж	64800	Выбрать Изменить
1212			63600	Выбрать Изменить
1231		Массаж	70706	Выбрать Изменить

Рисунок 26 – Страница подтверждения удаления объекта

6. Выбор дополнительных услуг. Заключительным этапом создания бронирования служит выбор дополнительных услуг. Алгоритм схож с тем, что был при выборе услуг в создании апартаментов (Рисунок 20). После того, как выбрали/не выбрали услуги, вас перенаправит на главную страницу, где представлены все бронирования (Рисунок 22).

6.2 Руководство программиста

Данное руководство содержит описание экшенов контроллеров разработанного приложения на платформе ASP.NET MVC.

ApartamentsController.cs:

1. `public async Task<IActionResult> Index():` Этот метод используется для отображения списка апартаментов. В зависимости от роли пользователя, список может быть разный. Для ролей "admin" и "manager" отображаются все апартаменты, в то время как для клиента отображаются только апартаменты, которые он забронировал.

2. `public async Task<IActionResult> Details(int? id):` Этот метод используется для получения детальной информации об апартаменте с указанным ID. Если ID не предоставлен или апартамент с таким ID не найден, будет возвращено значение `NotFound()`. В противном случае будет возвращено представление с информацией об апартаменте.

3. `Public IActionResult Create():` данный метод имеет атрибут `[Authorize(Roles = "admin")]`, что означает, что только пользователи с ролью "admin" имеют доступ к этому методу. Он выполняет перенаправление на действие "FirstStep" контроллера "NewApartment". Этот метод используется для отображения формы создания нового объекта типа "Apartment" и перенаправления пользователя на первый шаг процесса создания нового апартамента.

4. `public IActionResult GetTariffTables():` данный метод имеет атрибут `[Authorize(Roles = "admin, manager, client")]`, что означает, что пользователи с ролями "admin", "manager" и "client" имеют доступ к этому методу. Он выполняет перенаправление на действие "Index" контроллера "Tariffs". Этот метод используется для перенаправления пользователя на страницу "Index" с информацией о тарифах.

5. `public async Task<IActionResult> Edit(int? id):` данный метод отвечает за обработку GET-запроса на редактирование апартамента. Он требует авторизации

					МИВУ.09.03.04-01.000 ПЗ	Лист
						34
Изм	Лист	№ докум.	Подп.	Дата		

для ролей "admin" и "manager". Входным параметром является id, представляющий идентификатор апартаментов для редактирования. Метод выполняет проверки на null и наличие апартаментов, и возвращает NotFound() в случае неудачи. В противном случае, метод выполняет поиск апартаментов по указанному id, создает SelectList для доступных тарифов и передает его в ViewData. Затем метод возвращает представление "Edit" с моделью апартаментов и списком тарифов для выбора.

6. public async Task<IActionResult> Edit(int id, [Bind("Id,Number,ImageUrl,Area,Price,TariffId")] Apartments apartments, IFormFile upload): данный метод обрабатывает POST-запрос на редактирование апартаментов. Он принимает идентификатор апартаментов, модель апартаментов для обновления и загружаемый файл. Метод проверяет соответствие идентификатора апартаментов, сохраняет загруженный файл на сервере, обновляет модель апартаментов и сохраняет изменения в базе данных. В конце метод перенаправляет на действие Index или возвращает представление "Edit" с моделью апартаментов для повторного редактирования.

7. public async Task<IActionResult> Delete(int? id): данный метод обрабатывает GET-запрос для удаления апартаментов. Он принимает идентификатор апартаментов и возвращает представление "Delete" с информацией об апартаментах для подтверждения удаления. Метод проверяет наличие идентификатора и соответствующего апартамента в базе данных. В случае успешного нахождения апартамента, метод возвращает представление "Delete" с информацией об апартаментах. Если апартамент не найден, метод возвращает страницу "Not Found".

8. public async Task<IActionResult> DeleteConfirmed(int id): данный метод обрабатывает POST-запрос для подтверждения удаления апартаментов. Он принимает идентификатор апартаментов и выполняет удаление из базы данных. Метод проверяет наличие апартамента с указанным идентификатором и при нахождении удаляет его из базы данных. Также метод проверяет наличие бронирования для данного апартамента и, если оно существует, удаляет его. После

					МИВУ.09.03.04-01.000 ПЗ	Лист
						35
Изм	Лист	№ докум.	Подп.	Дата		

удаления данных метод сохраняет изменения в базе данных и перенаправляет пользователя на страницу "Index".

9. `public IActionResult DeleteFacility(int facilityId, int apartmentId):` данный метод удаляет объект Facility из списка Facility, связанного с определенным апартаментом. Он принимает два параметра: `facilityId` (идентификатор удаляемого Facility) и `apartmentId` (идентификатор апартамента). В методе происходит удаление Facility из списка Facility апартамента и сохранение изменений в базе данных. Затем пользователь перенаправляется на страницу редактирования апартамента.

10. `public async Task<IActionResult> AddFacilities(int apartmentId):` этот метод асинхронно получает из базы данных все объекты Facility, которые не связаны с указанным апартаментом (объектом Apartment), и возвращает представление (View) с этим списком.

11. `public async Task<IActionResult> AddFacilities(int apartmentId, List<int> facilitiesIds):` Этот асинхронный метод добавляет выбранные объекты Facility к указанному объекту Apartment, сохраняет изменения в базе данных и перенаправляет пользователя на страницу редактирования этого апартамента.

12. `public IActionResult DeleteService(int serviceId, int apartmentId):` Метод находит в базе данных указанный объект Service и апартамент, удаляет этот объект Service из списка сервисов апартамента, сохраняет изменения и перенаправляет пользователя на страницу редактирования апартамента.

13. `public async Task<IActionResult> AddServices(int apartmentId):` Этот метод асинхронно получает из базы данных все объекты Service, которые не связаны с указанным апартаментом, и возвращает представление (View) с этим списком.

14. `public async Task<IActionResult> AddServices(int apartmentId, List<int> servicesIds):` Этот асинхронный метод добавляет выбранные объекты Service к указанному апартаменту, сохраняет изменения в базе данных и перенаправляет пользователя на страницу редактирования апартамента.

					МИВУ.09.03.04-01.000 ПЗ	Лист
						36
Изм	Лист	№ докум.	Подп.	Дата		

15. `public FileResult GetAvaibleApartamentsReport(DateTime startDate, DateTime endDate)`: Этот метод генерирует отчёт в формате Excel, который содержит список апартаментов, доступных в указанный период времени. Отчёт основывается на шаблоне, сохранённом в указанном файле, и сохраняется в новом файле. Затем метод возвращает этот файл в ответ на запрос.

BookingsController.cs:

1. `public async Task<IActionResult> Index()`: Данный метод используется для отображения списка всех активных бронирований. Если текущий пользователь является администратором или менеджером, то ему предоставляется доступ ко всем активным бронированиям. Если пользователь является клиентом, то он видит только свои активные бронирования

2. `public async Task<IActionResult> Details(int? id)`: Данный метод используется для отображения деталей бронирования по указанному идентификатору. Если идентификатор бронирования не передан или бронирования с таким идентификатором нет, то возвращает ошибку "Не найдено".

3. `public IActionResult Create()`: Данный метод используется для перенаправления к шагу создания нового бронирования для администратора или менеджера.

4. `public IActionResult GetServicesTabless()`: Данный метод используется для перенаправления к списку всех доступных услуг.

5. `public async Task<IActionResult> Delete(int? id)`: Данный метод используется для отображения представления подтверждения удаления бронирования с указанным идентификатором.

6. `public async Task<IActionResult> DeleteConfirmed(int id)`: Данный метод используется для удаления указанного бронирования из базы данных и добавления информации о бронировании в архив. Если бронирования с указанным идентификатором нет, возвращает ошибку.

7. `public FileResult GetAllBookingsReport()`: Данный метод используется для создания отчета в формате Excel о всех активных бронированиях. В отчете

					МИВУ.09.03.04-01.000 ПЗ	Лист
Изм	Лист	№ докум.	Подп.	Дата		37

указывается информация о каждом бронировании, включая номер апартаментов, ФИО клиента, название тарифа, начальную и конечную дату бронирования, список услуг и итоговую стоимость. После создания отчета он сохраняется в файл и возвращается пользователю для скачивания.

NewApartamentController.cs:

1. Метод FirstStep (GET): Этот метод возвращает представление, предназначенное для первого шага процесса выбора свойств апартамента.

2. Метод FirstStep (POST): Этот метод сохраняет номер и площадь апартамента, переданные пользователем, во временном хранилище и перенаправляет пользователя к следующему шагу процесса - выбору тарифа.

3. Метод SecondStepTariff (GET): Этот метод асинхронно получает из базы данных все тарифы и возвращает представление с этим списком.

4. Метод SecondStepTariff (POST): Этот метод сохраняет ID выбранного пользователем тарифа во временном хранилище и перенаправляет пользователя к следующему шагу процесса - выбору удобств.

5. Метод ThirdStepFacilities (GET): Этот метод асинхронно получает из базы данных все удобства и возвращает представление с этим списком.

6. Метод ThirdStepFacilities (POST): Этот метод сохраняет ID выбранных пользователем удобств во временном хранилище и перенаправляет пользователя к следующему шагу процесса - выбору услуг.

7. Метод FourthStepServices (GET): Этот метод асинхронно получает из базы данных все услуги и возвращает представление с этим списком.

8. Метод FourthStepServices (POST): Этот метод сохраняет ID выбранных пользователем услуг во временном хранилище и перенаправляет пользователя к последнему шагу процесса - выбору фотографии.

9. Метод FifthStepPhoto (GET): Этот метод возвращает представление, предназначенное для выбора фотографии апартамента.

10. Метод FifthStepPhoto (POST): Этот метод сохраняет загруженную пользователем фотографию апартамента, создает новый объект Apartments на

					МИВУ.09.03.04-01.000 ПЗ	Лист
						38
Изм	Лист	№ докум.	Подп.	Дата		

основе данных, сохраненных во временном хранилище, добавляет его в базу данных и перенаправляет пользователя на главную страницу апартаментов.

11. Методы EditTariff, CreateTariff, EditFacility, CreateFacility, EditService и CreateService: Эти методы перенаправляют пользователя на страницу редактирования или создания соответствующего объекта (тарифа, удобства или услуги), передавая ID этого объекта (для методов редактирования).

NewBookingController.cs:

1. Метод FirstStepClient (GET): Этот метод асинхронно получает список всех клиентов из базы данных и возвращает представление (View) с этим списком.

2. Метод FirstStepClient (POST): Метод принимает идентификатор клиента и сохраняет его во временное хранилище, а затем перенаправляет пользователя на следующий шаг - выбор дат.

3. Метод SecondStepDate (GET): Метод возвращает представление для выбора даты начала и окончания бронирования.

4. Метод SecondStepDate (POST): Метод принимает даты начала и окончания бронирования, сохраняет их во временное хранилище и перенаправляет пользователя на следующий шаг - выбор тарифа.

5. Метод ThirdStepTariff (GET): Метод асинхронно получает список всех тарифов из базы данных и возвращает представление с этим списком.

6. Метод ThirdStepTariff (POST): Метод принимает идентификатор тарифа, сохраняет его во временное хранилище и перенаправляет пользователя на следующий шаг - выбор апартаментов.

7. Метод FourthStepApartment (GET): Метод асинхронно получает список апартаментов, доступных для выбранного тарифа и указанных дат, и возвращает представление с этим списком.

8. Метод FourthStepApartment (POST): Метод принимает идентификатор апартамента, сохраняет его во временное хранилище и перенаправляет пользователя на следующий шаг - выбор услуг.

					МИВУ.09.03.04-01.000 ПЗ	Лист
						39
Изм	Лист	№ докум.	Подп.	Дата		

9. Метод SixthStepServices (GET): Метод асинхронно получает список всех услуг, которые не связаны с выбранным апартаментом, и возвращает представление с этим списком.

10. Метод SixthStepServices (POST): Метод принимает список идентификаторов услуг, создает новую бронь с этими услугами, сохраняет изменения в базе данных и перенаправляет пользователя на страницу бронирований.

11. Методы EditClient, CreateClient, EditTariff, CreateTariff, EditApartament, CreateApartament, EditService, и CreateService: Эти методы перенаправляют пользователя на соответствующие страницы для редактирования или создания новых экземпляров клиентов, тарифов, апартаментов и услуг.

ClientsController.cs:

1. Метод Index: Этот метод предназначен для отображения списка клиентов. Он выполняет асинхронный запрос для получения текущего пользователя, а затем проверяет, имеет ли пользователь роль "админ" или "менеджер". Если да, то метод возвращает все клиенты из контекста базы данных. Если пользователь не является администратором или менеджером, то возвращаются только клиенты, чей адрес электронной почты совпадает с адресом электронной почты текущего пользователя.

2. Метод Details: Этот метод асинхронно получает детальную информацию о конкретном клиенте по его идентификатору. Он возвращает представление с моделью, которая содержит информацию о клиенте и его активных бронированиях.

3. Метод Create: Этот метод отображает представление для создания нового клиента.

4. Метод Create (HttpPost): Этот метод принимает данные формы из представления создания нового клиента, валидирует их и, если они действительны, добавляет нового клиента в базу данных и перенаправляет пользователя обратно на страницу Index.

					МИВУ.09.03.04-01.000 ПЗ	Лист
						40
Изм	Лист	№ докум.	Подп.	Дата		

5. Метод Edit: Этот метод принимает идентификатор клиента и асинхронно находит соответствующего клиента в базе данных, а затем возвращает представление с этим клиентом для редактирования.

6. Метод Edit (HttpPost): Этот метод принимает идентификатор клиента и объект клиента из формы редактирования. Если идентификаторы совпадают и данные проходят валидацию, метод обновляет данные клиента в базе данных и перенаправляет пользователя на страницу Index.

7. Метод Delete: Этот метод принимает идентификатор клиента, асинхронно находит соответствующего клиента в базе данных и возвращает представление с этим клиентом для подтверждения удаления.

8. Метод DeleteConfirmed: Этот метод принимает идентификатор клиента и асинхронно удаляет соответствующего клиента из базы данных, включая все его бронирования, а затем перенаправляет пользователя на страницу Index.

9. Метод GetBookingsForCurrentClient: Этот метод генерирует отчет в формате Excel, содержащий все активные бронирования для определенного клиента. Он возвращает этот файл пользователю.

BookingsArchiveController.cs:

1. Метод Index: Этот метод асинхронно извлекает все записи из архива бронирований (BookingsArchive) в контексте базы данных и возвращает представление (View) с этим списком.

FacilitiesController.cs:

1. Метод Index: Этот асинхронный метод возвращает представление со списком всех объектов Facilities из базы данных, если список существует, иначе возвращает сообщение об ошибке.

2. Метод Create (GET): Этот метод возвращает представление, которое позволяет пользователю создать новый объект Facilities.

3. Метод Create (POST): Этот асинхронный метод принимает объект Facilities, проверяет его на валидность, добавляет его в базу данных и возвращает

					МИВУ.09.03.04-01.000 ПЗ	Лист
						41
Изм	Лист	№ докум.	Подп.	Дата		

пользователя на главную страницу, если объект валиден. Если объект не валиден, метод возвращает представление с этим объектом для дальнейшего редактирования.

4. Метод Edit (GET): Этот асинхронный метод ищет объект Facilities с указанным идентификатором в базе данных и возвращает представление с этим объектом для его редактирования. Если такого объекта не существует, метод возвращает ошибку 404.

5. Метод Edit (POST): Этот асинхронный метод принимает объект Facilities и его идентификатор, проверяет объект на валидность, обновляет его в базе данных и возвращает пользователя на главную страницу, если объект валиден и его идентификатор совпадает с указанным. Если объект не валиден или его идентификатор не совпадает с указанным, метод возвращает ошибку 404 или представление с этим объектом для дальнейшего редактирования.

6. Метод Delete (GET): Этот асинхронный метод ищет объект Facilities с указанным идентификатором в базе данных и возвращает представление с этим объектом и запросом на его удаление. Если такого объекта не существует, метод возвращает ошибку 404.

7. Метод DeleteConfirmed (POST): Этот асинхронный метод удаляет объект Facilities с указанным идентификатором из базы данных и возвращает пользователя на главную страницу. Если такого объекта не существует, метод возвращает ошибку.

HomeController.cs:

1. Метод Index: Этот метод возвращает главное представление (View), которое обычно служит входной точкой для пользователей при посещении сайта.

2. Метод Privacy: Этот метод возвращает представление (View), которое обычно содержит политику конфиденциальности сайта.

3. Метод Error: Этот метод возвращает представление ошибки. Он создаёт и передаёт в представление экземпляр ErrorViewModel, в котором содержится идентификатор запроса, вызвавшего ошибку. Декоратор ResponseCache

					МИВУ.09.03.04-01.000 ПЗ	Лист
						42
Изм	Лист	№ докум.	Подп.	Дата		

устанавливает параметры кеширования для этого метода: он не кеширует результат (NoStore = true), и не сохраняет его ни в каком расположении (Location = ResponseCacheLocation.None).

ServicesController.cs:

1. Метод Index: Этот асинхронный метод получает список всех объектов Services из базы данных и возвращает представление (View) с этим списком. Если сущности Services в контексте базы данных не существует, то возвращается сообщение об ошибке.

2. Метод Details: Этот асинхронный метод получает из базы данных объект Services с указанным идентификатором и возвращает представление (View) с этим объектом. Если идентификатор или сущности Services в контексте базы данных не существует, то возвращается сообщение об ошибке "NotFound".

3. Метод Create (GET): Этот метод возвращает представление (View), которое содержит форму для создания нового объекта Services.

4. Метод Create (POST): Этот асинхронный метод получает данные из формы, создаёт новый объект Services, добавляет его в базу данных и перенаправляет пользователя на страницу со списком всех объектов Services. Если полученные данные не проходят проверку, то пользователь остаётся на текущей странице с этими данными.

5. Метод Edit (GET): Этот асинхронный метод получает из базы данных объект Services с указанным идентификатором и возвращает представление (View) с этим объектом для его редактирования. Если идентификатор или сущности Services в контексте базы данных не существует, то возвращается сообщение об ошибке "NotFound".

6. Метод Edit (POST): Этот асинхронный метод получает данные из формы, обновляет существующий объект Services в базе данных и перенаправляет пользователя на страницу со списком всех объектов Services. Если идентификаторы не совпадают, или полученные данные не проходят проверку, то пользователь остаётся на текущей странице с этими данными.

					МИВУ.09.03.04-01.000 ПЗ	Лист
Изм	Лист	№ докум.	Подп.	Дата		43

7. Метод Delete (GET): Этот асинхронный метод получает из базы данных объект Services с указанным идентификатором и возвращает представление (View) с этим объектом для подтверждения его удаления. Если идентификатор или сущности Services в контексте базы данных не существует, то возвращается сообщение об ошибке "NotFound".

8. Метод DeleteConfirmed (POST): Этот асинхронный метод удаляет из базы данных объект Services с указанным идентификатором и перенаправляет пользователя на страницу со списком всех объектов Services. Если сущности Services в контексте базы данных не существует, то возвращается сообщение об ошибке.

TariffsController.cs:

1. Метод Index: Этот метод асинхронно получает все тарифы из базы данных и возвращает представление (View) со списком этих тарифов. Если набор тарифов пуст, то метод возвращает проблему с соответствующим сообщением.

2. Метод Details: Этот метод асинхронно получает из базы данных тариф с указанным идентификатором и возвращает представление (View) с данными этого тарифа. Если идентификатор или набор тарифов пуст, то метод возвращает страницу с сообщением об ошибке (NotFound).

3. Метод Create (GET): Этот метод возвращает представление (View) для создания нового тарифа.

4. Метод Create (POST): Этот асинхронный метод добавляет новый тариф в базу данных, сохраняет изменения и перенаправляет пользователя на страницу со списком всех тарифов. Если состояние модели не действительно, метод возвращает представление с тарифом, чтобы его можно было исправить.

5. Метод Edit (GET): Этот метод асинхронно получает из базы данных тариф с указанным идентификатором и возвращает представление (View) для редактирования данных этого тарифа. Если идентификатор или набор тарифов пуст, то метод возвращает страницу с сообщением об ошибке (NotFound).

					МИВУ.09.03.04-01.000 ПЗ	Лист
						44
Изм	Лист	№ докум.	Подп.	Дата		

6. Метод Edit (POST): Этот асинхронный метод обновляет данные тарифа в базе данных, сохраняет изменения и перенаправляет пользователя на страницу со списком всех тарифов. Если идентификатор в представлении не совпадает с идентификатором в базе данных, метод возвращает страницу с сообщением об ошибке (NotFound).

7. Метод Delete (GET): Этот метод асинхронно получает из базы данных тариф с указанным идентификатором и возвращает представление (View) для подтверждения удаления этого тарифа. Если идентификатор или набор тарифов пуст, то метод возвращает страницу с сообщением об ошибке (NotFound).

8. Метод DeleteConfirmed (POST): Этот асинхронный метод удаляет тариф с указанным идентификатором из базы данных, сохраняет изменения и перенаправляет пользователя на страницу со списком всех тарифов. Если набор тарифов пуст, то метод возвращает проблему с соответствующим сообщением.

Заключение

В ходе выполнения данного курсового проекта, была разработана система для автоматизации рабочего процесса администратора гостиницы, основанная на технологии ASP.NET MVC. Система построена на основе распределенной архитектуры, позволяющей разделить её функционал между клиентской и серверной сторонами. Взаимодействие с базой данных организовано при помощи Entity Framework Core, предоставляющим удобные инструменты для работы с данными.

Проект был реализован с использованием современных инструментов разработки и применением лучших практик программирования. Результатом стало функциональное веб-приложение, способное обеспечить удобный и надежный инструментарий для управления процессами бронирования и работы с клиентской базой в сфере гостиничного бизнеса. Данный проект успешно протестирован, подтвердив свою работоспособность и эффективность.

Основной итог данного курсового проекта — это полноценное веб-приложение, которое может быть реально применено для автоматизации работы администраторов и упрощения взаимодействия с клиентами в гостиничном бизнесе.

В перспективе, система может быть дополнена новыми функциями, такими как уведомления о статусах бронирований, дополнительные аналитические отчеты и усовершенствованные возможности управления пользователями и доступом. Оптимизация и улучшение пользовательского интерфейса также являются важными аспектами для дальнейшего улучшения системы.

В заключение, создание распределенной системы автоматизации рабочего места администратора гостиницы на базе ASP.NET MVC — это актуальная и перспективная задача, способная значительно улучшить эффективность управления в сфере гостиничного бизнеса.

					МИВУ.09.03.04-01.000 ПЗ	Лист
Изм	Лист	№ докум.	Подп.	Дата		46

Приложение

Ссылка на GitHub репозиторий с приложением:
<https://github.com/Muntissa/DDPS>

					МИВУ.09.03.04-01.000 ПЗ	Лист
						47
Изм	Лист	№ докум.	Подп.	Дата		