

**ОТДЕЛЕНИЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ**

**СТРУКТУРНОЕ ПОДРАЗДЕЛЕНИЕ 10**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Допустить к защите** | | |
| Заведующий отделения | | | |
|  | | Селезнева А.В. | |
|  | | | 2024 год |

**КУРСОВАЯ РАБОТА**

|  |
| --- |
| **Разработка сайта для бронирования столиков в ресторане** |
| **Специальность \_\_09.02.07 Информационные системы и программирование.**  код, название |
| **КР.09.02.07.1.31.6** |

**(шифр работы)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент: |  |  |  |  | Верзаков И.И. |
|  | (подпись) |  | (дата) |  | (Ф.И.О.) |
| Проверил преподаватель: |  |  |  |  | Емельянов С.А. |
|  | (подпись) |  | (дата) |  | (Ф.И.О.) |
| Оценка: |  |  |  |  |  |

**Москва**

2024г.

**Оглавление**

[Введение 3](#_Toc159501386)

[Глава 1. Теоретическая часть 4](#_Toc159501387)

[1.1 Виды систем бронирования 4](#_Toc159501388)

[1.2 Система уведомлений для подтверждения бронирования 22](#_Toc159501389)

[1.3 Простой и интуитивно понятный интерфейс 22](#_Toc159501390)

[1.4 Онлайн-бронирование через веб-сайт ресторана 23](#_Toc159501391)

[1.5 Технологии и инструменты для разработки 24](#_Toc159501392)

[Глава 2. Практическая часть 41](#_Toc159501393)

[2.1 Создание таблиц 41](#_Toc159501394)

[2.2 Создание запроса 43](#_Toc159501395)

[2.3 Создание формы 46](#_Toc159501396)

[2.4 Создания отчета 51](#_Toc159501397)

[Заключение 54](#_Toc159501398)

[Список литературы 56](#_Toc159501399)

# Введение

В современном обществе, где технологии меняются быстрее, чем когда-либо, ресторанный бизнес должен постоянно адаптироваться и привлекать клиентов. Онлайн-бронирование столов играет ключевую роль в этом, помогая улучшить обслуживание и оптимизировать процессы. Это не просто удобство для посетителей, а важный элемент для успешной адаптации к изменяющемуся рынку и сохранения конкурентоспособности. Исследование направлено на анализ этой роли, подчеркивая необходимость инноваций для долгосрочного успеха в будущем.

Актуальность: Актуальность исследования подчеркивается растущей потребностью в цифровых решениях для бронирования и обширными перспективами для рестораторов. Внедрение инновационных цифровых платформ становится необходимым шагом для успешного развития в условиях быстрого изменения требований рынка и технологий.

Цель работы: Разработать сайт для бронирования столиков в ресторане.

Задачи:

1. Разработать интерактивный календарь для выбора даты бронирования;
2. Разработать систему уведомлений для подтверждения бронирования стола;
3. Обеспечить онлайн-бронирование через веб-сайт ресторана;
4. Использовать HTML, CSS и JavaScript для разработки онлайн-бронирования.

# **Глава 1. Теоретическая часть**

# **1.1 Виды систем бронирования**

Предоставление пользователю интерактивного календаря.

Отображение календаря:

1. Пользователю предоставляется интерактивный календарь на экране;
2. Каждый день представлен в виде ячейки, которая обычно содержит численное обозначение дня месяца.

Навигация по календарю:

1. Пользователь может легко перемещаться между месяцами и годами, используя соответствующие элементы управления.

Функции выбора даты:

1. Пользователь может выбрать дату, кликнув на соответствующую ячейку в календаре;
2. Выбранная дата может быть визуально выделена или отображена в отдельном поле для подтверждения выбора.

Адаптивность и удобство использования:

1. Календарь должен быть адаптивным и отзывчивым, чтобы обеспечить комфортное использование на различных устройствах и экранах;
2. Элементы управления календарем должны быть достаточно крупными и понятными, чтобы пользователи могли легко взаимодействовать с ними, даже на мобильных устройствах.

Возможность быстрого переключения между месяцами и годами.

Элементы управления:

1. Пользователю предоставляются элементы управления для быстрого переключения между месяцами и годами;
2. Обычно это стрелки или кнопки для перехода к предыдущему и следующему месяцам, а также для выбора других лет.

Быстрое перемещение:

1. Пользователь может быстро перемещаться вперед и назад по месяцам, нажимая соответствующие кнопки;
2. Для быстрого выбора конкретного месяца или года может быть предусмотрен выпадающий список или всплывающее окно с календарем.

Удобство использования:

1. Элементы управления должны быть расположены таким образом, чтобы пользователи могли легко обнаружить их и использовать без лишних усилий;
2. Интерфейс должен быть интуитивно понятным, даже для пользователей с минимальным опытом работы с подобными приложениями.

Визуализация изменений:

1. При переключении между месяцами и годами календарь должен мгновенно обновляться, чтобы пользователь мог видеть изменения в реальном времени;
2. Выделение текущего месяца и года может быть изменено соответственно, чтобы пользователь всегда знал, в каком периоде он находится.

Адаптивность:

1. Элементы управления должны быть адаптивными и подходить для различных размеров экранов и устройств, включая мобильные устройства.

Реакция на действия пользователя с минимальной задержкой.

Моментальное обновление интерфейса:

1. Система обновляет интерфейс непосредственно после действий пользователя без задержек, чтобы обеспечить плавный и мгновенный отклик;
2. Это позволяет пользователям моментально видеть результаты своих действий и продолжать взаимодействие с сайтом без ожидания.

Оптимизация клиентской стороны:

1. Код JavaScript на клиентской стороне оптимизирован для выполнения вычислений и обновления интерфейса с минимальной задержкой;
2. Это обеспечивает быстрый отклик интерфейса на действия пользователя даже при низкой пропускной способности сети или медленном устройстве.

Мониторинг и оптимизация производительности:

1. Производительность системы регулярно мониторится, и проводятся оптимизации для улучшения времени отклика и обеспечения плавного взаимодействия с пользователем;

Интуитивно понятный интерфейс с возможностью быстрого возврата к предыдущим этапам выбора.

Навигационная структура:

1. Интерфейс разработан с учетом логической навигационной структуры, позволяющей пользователям легко перемещаться между различными этапами выбора;
2. Это включает в себя четкое разделение на различные шаги бронирования и наличие навигационных элементов для перехода между ними.

Понятные иконки и подписи:

1. Интерфейс содержит понятные иконки и подписи, которые помогают пользователям понять функционал каждого элемента и совершать действия с уверенностью.

Тестирование пользовательского опыта:

1. Перед выпуском системы проводится тестирование пользовательского опыта для убедительности в том, что интерфейс интуитивно понятен и обеспечивает удобство возвращения к предыдущим этапам выбора.

# **1.2 Система уведомлений для подтверждения бронирования**

Система уведомлений для подтверждения бронирования предоставляет механизм, который информирует пользователя о статусе его бронирования. После успешного оформления бронирования пользователю автоматически отправляется уведомление, подтверждающее получение запроса на бронирование. Это уведомление может содержать детали бронирования, такие как выбранная дата, время и количество гостей, а также информацию о дальнейших шагах, которые могут потребоваться от пользователя, например, подтверждение по электронной почте или SMS. Система также может предоставлять дополнительные уведомления, например, напоминания о предстоящем бронировании или уведомления об изменениях в статусе бронирования. Этот процесс уведомлений помогает пользователям чувствовать себя уверенно в своих бронированиях и предоставляет им информацию о статусе и дальнейших действиях, если таковые требуются.

# **1.3 Простой и интуитивно понятный интерфейс**

Простой и интуитивно понятный интерфейс разработан с учетом потребностей пользователей и направлен на максимальное удобство взаимодействия. Он предоставляет пользователю понятную и легко воспринимаемую структуру, что делает процесс бронирования столов в ресторане незамысловатым и удобным. Ключевые принципы такого интерфейса включают в себя интуитивную навигацию, понятные и понятные иконки и подписи, а также простоту в использовании без необходимости дополнительного обучения. Каждый этап процесса бронирования представлен ясно и доступно, чтобы пользователи могли легко совершать необходимые действия без запинок или затруднений. Такой подход к разработке интерфейса способствует удовлетворенности пользователей и повышает их уверенность в использовании системы бронирования столов в ресторане.

# **1.4 Онлайн-бронирование через веб-сайт ресторана**

Онлайн-бронирование через веб-сайт ресторана представляет собой эффективный и удобный способ для клиентов сделать заказ, а также для ресторана контролировать поток посетителей.

Плюсы онлайн-бронирования включают в себя:

1. Удобство и доступность: Клиенты могут забронировать столик в любое удобное для них время, не выходя из дома или офиса. Это особенно удобно для тех, кто имеет ограниченное свободное время или предпочитает планировать заранее;
2. Экономия времени и усилий: Клиентам не нужно звонить в ресторан или посещать его лично для совершения бронирования. Это экономит их время и упрощает процесс заказа;
3. Улучшение управления потоком клиентов: Для ресторанов онлайн-бронирование помогает эффективно планировать рабочий день, распределять ресурсы и избегать перегрузок или пустых периодов.

Однако есть и некоторые потенциальные недостатки:

1. Технические проблемы: Возможны сбои в работе веб-сайта или проблемы с онлайн-платежами, что может привести к недовольству клиентов и потере заказов;
2. Неудовлетворенность клиентов: Некоторые клиенты могут предпочитать более традиционные способы бронирования или испытывать неуверенность в безопасности передачи личной информации онлайн;

# **1.5 Технологии и инструменты для разработки**

HTML.

HTML является основой для веб-разработки и играет ключевую роль в создании веб-страниц. Вот почему:

1. Необходимый язык: HTML является обязательным языком для создания веб-страниц. Это основа, на которой строится вся структура и содержимое страницы;
2. Интерпретация браузером: Браузеры интерпретируют HTML-код и отображают его содержимое на экране пользователя. Без HTML страницы просто не могут быть отображены в веб-браузере;
3. Структурирование контента: HTML позволяет структурировать контент на веб-странице, делая его более организованным и понятным для пользователей;
4. Поддержка мультимедиа: С помощью HTML можно встраивать изображения, видео, аудио и другие мультимедийные элементы на веб-страницы, делая их более интерактивными и привлекательными для посетителей;
5. Основа для других технологий: HTML часто используется в сочетании с CSS для стилизации страниц и JavaScript для добавления интерактивности. Он также используется вместе с другими технологиями, такими как PHP, Python и Ruby, для создания динамических веб-сайтов.

Таким образом, HTML является неотъемлемой частью веб-разработки и является основой для создания любого веб-сайта. Без него невозможно представить себе современный интернет.

HTML определяет структуру веб-страницы с помощью различных элементов, каждый из которых имеет свою уникальную роль в организации и представлении информации. Вот более подробное описание:

1. Заголовки (Headings): Заголовки (<h1>, <h2>, <h3>, <h4>, <h5>, <h6>`) используются для выделения основных разделов страницы. `<h1> обычно используется для самого важного заголовка, а <h6> - для менее значимых;
2. Параграфы (Paragraphs): Параграфы (<p>) используются для организации текста в отдельные абзацы. Они делают текст более читаемым и удобным для восприятия;
3. Списки (Lists): HTML поддерживает два типа списков: ненумерованные (<ul>) и нумерованные (<ol>). Ненумерованные списки представляют элементы без порядковых номеров, а нумерованные - с порядковыми номерами;
4. Таблицы (Tables): Таблицы (<table>) используются для представления структурированных данных в виде строк и столбцов. Они часто используются для отображения табличных данных, таких как расписание или списки товаров;
5. Формы (Forms): Формы (<form>) позволяют пользователям отправлять данные на сервер, например, при регистрации или отправке комментария. Они могут содержать различные элементы ввода, такие как текстовые поля, флажки, кнопки и т. д.;
6. Изображения (Images): Изображения (<img>) используются для вставки графических файлов на страницу. Они могут быть важным элементом дизайна и помогают визуализировать информацию.

Атрибуты элементов HTML - это дополнительная информация, которая прикрепляется к каждому элементу и позволяет дополнительно настраивать их поведение или внешний вид. Вот некоторые распространенные примеры атрибутов и их использование:

1. `src` (source): Этот атрибут используется для указания пути к файлу изображения, видео или звука, который должен быть отображен или воспроизведен в элементе. Например, <img src="image.jpg">;
2. `alt` (alternative text): Этот атрибут предоставляет текстовое описание изображения для пользователей, которые не могут просматривать изображения из-за ограничений своих возможностей или настроек браузера. Например, <img src="image.jpg" alt="Описание изображения">
3. `href` (hypertext reference): Этот атрибут используется для указания URL-адреса, на который должна вести ссылка. Например, <a href="https://example.com">Ссылка</a>;
4. `title`: Этот атрибут предоставляет всплывающую подсказку или описание элемента, которое отображается при наведении курсора мыши на элемент. Например, <img src="image.jpg" alt="Описание изображения" title="Дополнительная информация">;
5. `class`: Этот атрибут используется для определения класса элемента, который может быть использован для стилизации элемента с помощью CSS или для идентификации элемента при помощи JavaScript. Например, <div class="container">;
6. `id`: Этот атрибут используется для определения уникального идентификатора элемента, который может быть использован для прямого доступа к элементу с помощью CSS или JavaScript. Например, <div id="header">.

CSS.

Структурирование стилей в CSS предоставляет разработчикам мощный инструмент для создания единообразного и привлекательного дизайна веб-страниц. Вот более подробное описание этого процесса:

1. Создание стилей для элементов HTML: CSS позволяет создавать стили, которые могут быть применены к различным элементам HTML на странице. Это включает в себя возможность изменения цвета текста, фона, размера шрифта, отступов, рамок и других аспектов внешнего вида элементов;
2. Использование селекторов: CSS использует селекторы для выбора конкретных элементов, к которым будут применены стили. Это может быть класс элемента, идентификатор, тип элемента или его состояние (например, :hover для стилизации при наведении мыши);
3. Каскадность стилей: CSS работает по принципу каскадности, что означает, что стили могут быть применены не только к самому элементу, но и к его дочерним элементам. Это позволяет создавать сложные иерархии стилей и эффективно управлять внешним видом всей страницы;
4. Применение единообразного дизайна: Благодаря CSS разработчики могут легко обеспечить единообразный и согласованный дизайн страницы, применяя одни и те же стили к различным элементам. Это повышает удобство использования для пользователей и улучшает визуальный облик веб-страницы;
5. Разделение структуры и стиля: CSS позволяет разделить структуру HTML-документа и его внешний вид, что делает код более чистым, организованным и легким для обслуживания. Это позволяет разработчикам изменять дизайн страницы без необходимости внесения изменений в сам HTML-код.

Структурирование стилей в CSS является важным этапом в создании качественных веб-страниц и помогает разработчикам создавать современные и привлекательные веб-сайты.

Простота использования CSS обеспечивается несколькими ключевыми аспектами, которые делают его доступным для разработчиков на всех уровнях опыта:

1. Простой синтаксис: CSS использует простой и понятный синтаксис, основанный на объявлении свойств и их значений для элементов HTML. Например, чтобы изменить цвет текста, достаточно написать color: red;;
2. Интуитивная логика: Основные концепции и правила CSS легко усваиваются благодаря интуитивной логике, которая определяет, как стили применяются к элементам на веб-странице;
3. Обширные ресурсы: Существует множество обучающих материалов, учебников и онлайн-курсов, посвященных CSS, что делает его изучение и использование еще более доступным и простым для новичков;
4. Превосходная документация: Официальная документация CSS и множество ресурсов сообщества предоставляют обширную информацию о каждом свойстве и функции CSS, что делает его использование более прозрачным и понятным;
5. Инструменты разработчика: Современные интегрированные среды разработки (IDE) и редакторы кода предоставляют различные инструменты и подсказки, которые упрощают написание и отладку CSS кода.

Благодаря этим аспектам CSS становится доступным даже для тех, кто только начинает свой путь в веб-разработке, и предоставляет им мощный инструмент для создания стильных и привлекательных веб-страниц.

CSS обладает разнообразием возможностей, которые позволяют стилизовать элементы веб-страниц и создавать эффектные и креативные дизайны. Вот некоторые из наиболее распространенных возможностей CSS:

1. Изменение цвета текста и фона: CSS позволяет легко изменять цвет текста и фона элементов, что позволяет создавать контрастные и привлекательные комбинации;
2. Настройка шрифтов и размеров: CSS предоставляет возможность выбора шрифта, его размера, начертания и толщины, что позволяет создавать разнообразные стили текста;
3. Управление отступами и интервалами: CSS позволяет задавать отступы и интервалы между элементами, что влияет на их визуальное расположение и оформление;
4. Добавление рамок и теней: CSS позволяет создавать рамки вокруг элементов и добавлять тени, что придает элементам глубину и объем;
5. Анимации и переходы: CSS позволяет создавать анимации и переходы между состояниями элементов, что добавляет интерактивности и динамичности веб-страницам;
6. Форматирование фоновых изображений: CSS позволяет добавлять фоновые изображения элементам и настраивать их положение, повторение и масштабирование;
7. Медиа-запросы для адаптивного дизайна: CSS поддерживает медиа-запросы, которые позволяют создавать адаптивные стили, меняющиеся в зависимости от размера экрана устройства пользователя;
8. Flexbox и Grid Layout: CSS предоставляет мощные инструменты для создания гибких макетов с помощью Flexbox и Grid Layout, что обеспечивает легкость в создании сложных структур страницы.

Эти возможности CSS позволяют веб-разработчикам создавать разнообразные и привлекательные дизайны веб-страниц, что делает их более функциональными и привлекательными для пользователей.

Принцип каскадности и наследование стилей в CSS обеспечивают гибкость и эффективное управление внешним видом веб-страниц. Вот как это работает:

1. Каскадность стилей: CSS позволяет определить несколько стилей для одного элемента, причем каждый стиль имеет свой приоритет. Если несколько стилей конфликтуют между собой, применяется правило каскадности, определяющее, какой стиль будет использоваться. Это позволяет создавать более гибкие и сложные дизайны, а также управлять приоритетом стилей в зависимости от их специфичности и применимости;
2. Наследование стилей: Некоторые стили, определенные для родительского элемента, могут быть автоматически унаследованы его дочерними элементами. Это означает, что если стиль не был явно определен для дочернего элемента, он может унаследовать стиль от своего родителя. Наследование стилей упрощает и ускоряет процесс стилизации веб-страниц, поскольку многие элементы могут использовать общие стили без необходимости повторного их определения;
3. Приоритет стилей: В CSS определены правила приоритета, которые определяют, какой стиль будет применен к элементу в случае конфликта между несколькими стилями. Это правило учитывает различные факторы, такие как специфичность селектора, порядок определения стилей и важность стиля (например, встроенные стили имеют больший приоритет, чем внешние таблицы стилей);
4. Эффективное управление стилями: Благодаря принципу каскадности и наследованию стилей разработчики могут эффективно управлять стилями на странице, минимизируя повторение кода и обеспечивая единообразие дизайна. Это позволяет создавать структурированный и легко поддерживаемый CSS-код.

Таким образом, принцип каскадности и наследование стилей в CSS играют ключевую роль в создании гибких и эффективных стилей для веб-страниц, обеспечивая их согласованность и профессиональный внешний вид.

Респонсивный дизайн в CSS представляет собой способность создавать веб-страницы, которые могут адаптироваться к различным размерам экранов и устройств, обеспечивая оптимальный пользовательский опыт. Вот ключевые аспекты респонсивного дизайна:

1. Медиа-запросы: CSS предоставляет механизм медиа-запросов, который позволяет задавать разные стили в зависимости от характеристик устройства, таких как ширина экрана, ориентация и разрешение. Это позволяет создавать адаптивные макеты, которые оптимизированы для просмотра на различных устройствах;
2. Гибкие единицы измерения: Использование относительных единиц измерения, таких как проценты, em и rem, позволяет создавать макеты, которые масштабируются и адаптируются к изменениям размеров экрана без необходимости изменения кода;
3. Flexbox и Grid Layout: Модули Flexbox и Grid Layout в CSS предоставляют мощные инструменты для создания гибких и адаптивных макетов, которые легко адаптируются к различным размерам экранов и устройств;
4. Изображения и медиа: CSS позволяет создавать адаптивные изображения и медиа-файлы, которые автоматически масштабируются и подстраиваются под размеры экрана, обеспечивая оптимальное качество и быстрое загрузку на различных устройствах;
5. Touch-функционал: CSS позволяет создавать стили, которые оптимизированы для использования на сенсорных устройствах, обеспечивая удобство навигации и взаимодействия для пользователей смартфонов и планшетов.

Респонсивный дизайн в CSS играет ключевую роль в обеспечении приятного и удобного пользовательского опыта на всех устройствах, от мобильных устройств до настольных компьютеров. Это помогает улучшить доступность, удобство использования и уровень удовлетворенности пользователей веб-сайтом.

JavaScript.

JavaScript играет ключевую роль в разработке современных веб-приложений, обеспечивая интерактивность, динамичность и удобство использования для пользователей.

Интерактивность является одним из ключевых аспектов JavaScript, который открывает широкие возможности для создания динамичного и увлекательного пользовательского опыта на веб-страницах. Вот несколько примеров того, как JavaScript обеспечивает интерактивность на веб-страницах:

1. Формы: JavaScript позволяет добавлять проверку данных введенных в формы, динамическое изменение содержимого формы в зависимости от действий пользователя, а также подсветку ошибок или успешной отправки формы без перезагрузки страницы;
2. Кнопки и ссылки: JavaScript может назначать действия на нажатие кнопок и ссылок, такие как переход на другую страницу, отправка формы, открытие модального окна или выполнение других действий без перезагрузки страницы;
3. Меню и навигация: JavaScript используется для создания интерактивных меню и навигационных элементов, которые могут разворачиваться, скрываться или изменяться в зависимости от действий пользователя, обеспечивая удобную навигацию по сайту;
4. Слайдеры и карусели: JavaScript позволяет создавать динамические слайдеры и карусели, которые могут автоматически пролистываться или реагировать на пользовательские действия, такие как щелчки мыши или смахивания на сенсорных устройствах;
5. Динамическое обновление содержимого: JavaScript может динамически изменять содержимое веб-страницы в ответ на действия пользователя, например, загрузка новых данных с сервера без перезагрузки страницы или изменение отображаемого контента в зависимости от выбранных параметров;
6. Анимации и эффекты: JavaScript используется для создания анимаций и специальных эффектов, таких как плавное появление элементов, анимация перемещения и изменения размеров, а также специальные эффекты при наведении курсора.

Все эти возможности JavaScript позволяют разработчикам создавать интерактивные и привлекательные веб-страницы, которые обогащают пользовательский опыт и делают взаимодействие с контентом более удобным и увлекательным.

Динамическое обновление содержимого - это одна из важных возможностей JavaScript, которая позволяет веб-страницам быть более реактивными и актуальными без перезагрузки. Вот некоторые примеры использования этой функциональности:

1. Динамические списки: JavaScript позволяет добавлять, удалять или изменять элементы списка на веб-странице в реальном времени в зависимости от пользовательских действий или внешних событий. Например, это может быть использовано для добавления новых элементов в список комментариев или обновления списка товаров на странице интернет-магазина без перезагрузки;
2. Обновление данных в реальном времени: JavaScript может использоваться для автоматического обновления данных на веб-странице в реальном времени без необходимости обновления всей страницы. Например, это может быть использовано для отображения актуальной информации о ценах акций на финансовом сайте или для отображения новых сообщений в чате без перезагрузки страницы;
3. Интерактивные формы: JavaScript позволяет динамически изменять содержимое форм на веб-странице в зависимости от введенных пользователем данных или выбранных параметров. Например, это может быть использовано для автоматического отображения дополнительных полей в форме в зависимости от выбранного типа товара или для динамической валидации данных, введенных пользователем;
4. Автоматическое обновление контента: JavaScript может использоваться для создания механизмов автоматического обновления содержимого на веб-странице через определенные промежутки времени или в ответ на определенные события. Например, это может быть использовано для автоматического обновления новостной ленты или для отображения актуальной информации о погоде.

Динамическое обновление содержимого с помощью JavaScript позволяет создавать более интерактивные и удобные для пользователя веб-страницы, обеспечивая актуальность и реактивность контента без необходимости перезагрузки всей страницы.

Обработка событий является важной возможностью JavaScript, позволяющей создавать интерактивные элементы на веб-страницах, которые реагируют на действия пользователя. Вот как JavaScript обрабатывает различные события:

1. Щелчки мыши: JavaScript позволяет привязывать функции к событию клика мыши на определенном элементе веб-страницы. Это позволяет создавать интерактивные кнопки, ссылки и элементы управления, которые реагируют на щелчок пользователя;
2. Нажатия клавиш: JavaScript позволяет отслеживать нажатия клавиш на клавиатуре и реагировать на них. Это может быть использовано для создания интерактивных форм, игр или других приложений, где пользователь может взаимодействовать с помощью клавиш;
3. Наведение курсора: JavaScript позволяет отслеживать события наведения курсора мыши на элементы веб-страницы. Это может быть использовано для создания эффектов всплывающих подсказок, изменения стилей элементов при наведении или отображения дополнительной информации;
4. Фокус и потеря фокуса: JavaScript позволяет отслеживать события фокуса и потери фокуса элементов формы, таких как текстовые поля или кнопки. Это может быть использовано для проверки введенных данных, активации кнопок или других действий при взаимодействии с формой;
5. Перетаскивание и бросание: JavaScript поддерживает события перетаскивания и бросания элементов на веб-странице. Это может быть использовано для создания интерактивных игр, элементов управления или приложений, где пользователь может перетаскивать и изменять расположение элементов;
6. События загрузки и выгрузки страницы: JavaScript позволяет отслеживать события загрузки и выгрузки страницы, что позволяет выполнять дополнительные действия при загрузке или выгрузке контента на веб-странице, например, выполнение анимации загрузки или инициализация скриптов.

Обработка событий в JavaScript является мощным инструментом для создания интерактивных и динамических веб-приложений, которые обеспечивают более удобный и эффективный пользовательский опыт.

Манипуляция DOM (Document Object Model) - это одна из ключевых возможностей JavaScript, которая позволяет разработчикам динамически изменять содержимое и структуру веб-страницы. Вот как JavaScript используется для манипуляции DOM:

1. Добавление элементов: JavaScript позволяет создавать новые элементы HTML и добавлять их в существующую структуру страницы. Например, новый элемент может быть создан с помощью document.createElement() и добавлен в DOM с помощью методов appendChild() или insertBefore();
2. Удаление элементов: JavaScript также позволяет удалять существующие элементы из DOM. Это может быть сделано с помощью метода removeChild(), который удаляет указанный элемент из его родительского элемента;
3. Изменение содержимого элементов: JavaScript позволяет изменять текстовое содержимое элементов, а также их атрибуты. Например, текст элемента может быть изменен с помощью свойства innerText или textContent, а атрибуты могут быть изменены с помощью свойства setAttribute();
4. Изменение стилей элементов: JavaScript позволяет изменять стили элементов на странице. Это может быть сделано с помощью свойства style, которое предоставляет доступ к CSS-свойствам элемента. Например, цвет текста или фона элемента может быть изменен непосредственно через JavaScript;
5. Обработка событий: JavaScript позволяет назначать обработчики событий на элементы страницы, что позволяет реагировать на различные действия пользователя, такие как клики мыши, наведение курсора и другие.

Манипуляция DOM является мощным инструментом, который позволяет создавать динамические и интерактивные веб-приложения, обогащая пользовательский опыт и делая веб-страницы более динамичными и удобными для использования.

Валидация форм является важной частью веб-разработки, и JavaScript позволяет реализовать ее на веб-страницах. Вот как JavaScript используется для валидации данных в формах:

1. Проверка обязательных полей: JavaScript может проверять, заполнены ли все обязательные поля в форме перед их отправкой на сервер. Это может быть достигнуто путем проверки наличия данных в соответствующих полях формы;
2. Проверка формата данных: JavaScript может проверять формат введенных данных, таких как электронная почта, номер телефона, дата и другие. Например, для проверки электронной почты можно использовать регулярные выражения для соответствия шаблону адреса электронной почты;
3. Предупреждение об ошибках: JavaScript может отображать сообщения об ошибках пользователю в случае неправильного заполнения формы. Это помогает пользователям понять, какие поля требуют исправления, чтобы форма могла быть успешно отправлена;
4. Проверка данных на стороне клиента: JavaScript выполняет валидацию данных на стороне клиента до их отправки на сервер. Это позволяет предотвратить отправку некорректных данных на сервер и уменьшить нагрузку на сеть;
5. Улучшение пользовательского опыта: Валидация форм с помощью JavaScript способствует повышению пользовательского опыта, предотвращая отправку некорректных данных и обеспечивая быструю обратную связь пользователю о статусе введенных данных;
6. Улучшение безопасности данных: Правильная валидация форм помогает защитить веб-приложение от нежелательного ввода данных и предотвращает возможные атаки, такие как инъекции SQL или взлом формы.

Все эти аспекты валидации форм делают использование JavaScript в этой области необходимым для обеспечения правильного и безопасного функционирования веб-приложений.

Модульность и повторное использование кода являются важными концепциями в разработке на JavaScript. Вот как JavaScript поддерживает эти концепции:

1. Модульная структура кода: JavaScript позволяет разбивать код на отдельные модули, каждый из которых выполняет определенную функцию или решает конкретную задачу. Это делает код более организованным и понятным, упрощает его поддержку и развитие;
2. Импорт и экспорт модулей: JavaScript предоставляет возможность импортировать и экспортировать функции, переменные и другие ресурсы между модулями. Это позволяет использовать код из одного модуля в другом, повторно используя его функциональность без необходимости дублирования кода;
3. Локальная область видимости: JavaScript модули имеют локальную область видимости, что означает, что переменные и функции, определенные внутри модуля, не являются доступными за его пределами. Это помогает избежать конфликтов имен и повышает безопасность исходного кода;
4. Улучшение поддерживаемости и расширяемости: Модульная структура кода делает его более поддерживаемым и расширяемым. Изменение или добавление новой функциональности в приложение становится проще, так как каждый модуль представляет собой независимую единицу, которую можно легко изменить или заменить;
5. Повторное использование кода: Модульность позволяет повторно использовать код в различных частях приложения или даже в различных приложениях. Это сокращает время разработки, уменьшает количество ошибок и повышает эффективность разработчика.

Благодаря модульной структуре и возможности повторного использования кода JavaScript позволяет создавать более гибкие, масштабируемые и поддерживаемые веб-приложения.

# Глава 2. Практическая часть

# 2.1 Создание таблиц

Для создания таблицы, нужно нажать на вкладку «Создание» и выбрать «Таблица»:

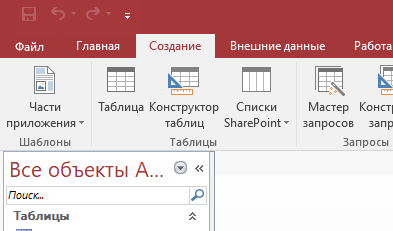


Рисунок 6 – Создание

В режиме конструктора задаем «Имя поля» и выбираем требуемый

«Тип данных»:

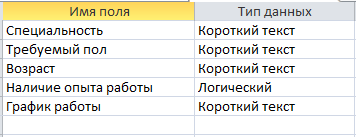


Рисунок 7 – Имя поля и Тип данных

Выделяем первую строку. На панели управления выбираем кнопку

«Ключевое поле»:

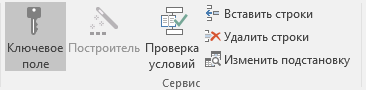


Рисунок 8 – Ключевое поле

Теперь переходим в режим таблицы. Нажимаем рисунок в левом верхнем углу выбираем режим таблицы. Сохраняем таблицу под название «Вакансии на работу».

Заполняем строки:

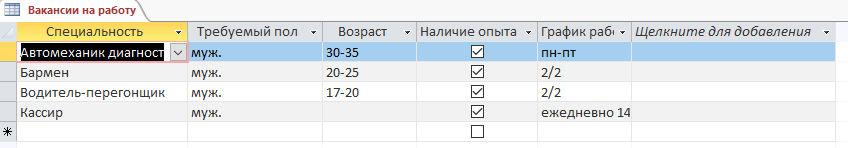


Рисунок 9 – Заполнение

Аналогично создать остальные таблицы.

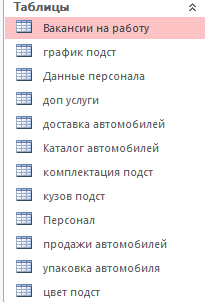


Рисунок 10 – Таблицы

Меж табличная связь для создание запросов и формы.

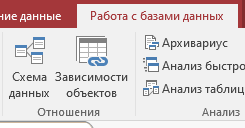


Рисунок 11 – Создание связи

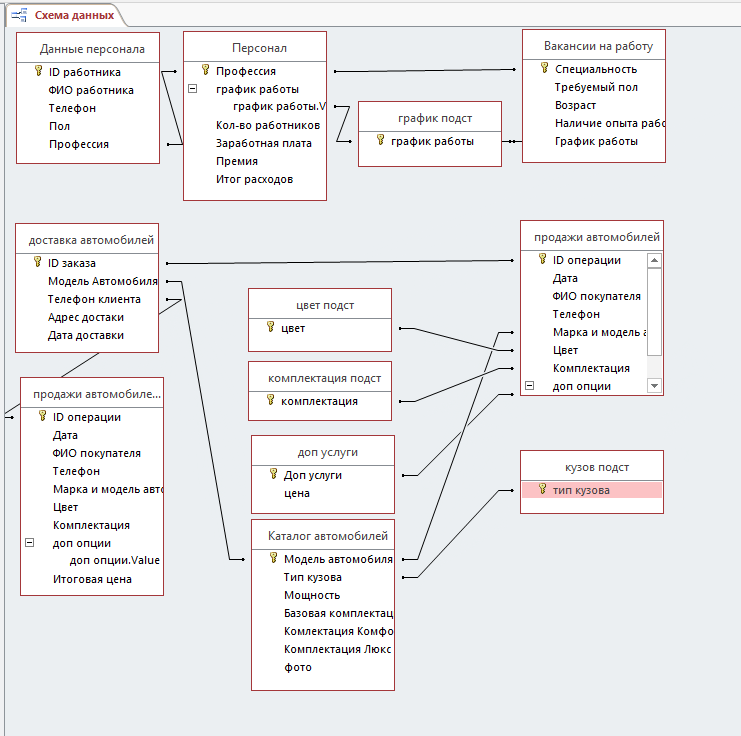


Рисунок 12 – Схема данных

# 2.2 Создание запроса

Запросы создаются через «Мастер запросов»

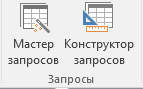


Рисунок 13 – Создание запросов

Откроется «Новый запрос». Выбираем «Простой запрос. На вкладке «Таблицы и запросы» выбираем таблицу, которая нам нужна. На вкладке «Доступные поля» нужно выбрать все поля, нажимаем на кнопку с двумя стрелочками:

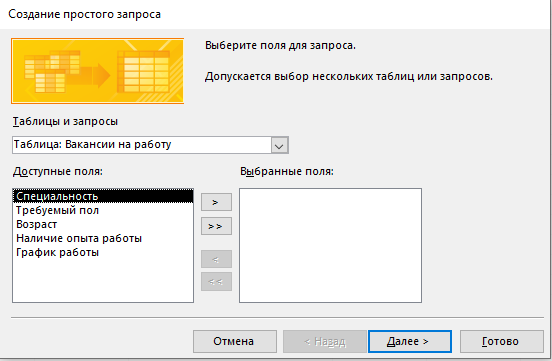


Рисунок 14 – Создание простого запроса

Затем два раза нажимаем «Далее», затем даем имя запросу «Вакансии на работу»:

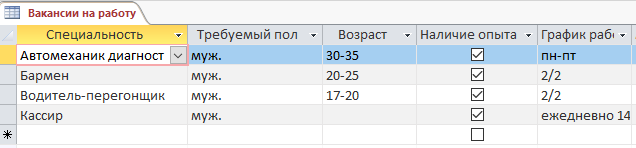


Рисунок 15 – Вакансии на работу

Переключаем в режим конструктора. В строке «Сортировка», для поля «ФИО» выбираем сортировку «по возрастанию»:

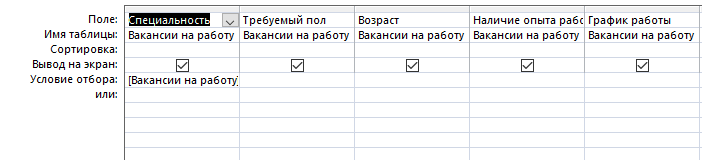


Рисунок 16 – Сортировка

То чтоб сформировать требование в базе 2-ух либо некоторых таблиц, в таком случае творим требование вместе с поддержкой «Конструктор запросов» (См. Злак. 7). Раскроется окошко «Добавление таблицы». Парным щелчком жмем в таблицы «Персонал» а также «Вакансия на работу» а также жмем Прикрыть. Далее в строчке «Поле:» подбираем необходимые нам степь:

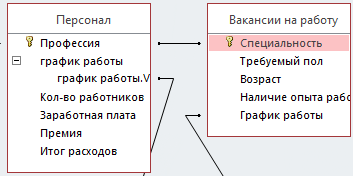


Рисунок 17 – «Персонал и Вакансии на работу»

Во вкладке «Конструктор» нажимаем выполнить:

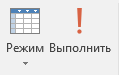


Рисунок 18 – Выполнить

Запрос готов:

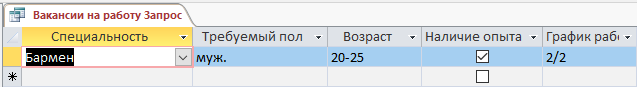


Рисунок 19 – Сортировка вакансии

Создаем запрос, который будем искать автомобиль по марке. Создаем запрос на основе таблицы «Каталог автомобилей». В режиме конструктора, в условие отбора, для поля «Модель автомобиля» пишем: Like [Введите Марку автомобиля:] & "\*". Выполняем. Появляется окно, набираем марку автомобиля:



Рисунок 20 – Сортировка автомобилей

# 2.3 Создание формы

Для того чтобы создать форму, надо на панелье управления перейти на вкладку «Создать», в поле «Другие формы» выбираем «Модальное диалоговое окно»:

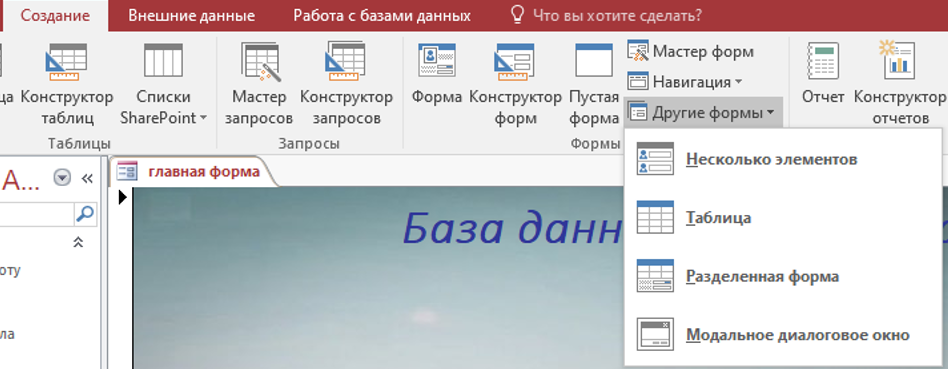


Рисунок 21 – Создание формы

Автоматически открывается форма, на которую мы будем добавлять кнопки и изображения:

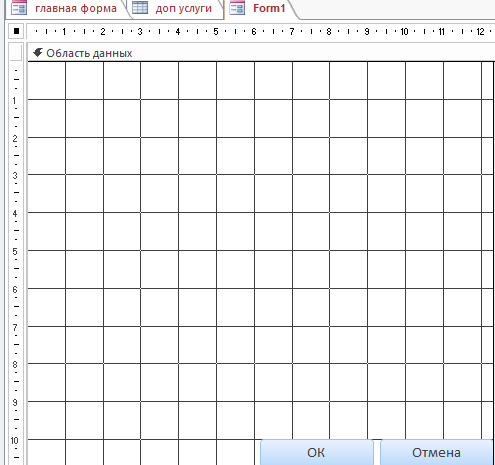


Рисунок 22 – Создание главной формы

Добавляем кнопки:

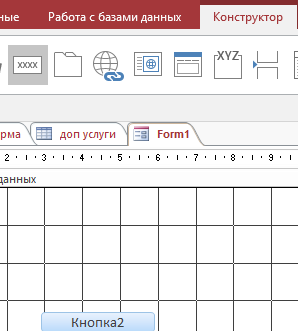


Рисунок 23 – Добавление кнопки

Меняем параметры кнопки:

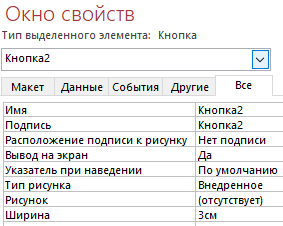


Рисунок 24 – Параметры кнопки

Добавляем действия для кнопки:

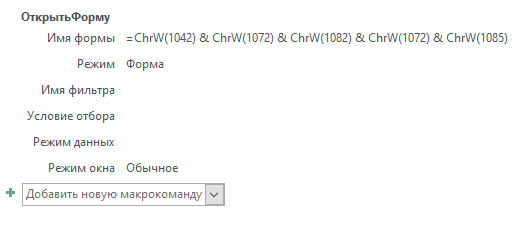


Рисунок 25 – Параметры кнопки

С остальными кнопками аналогично:

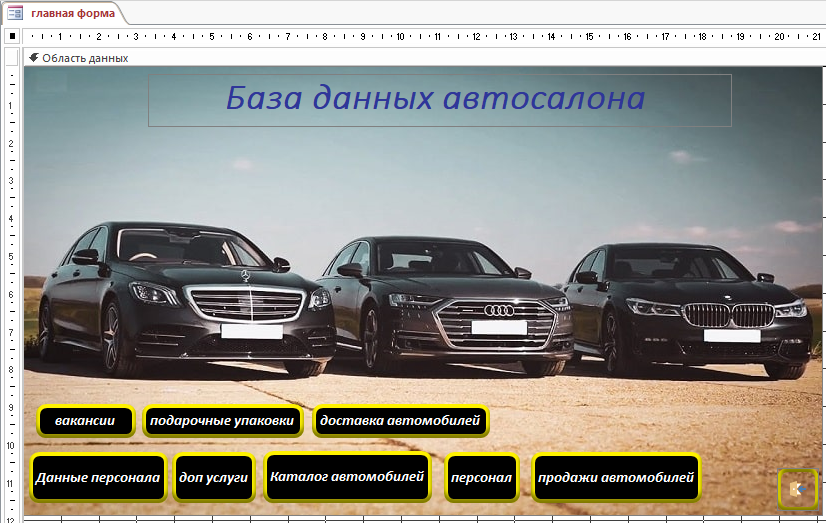


Рисунок 26 – Добавление кнопок

Для того чтобы добавить фон надо открыть «Конструктор» выбрать на понелье «Рисунок», потом мышкой кликаем на свободное место в «Форме» и открывается папка в которой выбираем нужный рисунок:

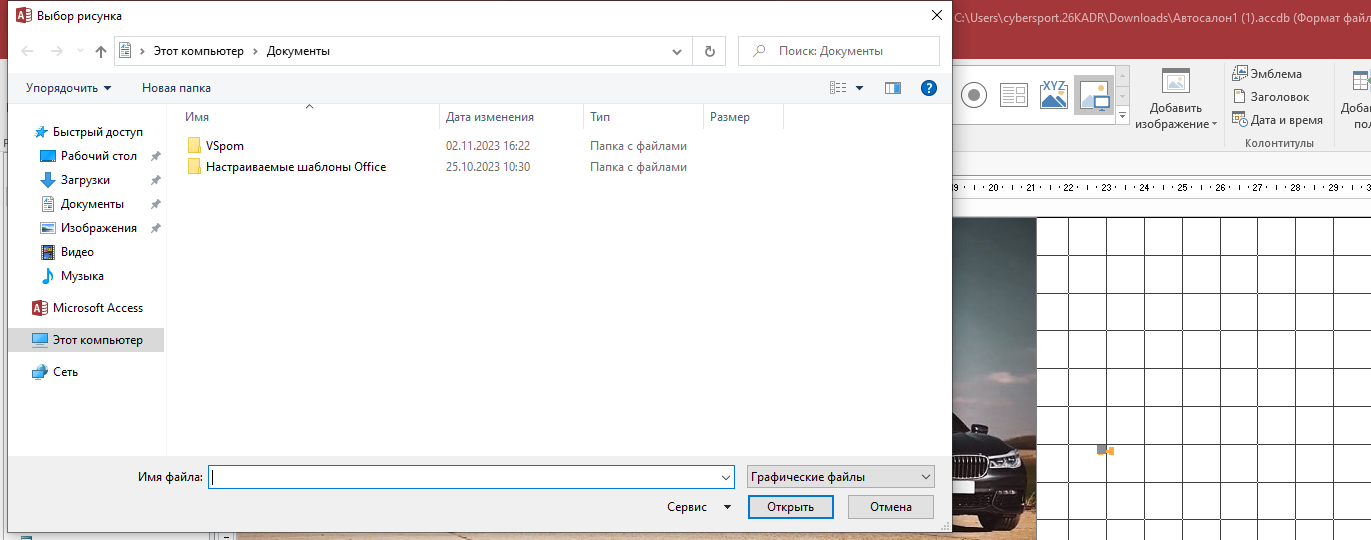


Рисунок 27 – Добавление фона

Для того чтобы создать форму выбираем таблицу «Каталог автомобилей» и в формах выбираем «Другие формы» находим и выбираем «Несколько элементов»:

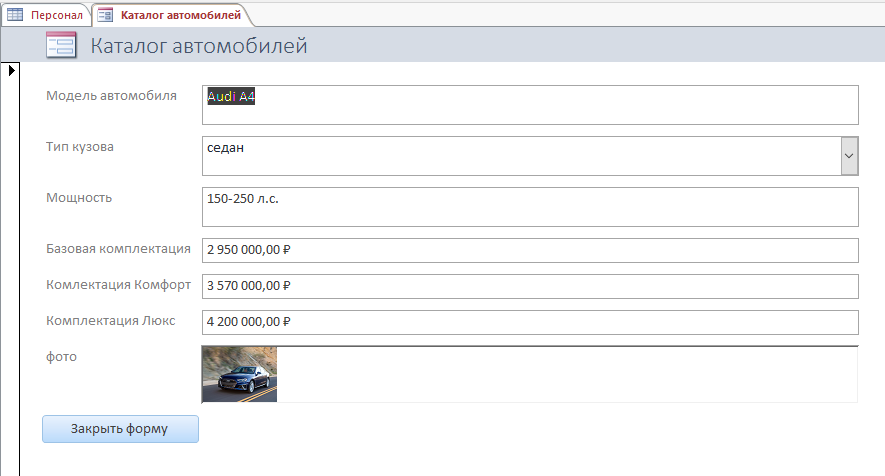


Рисунок 28 – Создание формы

Так же создаются остальные формы:

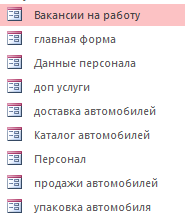


Рисунок 29 – Готовые формы

# 2.4 Создания отчета

Для того чтобы создать отчет, выбираем таблицу с которой будем работать и во вкладке «Создать» найти «Отчеты»

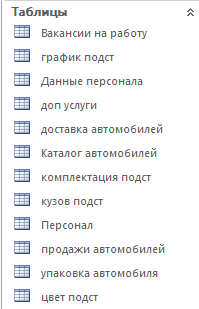


Рисунок 30 – Таблицы

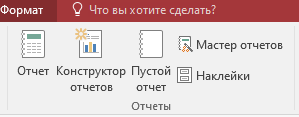


Рисунок 31 – Создание отчета

Отчет создается с помощью «Мастера отчетов». Создаём отчет на базе таблицы «Каталог автомобилей», находим в «Отчетах» «Мастер отчета», раскроется «Создание отчета». Выбираем «вид макета отчета». Макет – Табличный, Ориентация – Альбомная. Называем «Каталог автомобилей»:

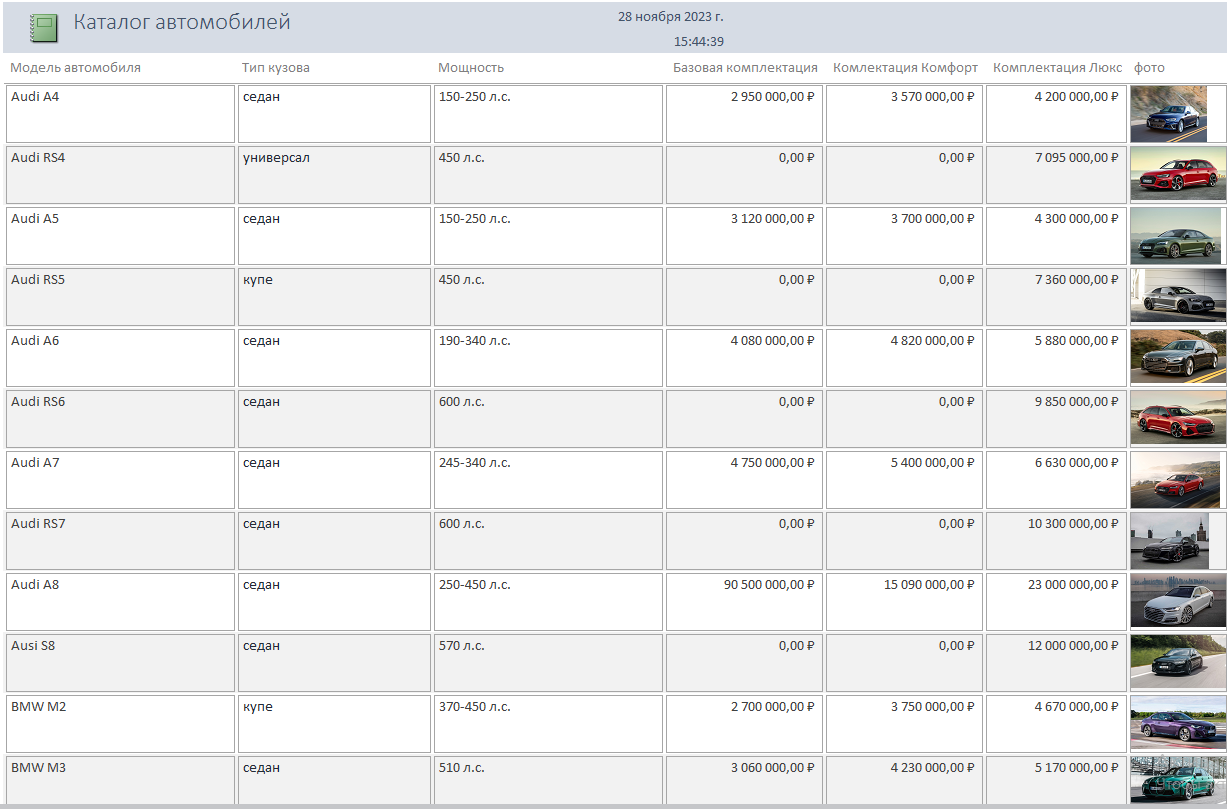


Рисунок 32 – Каталог автомобилей

В таком же формате создаем данные персонала:

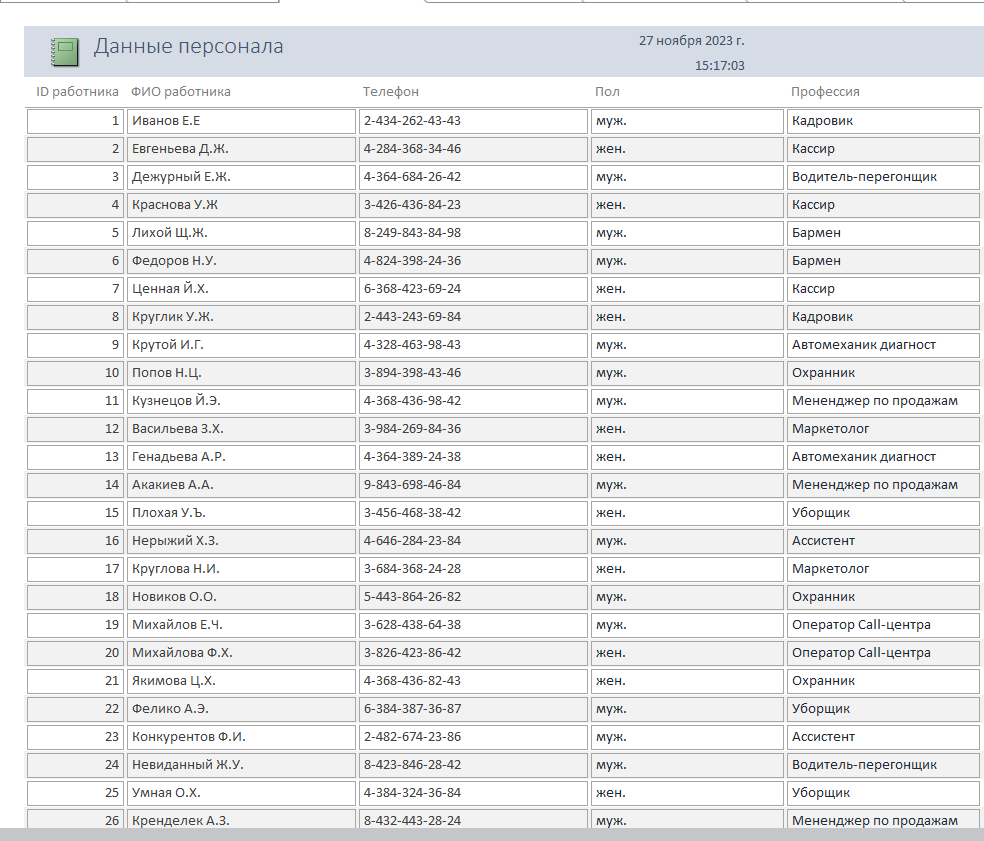


Рисунок 33 – Данные персонала

# Заключение

В результате выполнения данной курсовой работы была разработана и реализована система обработки и анализа данных, нацеленная на оптимизацию работы производственного предприятия. Цель работы была достигнута путем решения следующих задач: анализ основных проблем производственных предприятий, построение эффективной системы сбора и обработки данных, разработка алгоритмов анализа данных и формирование рекомендаций для оптимизации производственных процессов.

Проблемы, с которыми сталкиваются производственные предприятия, включают в себя неэффективное использование ресурсов, недостаточную оптимизацию производственных процессов, а также недостаточно точное прогнозирование спроса и управление запасами. Поэтому, разработка системы обработки и анализа данных, способной решать эти проблемы, является актуальной.

В процессе работы была проведена обзорная аналитическая работа, в ходе которой были рассмотрены существующие методы и подходы к обработке и анализу данных в производственных предприятиях. Были изучены научные исследования, публикации и опытные отчеты, касающиеся данной проблемы. Это позволило выявить основные факторы, влияющие на эффективность работы производственного предприятия, и разработать принципы работы системы обработки и анализа данных.

Созданная система обработки и анализа данных состоит из следующих компонентов: системы сбора данных, системы хранения данных, системы обработки данных и системы анализа данных. Система сбора данных осуществляет сбор информации из различных источников, таких как сенсоры, базы данных, файлы в форматах CSV и Excel. Система хранения данных обеспечивает сохранение полученных данных для последующей обработки и анализа. Система обработки данных включает в себя процессы стандартизации, фильтрации, агрегации и очистки данных. Система анализа данных использует методы статистики и машинного обучения для исследования данных, выявления закономерностей и прогнозирования производственных показателей.

Проведенное тестирование разработанной системы на реальных данных производственного предприятия показало ее эффективность и способность к повышению эффективности производственных процессов. Анализ полученных результатов свидетельствует о возможности улучшения управленческих решений, оптимизации производственной мощности и сокращении издержек.

В заключение можно сказать, что создание системы обработки и анализа данных является важным шагом в развитии производственного предприятия. Реализованная система позволяет увеличить эффективность работы предприятия, повысить качество выпускаемой продукции, снизить издержки и повысить конкурентоспособность на рынке. Разработанные рекомендации и результаты исследования имеют практическую ценность и могут быть использованы в реальной деятельности предприятия.

Таким образом, создание системы обработки и анализа данных для оптимизации работы производственного предприятия является актуальной и перспективной задачей, которая может принести значительные пользу организации, способствуя ее развитию и успешной деятельности.

# Список литературы

1. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли. - М.: Вильямс И.Д., 2020 - 1440 c.
2. Кошелев,  В.Е. Access 2020 Практическое  руководство /  В.Е.  Кошелев. – М. : ООО «Бином-Пресс», 2022 – 464 с.
3. Когаловский М.Р. Технология баз данных на персональных ЭВМ. – М.: Финансы и статистика, 2021 – 224 с.
4. Конноли Т., Бэгг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика. 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2021 – 1120 с.
5. Корнеев В.В., Гариев А.Ф., Васютин С.В., Райх В.В. Базы данных. Интеллектуальная обработка информации. – М.: Нолидж, 2021– 352 с.
6. Крёнке Д. Теория и практика построения баз данных. 8-е изд. – СПб.: Питер, 2020 – 800 с.
7. Кузин, А.В. Разработка баз данных в системе Microsoft Access: Учебник / А.В. Кузин, В.М. Демин. - М.: Форум, 2020 - 544 c.
8. Лабораторный  практикум  по  информатике:  Учебное пособие  для  вузов/ Под ред. В.А. Острейковского.- М.: Высшая школа, 2020
9. Лихтенштейн, В.Е. Стандартизация и разработка программных систем: Учебное пособие / В.Е. Лихтенштейн. - М.: Финансы и статистика, 2022 - 288 c.
10. Лукин, В.Н. Введение в проектирование баз данных / В.Н. Лукин. - М.: Вузовская книга, 2022 - 144 c.
11. Майерс Г. Архитектура современных ЭВМ. Т.2. М.: Мир, 1985.
12. Макин, Дж.К. Проектирование серверной инфраструктуры баз данных Microsoft SQL Server 2020 / Дж.К. Макин. - М.: Русская редакция, 2023. - 560 c.
13. Малыхина, М.П. Базы данных: основы, проектирование, использование / М.П. Малыхина. - СПб.: BHV, 2023. - 528 c.
14. Мамаев Е. Microsoft SQL Server 2020 в подлиннике. СПб.: Изд-во BHV, 2020.
15. Маркин, А.В. Разработка отчетов в информационных системах: Учебное пособие / А.В. Маркин. - М.: Диалог-МИФИ, 2021. - 312 c.
16. Мартин Дж. Организация баз данных в вычислительных системах: Пер. с англ. /Под ред. А.А. Стогния и А.Л. Щерса. – М.: Мир, 2020. – 664 с.
17. Мартишин, С.А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench: Методы и средства проектирования информационных систем и техноло / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, 2021. - 61 c.
18. Мейер Д. Теория реляционных баз данных: Пер. с англ. – М.: Мир,2022. – 608 с.
19. Мезенцев, К.Н. Автоматизированные информационные системы: Учебник для студентов учреждений среднего профессионального образования / К.Н. Мезенцев. - М.: ИЦ Академия, 2020 - 176 c.
20. Морган, С. Проектирование и оптимизация доступа к базам данных Microsoft SQL Server / С. Морган. - М.: Русская редакция, 2022. - 480 c.
21. Новалис С. Access 2021 Руководство по VBA: Пер. с англ. М.: Лори, 2021.506с.
22. Новиков Ф. А., Яценко А. Д. Microsoft Office 2000 в целом. СПб.: БХВ, 2023. 727 с.
23. Орлов Г. С. Изучение основ работы с базой данных Microsoft Access 97: Метод, указ, к лаб. работам / РГРТА. Рязань, 2020. 16 с.
24. Пасько В. Access 97. Русифицированная версия. Киев: BHV «Ирина», 2020 346 с.
25. Патрушина, С.М. Информационные системы в экономике: Учебное пособие / С.М. Патрушина, Н.А. Аручиди. - М.: Мини Тайп, 2022. - 144 c.
26. Перлова, О.Н. Проектирование и разработка информационных систем: Учебник / О.Н. Перлова. - М.: Академия, 2021. - 272 c.
27. Пирогов, В.Ю. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В.Ю. Пирогов. - СПб.: БХВ-Петербург, 2020. - 528 c.
28. Попов А.А. Создание приложений для FoxPro 2.5/2.6 в DOS и WINDOWS. – М.: Издательство «ДЕСС»,2020. – 672 с.
29. Починок, Н.Б. Разработка системы информационного обеспечения актуарной деятельности в пенсионном фонде российской федерации / Н.Б. Починок, М.В. Виноградова, А.Н. Малолетко. - М.: Русайнс, 2022. - 203 c.
30. Преснякова, Г.В. Проектирование интегрированных реляционных баз данных: Учебное пособие / Г.В. Преснякова. - М.: КДУ , 2020. - 224 c.
31. Проектирование баз данных. СУБД Microsoft Access: Учебное пособие для вузов / Н. Н. Гринченко, Е. В. Гусев, Н. П. Макаров, А. Н. Пылькин, Н. И. Цуканова. - М.: Горячая линия-Телеком, 2021. -240с.: ил.