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 Информационных технологий и радиоэлектроники

Кафедра ПМИ

КУРСОВАЯ РАБОТА

по \_\_\_\_\_\_\_\_\_\_Базам данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование дисциплины)

Тема Информационная система на тему: “Зоопарк”\_\_\_\_\_

Руководитель

Колпаков А.А.

(фамилия, инициалы)

(подпись) (дата)

Студент ИБ-122

(группа)

Мошков Т. Д.

(фамилия, инициалы)

(подпись) (дата)

Муром 2024

Содержание

[**Введение** 3](#_Toc184762231)

[**1. Анализ технического задания** 4](#_Toc184762232)

[**1.1. Описание предметной области** 4](#_Toc184762233)

[**1.2. Аналоги** 6](#_Toc184762234)

[**1.3. Анализ и выявление лучшей среды разработки.** 8](#_Toc184762235)

[**2. Разработка моделей данных.** 11](#_Toc184762236)

[**2.1. Концептуальная модель** 11](#_Toc184762237)

[**2.2. Логическая модель** 17](#_Toc184762238)

[**2.3 Приведение логической модели к физической.** 20](#_Toc184762239)

[**2.4. Создание таблиц** 24](#_Toc184762240)

[**3. Создание SQL-запросов** 28](#_Toc184762241)

[**4. Тестирование ИС** 31](#_Toc184762242)

[**Заключение** 37](#_Toc184762243)

[Приложение А: Модели данных 38](#_Toc184762244)

[Приложение Б: «Ссылка на GitHub» 39](#_Toc184762245)

[Список литературы 40](#_Toc184762246)

# **Введение**

Современные зоопарки являются не только местом для развлечения и отдыха, но и важными учреждениями, занимающимися сохранением биоразнообразия, образованием и научными исследованиями. В условиях растущего интереса к экологии и защите животных необходимо эффективно управлять данными о большом количестве видов, их характеристиках, состоянии здоровья и условиях содержания. Учитывая разнообразие животных, которые содержатся в зоопарках, включая млекопитающих, птиц, рептилий и амфибий, адекватная обработка и хранение информации становятся актуальными задачами.

Сложность учета данных о животных возрастает из-за необходимости хранить информацию о различных видах, их индивидуальных характеристиках, ветеринарных карточках, а также взаимодействии с посетителями и образовательными программами. Ручная обработка данных становится неэффективной и может привести к ошибкам, что может негативно сказаться на состоянии животных и репутации зоопарка. Кроме того, отсутствие оперативного доступа к информации о состоянии здоровья, питании и поведении животных может затруднить принятие решений в экстренных ситуациях.

Целью данной курсовой работы является разработка информационной системы для работы с базой данных зоопарка, которая будет эффективно организовывать хранение, обработку и анализ информации о животных. В результате реализации данной системы каждый сотрудник зоопарка сможет легко получать доступ к необходимым данным, что значительно повысит эффективность работы учреждения и обеспечит более качественный уход за животными.

# **1. Анализ технического задания**

## **1.1. Описание предметной области**

Предметная область данной курсовой работы — зоопарк, который представляет собой часть реального мира, данные о которой мы хотим отразить в базе данных. Основной задачей является проектирование и разработка приложения для автоматизации управления учётом животных, их здоровья и взаимодействия с посетителями. Предполагаемая база данных должна обеспечивать работу зоопарка по учету различных видов животных, автоматизированную выдачу отчетов о состоянии популяций, здоровье животных и посещаемости.

Процесс управления данными о животных осуществляется следующим образом:

Поиск информации о животных. Сотрудники зоопарка, включая ветеринаров и кураторов, взаимодействуют с базой данных для получения информации о каждом животном, включая его характеристики, состояние здоровья и особенности содержания. На этом этапе сотрудники могут также получить доступ к рекомендациям по уходу и питанию.

Регистрация новых животных. При поступлении нового животного в зоопарк главный ветеринар или специалист по содержанию регистрируют его в базе данных. Для регистрации необходимо предоставить информацию о виде, возрасте, состоянии здоровья, а также ветеринарные документы.

Возможное событие «Животное недоступно». В некоторых случаях животное может быть временно недоступно для посетителей, например, из-за болезни или проведения ветеринарных процедур. В таких ситуациях сотрудники должны иметь возможность обновлять информацию в базе данных, чтобы избежать недоразумений.

Взаимодействие с посетителями. Сотрудники зоопарка могут регистрировать посещения, предоставлять информацию о животных и организовывать мероприятия. При этом важно отслеживать отзывы и пожелания посетителей для улучшения услуг.

Выделим базовые сущности этой предметной области:

Животные.

Атрибуты животных: название, вид, класс, возраст, пол, страна обитания, масса, высота тела, продолжительность жизни, номер ветеринарной карточки.

Сотрудники зоопарка.

Атрибуты персонала: ФИО, должность, телефон. Эти данные позволят эффективно управлять процессами и взаимодействием между сотрудниками.

Посетители.

Атрибуты посетителей: ФИО, контактные данные.

Таким образом, создание базы данных для зоопарка позволит автоматизировать процессы учета, обеспечить более качественный уход за животными и улучшить взаимодействие с посетителями, что в итоге повысит эффективность работы учреждения.

## **1.2. Аналоги**

В настоящее время реализовано огромное количество АС для автосалонов.

Примером такой системы может являться управление-зоопарком.рф онлайн сервис для управления зоопарком на базе “1C:Предприятие”.

Достоинства:

* Автоматизация процессов: Система автоматизирует основные задачи управления зоопарком, такие как учет животных, управление их кормлением и медицинским обслуживанием.
* Удобный интерфейс: Интерфейс сделан интуитивно понятным, что упрощает обучение сотрудников и позволяет быстро начинать работу в системе.
* Поддержка учета и статистики: Поддержка аналитики позволяет следить за изменениями в популяции животных, фиксировать данные о болезнях, смертности и рождении.
* Уведомления и напоминания: Система может автоматически напоминать о важных задачах, таких как график вакцинации или другие медицинские процедуры.
* Мобильное приложение: Наличие мобильного приложения позволяет сотрудникам оперативно вносить данные и просматривать нужную информацию непосредственно на месте.

Недостатки:

* Зависимость от Интернета: Система требует постоянного подключения к Интернету, что может быть проблематично в случае временной недоступности сети.
* Затраты на внедрение: Стоимость внедрения и адаптации системы может быть значительной, особенно для небольших зоопарков.
* Ограниченная кастомизация: Возможности настройки системы под специфические нужды зоопарка могут быть ограничены.
* Потребность в обучении: для полного использования функционала требуется обучение персонала, что может занять некоторое время.
* Отсутствие некоторых функций: Некоторые функции, которые могли бы быть полезными, например, интеграция с внешними системами учета или финансирования, могут отсутствовать.

В качестве другого примера можно привести систему «Матрица. Обмен с ГИС Меркурий» – программное обеспечение, предназначенное для автоматизации учета, планирования и анализа работы предприятий, связанных с животноводческой продукцией: от производителей и переработчиков до торговых компаний, складов и ветеринарных служб.

Достоинства системы «Матрица. Обмен с ГИС Меркурий»:

* Полная автоматизация процессов обмена данными с ГИС «Меркурий», что снижает трудозатраты и повышает точность учёта продукции.
* Обеспечение единой базы данных для животноводческой продукции, контрагентов и операций, связанной с её оборотом.
* Инструменты для гибкого управления производственными и логистическими процессами в соответствии с требованиями системы ветеринарного контроля.
* Поддержка аналитики, отчетности и панелей мониторинга для отслеживания статуса продукции и выполнения требований законодательства.
* Возможность интеграции с другими учетными и ERP-системами для комплексного управления данными предприятия.
* Недостатки:
* Высокие первоначальные затраты на внедрение и обучение персонала.
* Возможные сложности при интеграции с существующими системами управления и учёта на предприятии.
* Необходимость адаптации бизнес-процессов предприятия к требованиям системы и законодательным стандартам.

## **1.3. Анализ и выявление лучшей среды разработки.**

Для разработки автоматизированной информационной системы (АИС) для зоопарка рассмотрим, какой язык программирования, среду разработки и базу данных лучше выбрать, учитывая, что для курсовой работы подойдут решения, которые оптимально сочетают функциональность и удобство.

Таблица 1 - C++ и C# для АИС «Зоопарк»

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **C++** | **C#** |
| **Удобство разработки** | Требует более сложного синтаксиса и управления памятью вручную, что может увеличить объем работы | Более простой, высокоуровневый синтаксис, автоматическое управление памятью, что делает разработку проще |
| **Поддержка ООП** | Полная поддержка ООП, но требует написания большего объема кода | Полная поддержка ООП, встроенные удобные средства для работы с объектами и коллекциями |
| **Скорость разработки** | Длительная, требует дополнительного времени на отладку и проверку работы с памятью | Высокая скорость разработки за счет сборки мусора и готовых библиотек для работы с базами данных |
| **Интеграция с базой данных** | Требуется использование сторонних библиотек для подключения к базам данных | Встроенные библиотеки для работы с базами данных, что упрощает интеграцию |
| **Применение** | Рекомендуется для системных и высокопроизводительных приложений | Идеально подходит для бизнес-логики и разработки информационных систем |

Для курсовой работы на тему АИС «Зоопарк» лучше выбрать C#, так как он предоставляет удобные инструменты для быстрой разработки информационных систем и обладает встроенными средствами для работы с базами данных. Благодаря поддержке объектно-ориентированного программирования разработка становится более структурированной и управляемой, что облегчает работу с большими объемами данных.

Кроме того, C# имеет обширную библиотеку и множество готовых компонентов, что ускоряет процесс разработки и позволяет сосредоточиться на реализации бизнес-логики, а не на рутинных задачах.

Таблица 2 - Visual Studio и SharpDevelop для разработки АИС

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Visual Studio** | **SharpDevelop** |
| **Функциональность** | Полный набор инструментов для разработки на C#, включая продвинутую отладку и рефакторинг кода | Базовый набор функций для разработки на C# |
| **Производительность** | Высокие требования к ресурсам, но идеален для крупных и средних проектов | Легковесный, занимает меньше ресурсов, но с ограниченными возможностями |
| **Интеграция с базами данных** | Поддержка интеграции с различными СУБД (MS SQL, SQLite и др.) через встроенные плагины и инструменты | Поддержка ограничена, но можно подключать базу данных вручную |
| **Расширяемость** | Широкий выбор расширений и плагинов для работы с разными фреймворками и библиотеками | Ограниченная поддержка плагинов, что уменьшает возможности настройки |
| **Подходит для учебных проектов** | Имеет бесплатную версию Community с полным функционалом, подходит для учебных проектов | Бесплатный, но с меньшим функционалом, требует больше ручных настроек |

Visual Studio Community – более удобный вариант для курсовой работы, так как предоставляет готовые инструменты для разработки информационных систем и включает встроенные средства для работы с базами данных.

Эта среда разработки предлагает широкий спектр функций, таких как мощный отладчик, поддержка различных языков программирования и интеграция с системами контроля версий. Кроме того, наличие активного сообщества и обширной документации позволяет легко находить решения для возникающих вопросов и проблем в процессе работы.

Таблица 3 – MS Access и SQLite для хранения данных АИС

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **MS Access** | **SQLite** |
| **Назначение** | Подходит для небольших локальных приложений и баз данных с ограниченным количеством пользователей | Встроенная база данных для мобильных и небольших настольных приложений |
| **Простота использования** | Дружелюбный интерфейс, позволяет легко настраивать таблицы и запросы через GUI | Легкая в использовании, интеграция с приложением напрямую через код |
| **Установка** | Требует Microsoft Office и работает только на Windows | Встроенная в приложение, не требует дополнительной установки |
| **Производительность** | Подходит для малых объемов данных, но не оптимизирована для высоких нагрузок | Подходит для малых и средних объемов данных, может работать в многопоточной среде |
| **Поддержка SQL** | Ограниченная поддержка SQL, более ориентирована на конечных пользователей | Полная поддержка SQL, что позволяет работать с запросами на уровне кода |
|  |  |  |

SQLite будет лучшим вариантом для курсовой работы, так как легко интегрируется с C# и требует меньше настроек, обеспечивая при этом высокую производительность для небольших и средних объемов данных.

# **2. Разработка моделей данных.**

Этот этап является ключевым при разработке автоматизированной информационной системы (АИС). В ходе работы определяются сущности, их атрибуты, а также устанавливаются связи между сущностями. На основе созданной диаграммы "Сущность – связь" или логической модели разрабатываются функциональные модели системы и диаграмма потоков данных. Для создания базы данных необходимо преобразовать логическую модель в физическую.

## **2.1. Концептуальная модель**

Создание концептуальной модели для автоматизированной информационной системы (АИС) зоопарка начинается с анализа предметной области и выделения основных сущностей. По условиям задачи, в зоопарке имеются животные, вольеры, посетители и работники.

Сущность "Животные" представляет собой конкретных животных, содержащихся в зоопарке, таких как тигры, слоны и другие виды. Важно учитывать связь животных с их местом содержания, поэтому выделена сущность "Вольеры", где содержится информация о вольерах для каждого животного. Для управления работой зоопарка выделена сущность "Работники", которая связана с животными и вольерами через атрибут "ОтветственныйСотрудникID", что позволяет определить, кто отвечает за уход за определённым животным и вольером.

Также существует связь между "Сотрудниками и "Билетами", так как сотрудники зоопарка занимаются продажей билетов. Сущность "Посетители" связана с "Билетами", что позволяет отслеживать, какие билеты были приобретены каждым посетителем, а также учитывать дату и номер билета. Таким образом, логическая модель отображает взаимодействие между ключевыми сущностями: посетители, билеты, сотрудники, животные, вольеры и услуги.

1. Сотрудники. Атрибуты сотрудников представлены на рисунке 1.

Изображение выглядит как текст, Шрифт, зарисовка, круг

Автоматически созданное описание

Рисунок 1 – атрибуты сотрудников

1. Животные

Атрибуты животных страна обитания, вид, название, номер ветеринарной карты, пол, возраст, масса, высота, семейство, класс, продолжительность жизни и ответственный сотрудник представлены на рисунке 2

Изображение выглядит как текст, круг, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 2 – атрибуты животных

1. Билет

Атрибуты билета: дата приобретения, кем выдан, вид услуг, суммарная стоимость и данные о посетителе представлены на рисунке 3

Изображение выглядит как текст, Шрифт, круг, диаграмма

Автоматически созданное описание

Рисунок 3 - атрибуты билета

1. Вольеры

Для вольера при построении базы данных мне будет важно знать вид вольера, вид корма, ответственного сотрудника и животное которое находится в вольере.

Необходимые атрибуты представлены на рисунке 4.

Изображение выглядит как текст, круг, Шрифт, зарисовка

Автоматически созданное описание

Рисунок 4 - атрибуты вольера

В результате анализа предметной области была составлена коцептуальная модель данных, представленная на рисунке 5.

Изображение выглядит как снимок экрана, круг, Графика, черно-белый

Автоматически созданное описание

Рисунок 5 – концептуальная модель данных

Описание связей между сущностями системы зоопарка:

1. Посетитель – Покупает – Билет:

Посетитель приобретает билет, который включает атрибуты:

* Номер билета
* Дата покупки
* Услуги, включенные в билет
* Суммарная стоимость
* Фамилия и имя покупателя
* Сотрудник, выдавший билет (фиксируется в системе)

1. Билет – Разрешает доступ – Животное:

Билет предоставляет посетителю право на просмотр животных в зоопарке.

1. Сотрудник – Продает – Билет:

Работник зоопарка отвечает за процесс продажи билетов. У сотрудника имеются следующие атрибуты:

* ФИО
* Возраст
* Должность
* Зарплата
* Стаж работы

1. Сотрудник – Ухаживает – Животное:

Сотрудники зоопарка осуществляют уход за животными. Каждый животное связано с ответственным сотрудником.

1. Животное – Содержится в – Вольер:

Животные размещаются в специальных вольерах, которые имеют следующие атрибуты:

* Вид вольера,
* Корм, который требуется для животных,
* Сотрудник, ответственный за вольер.

1. Животное – Имеет характеристики:

Животное обладает следующими атрибутами:

* Название
* Вид
* Пол
* Возраст
* Номер ветеринарной карты
* Высота
* Масса
* Продолжительность жизни
* Класс и семейство

## **2.2. Логическая модель**

Логическая модель данных — это абстрактное представление структуры данных, используемое для проектирования и планирования баз данных.  
Её создание начинается с анализа предметной области и выделения ключевых сущностей. Данная логическая модель данных отражает основные сущности, связанные с функционированием зоопарка, включая посетителей, билеты, сотрудников, животных и вольеры. Она позволяет эффективно управлять данными, обеспечивая работу зоопарка.

Основные сущности и их атрибуты:

Посетители:

* ID – Уникальный идентификатор посетителя (первичный ключ).
* Фамилия.
* Имя.

Билет:

* ID – Уникальный идентификатор билета (первичный ключ).
* Дата\_покупки – Дата приобретения билета.
* Суммарная\_стоимость – Общая стоимость билета.
* Услуги – Перечень услуг, предоставляемых по билету.
* Имя\_и\_фамилия\_покупателя – Персональные данные посетителя, оформившего билет.
* ID\_сотрудника – Внешний ключ, указывающий на сотрудника, оформившего билет.

Сотрудники:

* ID – Уникальный идентификатор сотрудника (первичный ключ).
* ФИО.
* Возраст.
* Должность.
* Зарплата.
* Стаж\_работы.

Животные:

* ID – Уникальный идентификатор животного (первичный ключ).
* Название.
* Пол.
* Возраст.
* Номер\_ветеринарной\_карты.
* Масса.
* Высота.
* Продолжительность\_жизни.
* Класс.
* Семейство.
* ID\_вольера – Внешний ключ, связывающий животное с вольером.

Вольеры:

* ID – Уникальный идентификатор вольера (первичный ключ).
* Вид\_вольера.
* Корм.
* ID\_сотрудника – Внешний ключ, связывающий вольер с ответственным сотрудником.

Связи между сущностями:

1. Посетитель покупает билет.
2. Билет предоставляет доступ к животным.
3. Сотрудники продают билеты.
4. Сотрудники ухаживают за животными.
5. Животные содержатся в вольерах, которые находятся под ответственностью сотрудников.

На основе перечисленных данных была построена логическая модель, которая обеспечивает структурированное управление данными в зоопарке.

На основе этих данных была составлена логическая модель данных, представленная на рисунке 7.

Изображение выглядит как текст, диаграмма, План, линия

Автоматически созданное описание

Рис. 7 – Логическая модель данных

## **2.3 Приведение логической модели к физической.**

Приведение логической модели к физической — это важный этап проектирования базы данных, на котором абстрактная логическая структура преобразуется в конкретные технические решения для реализации в системе управления базами данных (СУБД). Логическая модель описывает основные сущности, атрибуты и связи между ними, но не учитывает такие аспекты, как типы данных, индексы, ограничения, способы хранения данных и их оптимизацию для производительности.

Физическая модель, в свою очередь, детализирует каждый элемент с учётом возможностей выбранной СУБД. На этом этапе каждому атрибуту сущности в логической модели назначаются конкретные типы данных (например, INT, VARCHAR, DATE). Определяются первичные и внешние ключи, создаются индексы для оптимизации запросов, а также устанавливаются правила целостности данных, такие как ограничения уникальности и проверки на null-значения. Кроме того, физическая модель учитывает параметры производительности, распределение данных по таблицам и их размещение на физическом носителе, чтобы обеспечить оптимальное хранение и доступ к данным.

Таким образом, переход от логической модели к физической включает в себя не только техническую конкретизацию структуры данных, но и оптимизацию системы для её эффективного функционирования в реальной среде.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рис.8 – Связь между таблицами «Билеты» и «Посетители»

Таблицы «Сотрудники» «Вольеры» и «Животные» связаны через поля ID, что показано на рисунке 9.

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рис. 9 – Связь между таблицами «Сотрудники» «Вольеры» и «Животные»

«СотрудникОтветственныйЗаВольер» в таблице "Вольеры" ссылается на «Сотрудник». Каждый вольер имеет ответственного сотрудника, который управляет его состоянием. «СотрудникОтветственныйЗаЖивотное» в таблице "Животные" ссылается на «СотрудникID». Здесь каждый сотрудник отвечает за конкретное животное. «НомерВетеринарнойКарты» (PK): уникальный идентификатор для каждого животного

Таблицы «Услуги» и «Билеты» связаны через ID услуги, при заполнении клиентом билета стоимость всех услуг суммируется и получается суммарная стоимость билета. Сотрудник выдавший билет необходим при выдаче билета непосредственно в зоопарке. Это можно увидеть на рисунке 10.

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рис. 10 – Связь между таблицами «Услуги» и «Билеты»

Полная схема физической модели данных представлена на рисунке 11.

Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание

Рис. 11 – Физическая модель данных

В моей базе данных имеется несколько таблиц, каждая из которых выполняет свою функцию. Таблица "Посетители" хранит информацию о посетителях зоопарка, включая их уникальный идентификатор (ПосетительID), имя, фамилию и возраст. Таблица "Сотрудники" содержит данные о сотрудниках, таких как их уникальный идентификатор (СотрудникID), полное имя (ФИО), возраст, зарплату, стаж работы и должность.

Таблица "Билеты" хранит информацию о проданных билетах, включая уникальный идентификатор билета (IDБилета), услуги, включенные в билет, суммарную стоимость, дату покупки, а также данные сотрудника, выдавшего билет, и посетителя, который приобрел билет. В таблице "Вольеры" указаны данные о вольерах, включая уникальный номер вольера, тип вольера, корм, необходимый для животных в вольере, а также ссылки на сотрудников, ответственных за вольеры, и животных, находящихся в вольере.

Таблица "Животные" содержит информацию о животных зоопарка, такую как номер ветеринарной карты, название, семейство, класс, возраст, пол, страну обитания, массу, высоту и продолжительность жизни, а также сотрудника, ответственного за животное. Таблица "Услуги" хранит информацию о предлагаемых услугах, таких как уникальный идентификатор услуги, название, описание и стоимость услуги.

Эти таблицы связаны между собой через внешние ключи, что позволяет эффективно управлять информацией и выполнять необходимые операции в базе данных.

## **2.4. Создание таблиц**

Для создания таблиц будет использоваться СУБД SQLite. Для создания таблиц в базе данных SQLite необходимо использовать SQL-команды, которые описывают структуру данных и их связи. SQLite — это легковесная СУБД, которая поддерживает стандарт SQL, позволяя создавать, изменять и удалять таблицы, а также выполнять другие операции с базой данных.

На рисунках 12- представлены структуры каждой таблицы:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 12 – таблица Архив

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 13 – таблица Билеты

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Рисунок 14 – таблица Вольеры

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 15 – таблица Животные

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 16 – таблица Посетители

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Рисунок 17 – таблица Услуги

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 18 – таблица Сотрудники

Чтобы обеспечить корректную работу базы данных, необходимо задать связи между таблицами, используя внешние и первичные ключи. С помощью следующего запроса мы можем увидеть, где используются внешние ключи таблицы 'Билеты'

PRAGMA foreign\_key\_list('Билеты');

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 19 – запрос для 'Билеты'

В таблице 'Вольеры' используются следующие внешние ключи:

PRAGMA foreign\_key\_list('Вольеры');

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 20 – запрос для 'Вольеры'

Такой же запрос напишем для таблиц 'Животные', 'Посетители' и 'Сотрудники'

PRAGMA foreign\_key\_list('Животные');

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 21 – запрос для 'Животные'

PRAGMA foreign\_key\_list('Посетители');

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 22 – запрос для 'Посетители'

PRAGMA foreign\_key\_list('Сотрудники');

Изображение выглядит как текст, линия, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 23 – запрос для 'Сотрудники'

**3. Создание SQL-запросов**

SQL-запросы являются ключевым инструментом взаимодействия с реляционными базами данных, такими как база данных зоопарка. С их помощью можно эффективно управлять информацией о животных, посетителях, сотрудниках, вольерах и услугах, включая операции хранения, извлечения, обновления и удаления данных.

Результатом выполнения запросов может быть представление — виртуальная таблица, представляющая собой поименованный запрос, который используется как подзапрос при дальнейшем взаимодействии с базой данных. В отличие от обычных таблиц, представления не хранят данные самостоятельно, а формируют их на основе других таблиц базы, таких как данные о билетах, вольерах или посетителях.

1. Запрос для вывода данных из таблицы животные:

SELECT

`Номер ветеринарной карты` AS 'ID',

`Название`,

`Семейство`,

`Класс`,

`Возраст`,

`Пол`,

`Страна обитания`,

`Масса`,

`Высота`,

`Продолжительность жизни`,

`Сотрудник ответственный за животное`

FROM Животные;

2) Запрос для вывода данных из таблицы Билеты:

SELECT

`ID Билета` AS 'ID',

`Услуги`,

`Суммарная стоимость` AS ' стоимость билета',

`Дата покупки`,

`Сотрудник, выдавший билет`,

`Имя`,

`Фамилия`

FROM Билеты

3) Запрос для вывода данных из таблицы Сотрудники:

SELECT

`ID Работника` AS 'ID',

`ФИО` AS 'Фамилия Имя Отчество',

`Возраст`,

`Зарплата`,

`Стаж работы`,

`Должность`

FROM

Сотрудники;

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 24 – представление таблицы 'Сотрудники'

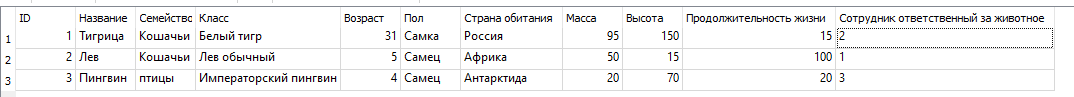


Рисунок 25 – представление таблицы 'Животные'

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Рисунок 26 – представление таблицы 'Билеты'

# **4. Тестирование ИС**

После завершения разработки системы необходимо провести тщательное тестирование на ее работоспособность при различных действиях пользователей. Программа зоопарка должна корректно отображать и выполнять действия пользователя в любых ситуациях. В ходе тестирования нужно проверить, как приложение реагирует на различные сценарии использования, включая как стандартные, так и исключительные случаи.

В исключительных моментах, таких как неправильный ввод данных, попытки доступа без должных прав или другие ошибки, система должна выводить окна с сообщениями, которые поясняют ситуацию и предлагают пути решения проблемы. Это поможет пользователям справляться с нештатными ситуациями, предоставляя четкие инструкции для дальнейших действий и минимизируя вероятность сбоя системы. Тестирование также должно включать проверку взаимодействия между различными формами и модулями, чтобы убедиться в их совместной работе без ошибок и зависаний.

Свое приложение я разделил на 4 различные формы: форма для выбора вида приложения, форма авторизации, интерфейс администратора и интерфейс пользователя.

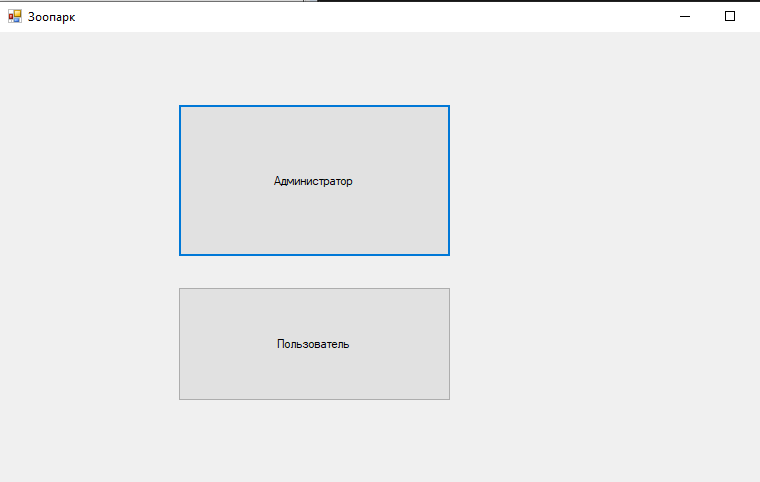


Рисунок 27 – форма 1

На первой форме представляется выбор зайти как администратор системы или как пользователь, здесь никаких тестов не требуется, обе кнопки работают.

Далее если выбрать Администратора выдастся окно для ввода логина и пароля:

Изображение выглядит как снимок экрана, текст, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 27 – форма 2

Все данные от учетных записей хранятся в бд, при попытке зайти под несуществующей учетной записью выдается ошибка:

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Автоматически созданное описание

Рисунок 28 – обработчик ошибок для формы 2

При вводе данных для существующей учетной записи нас перекинет на форму 3, которая называется интерфейс администратора:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 29 – интерфейс для администратора

Здесь я реализовал для администратора возможность регулировать базу данных, корректировать данные во всех имеющихся таблицах, создавать новые данные, так же реализован архив билетов, куда они заносятся после удаления.

Если же на форме 1 выбрать зайти за пользователя нас перекинет на форму 3 с интерфейсом уже для пользователя:  
Изображение выглядит как текст, снимок экрана, программное обеспечение, дизайн

Автоматически созданное описание

Рисунок 30 – интерфейс для пользователя

На этой форме пользователь может узнать информацию о животных, содержащихся в зоопарке, например их рост вес и т. д. Таким же образом пользователь может оформить билет для похода в зоопарк, выбрав интересующие его услуги заполнив имя и фамилию и нажав “Купить билет”

Так же для администратора предусмотрен контроль за всеми имеющимися таблицами в базе данных, представление таблицы показано на рисунке ниже.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 31 – Таблица вольеры

Таким же образом реализован интерфейс для остальных таблиц, который показан на рисунках 32–35.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 32 – Таблица сотрудники

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 33 – Таблица услуги

В таблице 'Билеты' для удобства пользования был реализован поиск по числу покупки билета, а также возможность посмотреть сколько раз человек посетил зоопарк и сколько всего потратил.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 34 – Таблица билеты

Для хранения удаленных из базы билетов был реализован архив, он показан на рисунке 33.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 35 – Таблица архив

# **Заключение**

В ходе выполнения курсовой работы на тему "Разработка автоматизированной системы управления зоопарком" были рассмотрены основные аспекты проектирования, разработки и внедрения информационной системы для оптимизации работы зоопарка. Цель работы заключалась в создании удобного инструмента для автоматизации процессов управления, учета и обслуживания, а также улучшения взаимодействия между персоналом, посетителями и системой.

Практическая значимость работы заключается в возможности использования предложенной системы в реальных условиях, что способствует повышению уровня автоматизации и организации работы зоопарков. Внедрение такой системы способствует не только улучшению качества обслуживания посетителей, но и повышению комфорта и безопасности животных за счет более точного контроля за их состоянием и содержанием.

Таким образом, в рамках данной курсовой работы была решена актуальная задача по созданию автоматизированной системы управления зоопарком, которая может быть использована для повышения эффективности управления и качества предоставляемых услуг.

# Приложение А: Модели данных

# Приложение Б: «Ссылка на GitHub»

# Список литературы

1. Кузнецов А. В., Иванов В. П. "Автоматизация процессов управления: теория и практика". — М.: Академия, 2018. — 384 с.

2. Воскресенский А. В., Платонов Ю. С. "Проектирование баз данных: Учебное пособие". — СПб.: Питер, 2020. — 336 с.

3. Ульман Д. Джеффри, Уидом Дж. "Основы систем баз данных". — 7-е изд. — М.: Вильямс, 2019. — 1248 с.

4. Стив Макконнелл. "Совершенный код. Практическое руководство по разработке программного обеспечения". — 2-е изд. — М.: Русская редакция, 2020. — 896 с.

5. Соколова М. В., Павлов И. А. "Основы зоотехники: содержание и учет животных". — СПб.: Профессия, 2017. — 208 с.

6. Бенджамен Н. "SQL для профессионалов: Полный справочник". — 3-е изд. — СПб.: БХВ-Петербург, 2021. — 560 с.