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 _____ ” Базы данных” _____
(наименование дисциплины)

Тема Автоматизированная 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 5
1. Анализ технического задания 6
1.1. Описание предметной области 6
1.2. Аналоги 7
1.3. Анализ и выявление лучшей среды разработки. 10
2. Разработка моделей данных. 13
2.1. Концептуальная модель 13
2.2. Логическая модель 17
2.3 Приведение логической модели к физической. 20
2.4. Создание таблиц 24
3. Разработка и реализация АИС 29
3.1. Создание SQL-запросов 31
3.2. Руководство пользователя 33
3.3. Руководство программиста 39
4. Тестирование АИС 48
Заключение 56
Список литературы 57
Приложение А: Модели данных 58
Приложение Б: «Ссылка на GitHub» 61

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 [1] 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 [2].

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 зоопарков.

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

Рассмотренные аналоги демонстрируют широкий функционал и возможности автоматизации, которые могут быть адаптированы для различных отраслей, включая управление зоопарками и учёт животноводческой продукции.

Однако, несмотря на преимущества, у них также есть недостатки, такие как высокие затраты на внедрение, необходимость обучения персонала, зависимость от интернет-соединения и ограничения в кастомизации. Эти аспекты следует учитывать при выборе или разработке аналогичных систем для автосалонов, чтобы обеспечить баланс между функциональностью, затратами и удобством использования.

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 [6].

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 "ОтвественныйСотрудникID", 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.

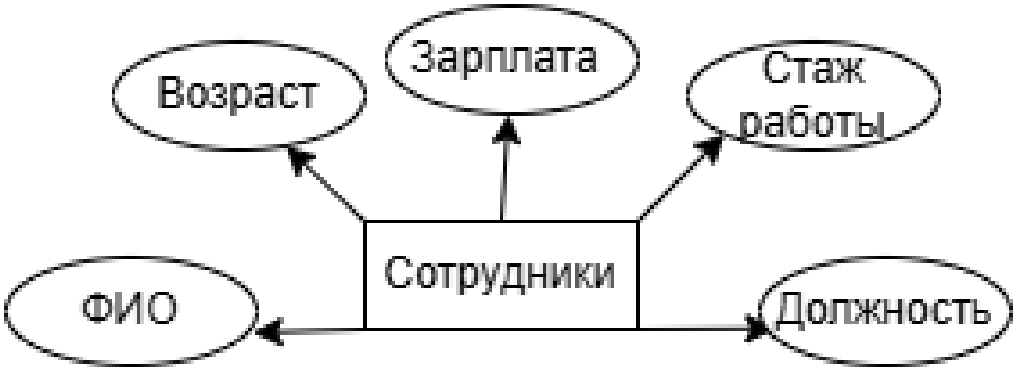


Рисунок 1 – атрибуты сотрудников

2) 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

Рисунок 2 – атрибуты животных

3) Билет

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

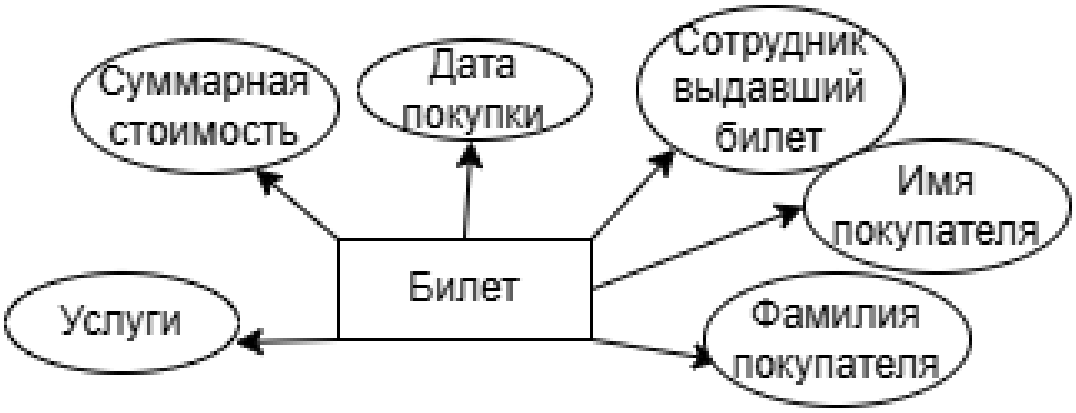


Рисунок 3 - атрибуты билета

4) 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.

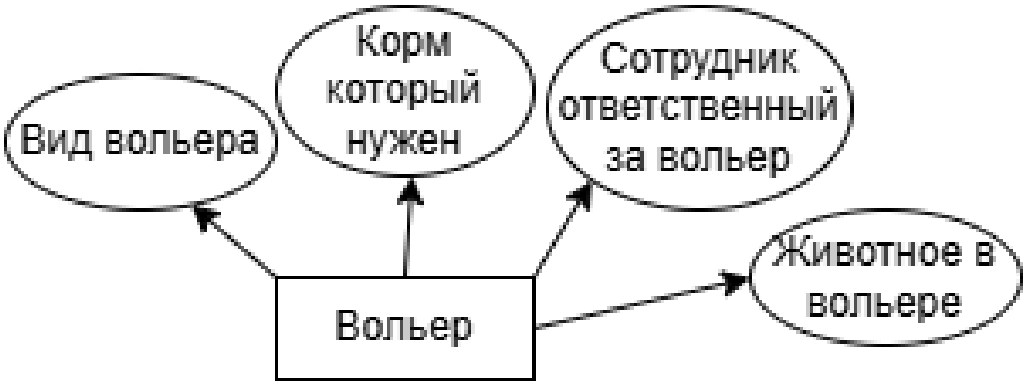


Рисунок 4 - атрибуты вольера

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

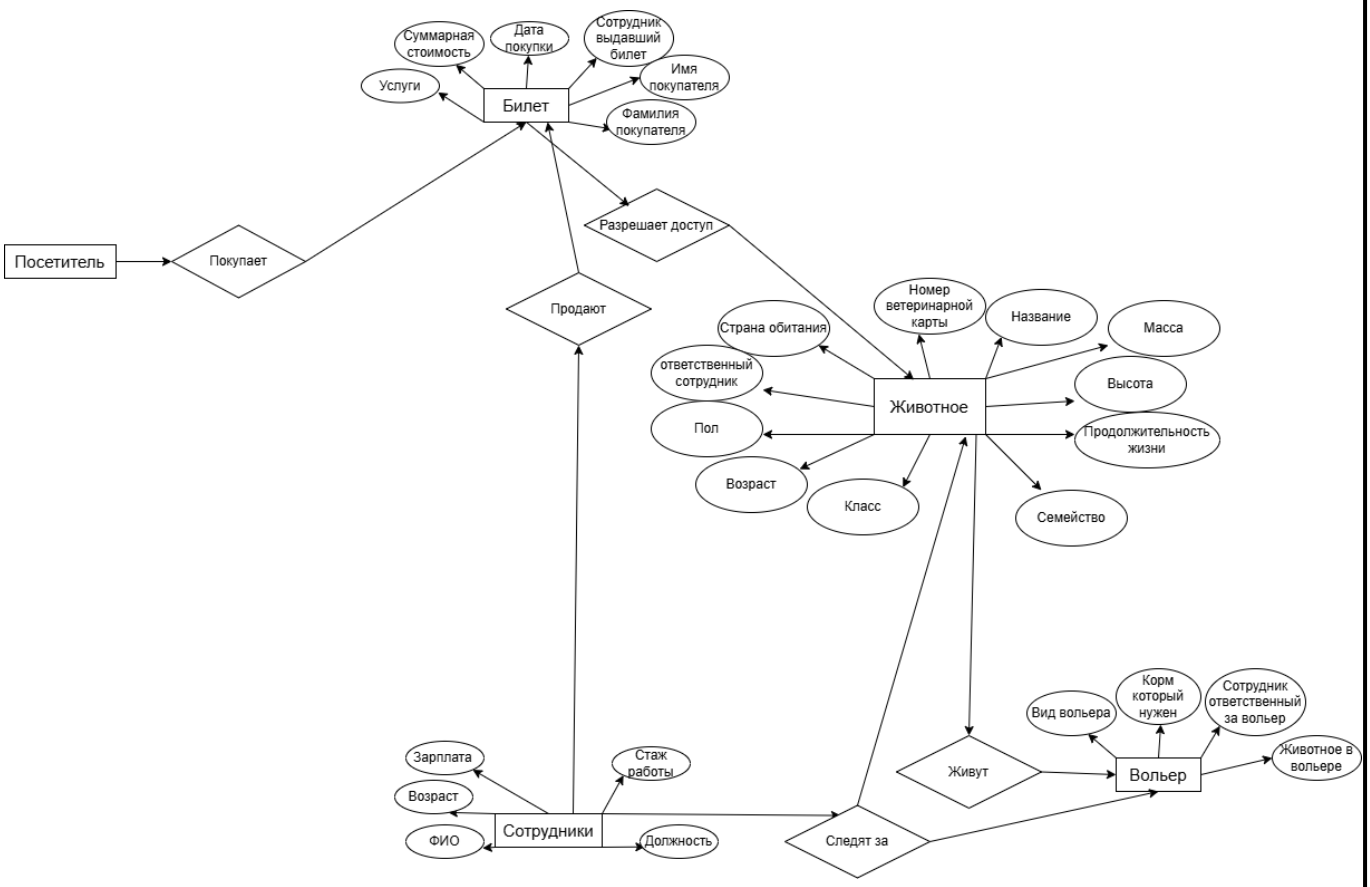


Рисунок 5 – концептуальная модель данных

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)

2. Билет – Разрешает доступ – Животное:

Билет предоставляет посетителю право на просмотр животных в зоопарке.

3. Сотрудник – Продает – Билет:

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

– ФИО
– Возраст
– Должность
– Зарплата
– Стаж работы

4. Сотрудник – Ухаживает – Животное:

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

5. Животное – Содержится в – Вольер:

Животные размещаются в специальных вольерах, которые имеют следующие атрибуты:

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

6. 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[3].

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.

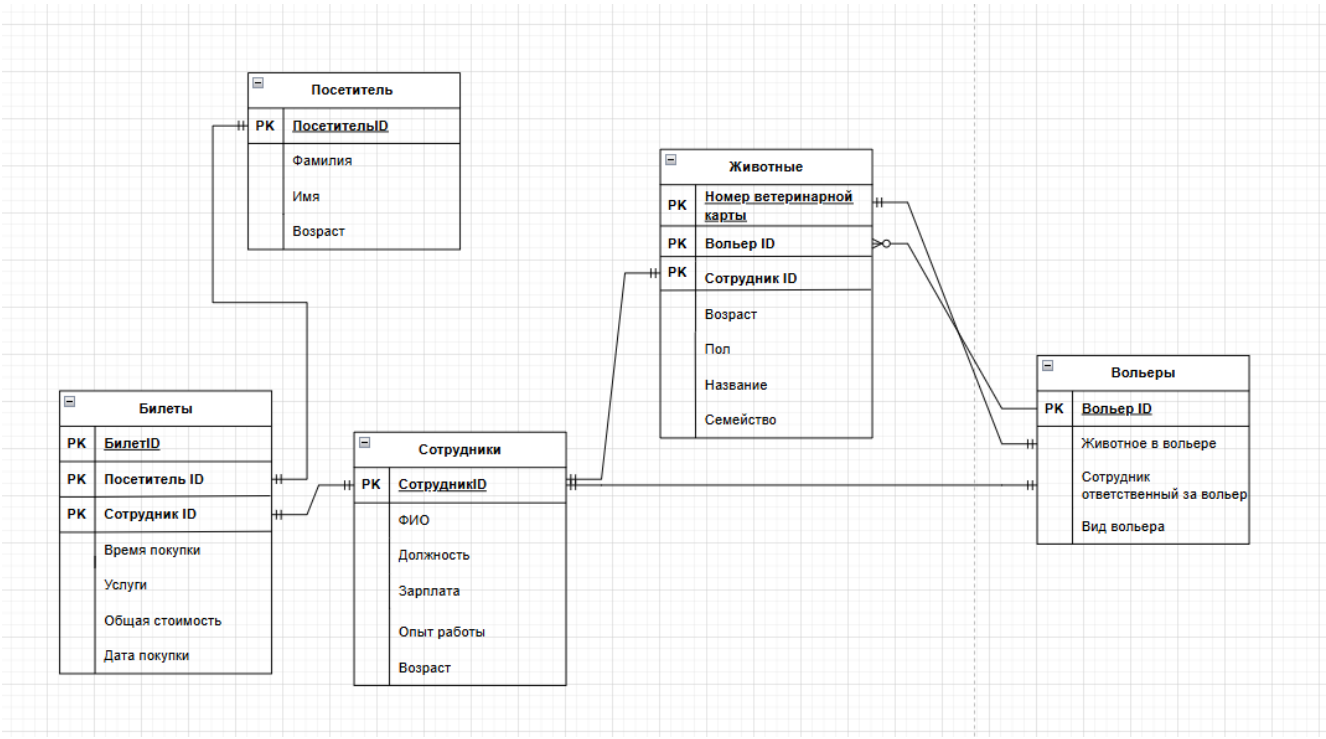


Рис. 7 – Логическая модель данных

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[2].

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.

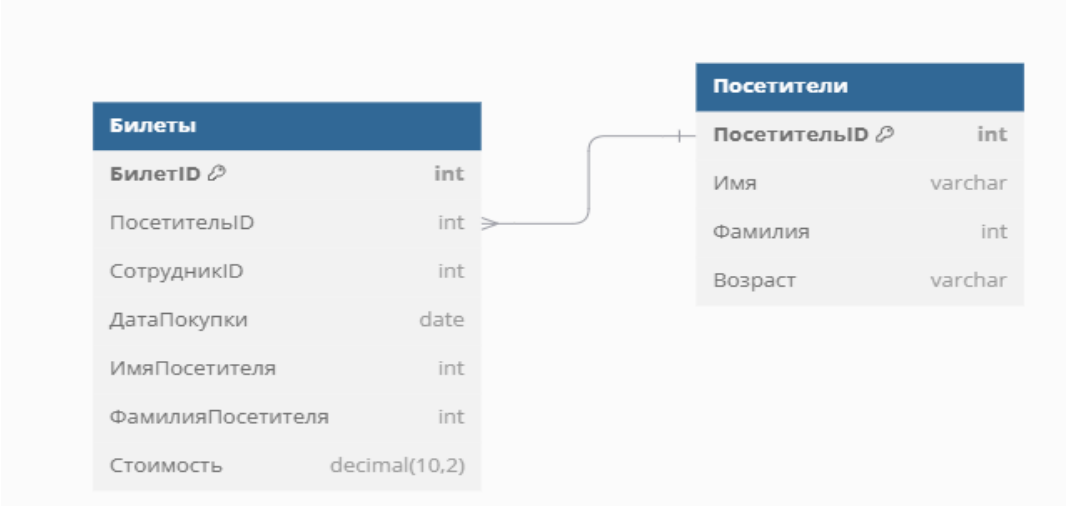


Рис.8 – Связь между таблицами «Билеты» и «Посетители»

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

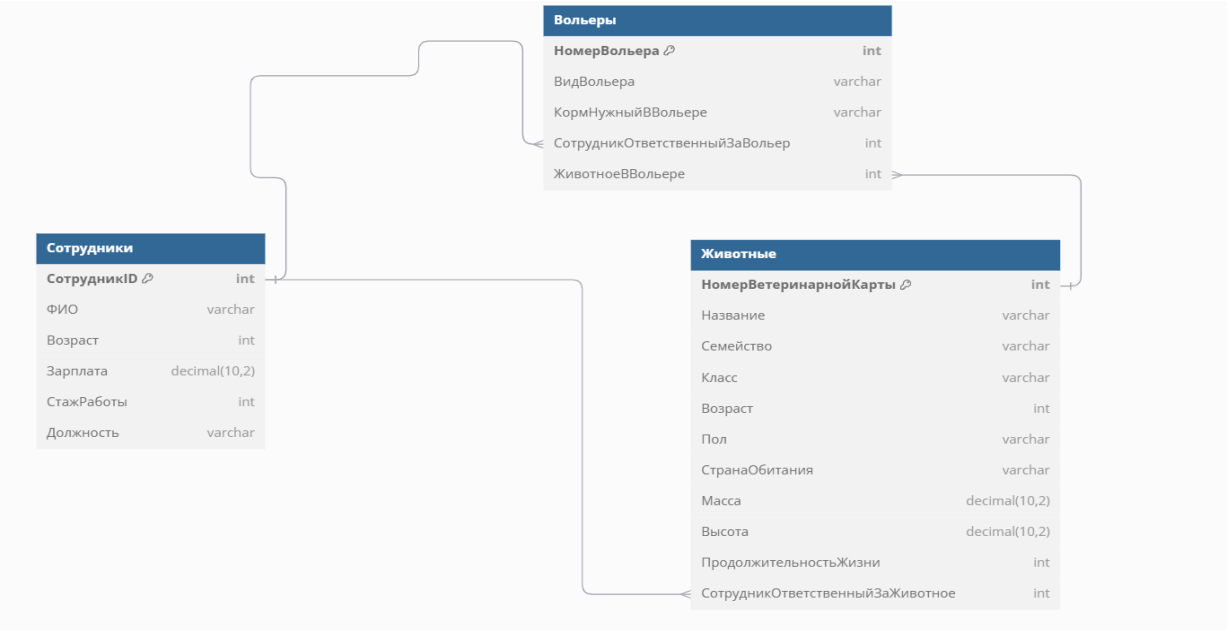


Рис. 9 – Связь между таблицами «Сотрудники» «Вольеры» и «Животные»

«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.

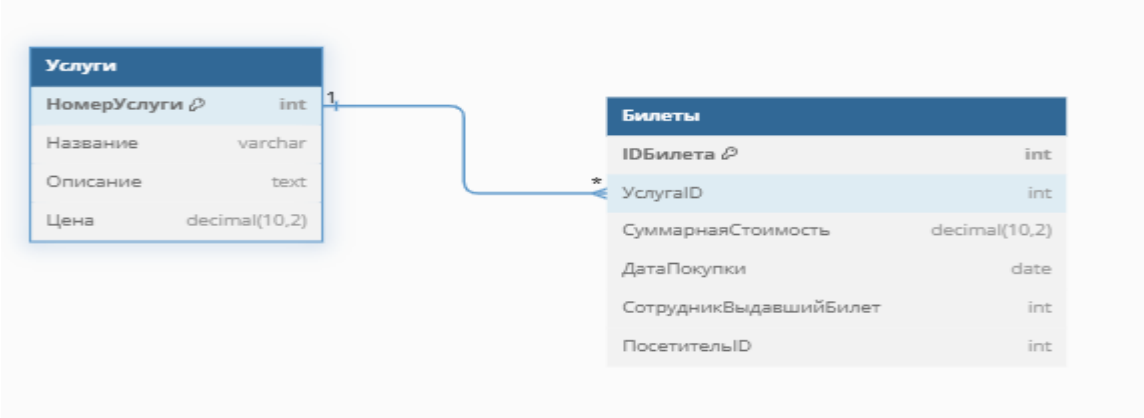


Рис. 10 – Связь между таблицами «Услуги» и «Билеты»

Полная схема физической модели данных представлена на рисунке 11.

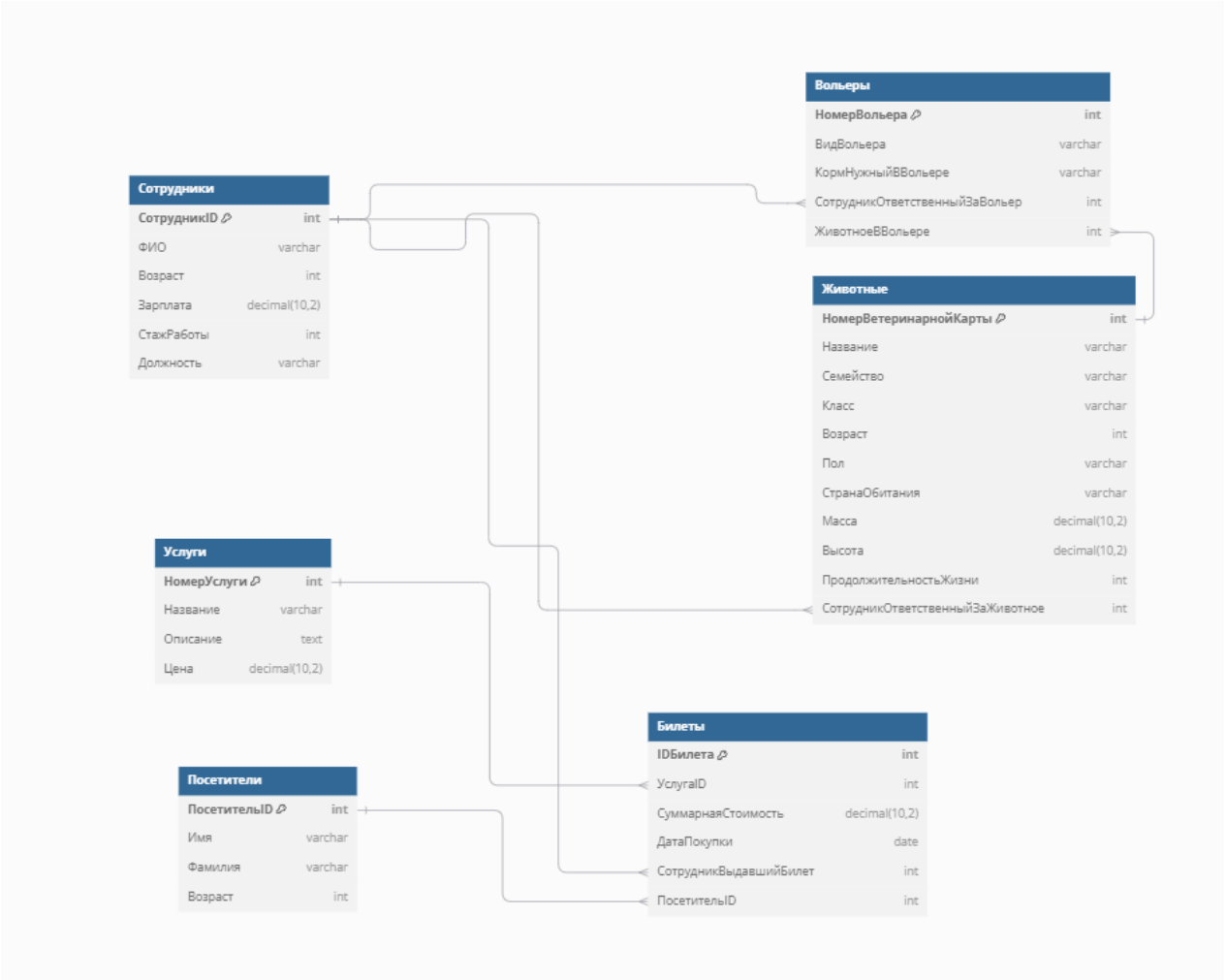


Рис. 11 – Физическая модель данных

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.

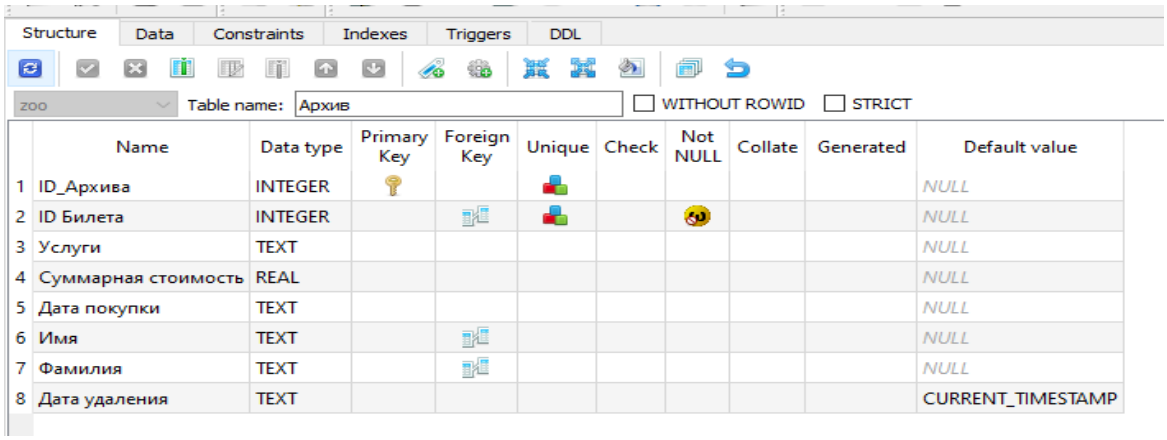
Таблица "Животные" содержит информацию о животных зоопарка, такую как номер ветеринарной карты, название, семейство, класс, возраст, пол, страну обитания, массу, высоту и продолжительность жизни, а также сотрудника, ответственного за животное. Таблица "Услуги" хранит информацию о предлагаемых услугах, 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[5].

На рисунках 12- представлены структуры каждой таблицы:



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	ID_Архива	INTEGER								NULL
2	ID Билета	INTEGER								NULL
3	Услуги	TEXT								NULL
4	Суммарная стоимость	REAL								NULL
5	Дата покупки	TEXT								NULL
6	Имя	TEXT								NULL
7	Фамилия	TEXT								NULL
8	Дата удаления	TEXT								CURRENT_TIMESTAMP

Рисунок 12 – таблица Архив

zoo Table name: Билеты WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	ID Билета	INTEGER								NULL
2	Услуги	TEXT								NULL
3	Суммарная стоимость	REAL								NULL
4	Дата покупки	TEXT								NULL
5	Сотрудник, выдавший билет	INTEGER								NULL
6	Имя	TEXT								NULL
7	Фамилия	TEXT								NULL

Рисунок 13 – таблица Билеты

zoo Table name: Вольеры WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	Номер вольера	INTEGER								NULL
2	Вид вольера	TEXT								NULL
3	Корм нужный в вольере	TEXT								NULL
4	Сотрудник ответственный за вольер	INTEGER								NULL
5	Животное в вольере	TEXT								NULL

Рисунок 14 – таблица Вольеры

zoo Table name: Животные WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	Номер ветеринарной карты	INTEGER								NULL
2	Название	TEXT								NULL
3	Семейство	TEXT								NULL
4	Класс	TEXT								NULL
5	Возраст	INTEGER								NULL
6	Пол	TEXT								NULL
7	Страна обитания	TEXT								NULL
8	Масса	REAL								NULL
9	Высота	REAL								NULL
10	Продолжительность жизни	INTEGER								NULL
11	Сотрудник ответственный за животное	TEXT								NULL

Рисунок 15 – таблица Животные

zoo Table name: Посетители WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	ПосетительID	INTEGER								NULL
2	Имя	INTEGER (1)								NULL
3	Фамилия	INTEGER								NULL
4	Возраст	TEXT								NULL

Рисунок 16 – таблица Посетители

zoo Table name: Услуги WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	Номер услуги	INTEGER								NULL
2	Название	TEXT								NULL
3	Описание	TEXT								NULL
4	Цена	REAL								NULL

Рисунок 17 – таблица Услуги

zoo Table name: Сотрудники WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	ID Работника	INTEGER								NULL
2	ФИО	TEXT								NULL
3	Возраст	INTEGER								NULL
4	Зарплата									NULL
5	Стаж работы	TEXT								NULL
6	Должность	TEXT								NULL

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');

Grid view | Form view

Total rows loaded: 3

	id	seq	table	from	to	on_update	on_delete	match
1	0	0	Сотрудники	Сотрудник, выдавший билет	ID Работника	NO ACTION	NO ACTION	NONE
2	1	0	Услуги	Услуги	Название	NO ACTION	NO ACTION	NONE
3	2	0	Архив	ID Билета	ID Билета	NO ACTION	NO ACTION	NONE

Рисунок 19 – ключ для 'Билеты'

В таблице 'Вольеры' используются следующие внешние ключи:

PRAGMA foreign_key_list('Вольеры');

Grid view | Form view

Total rows loaded: 2

	id	seq	table	from	to	on_update	on_delete	match
1	0	0	Сотрудники	Сотрудник ответственный за вольер	ID Работника	NO ACTION	NO ACTION	NONE
2	1	0	Животные	Животное в вольере	Название	NO ACTION	NO ACTION	NONE

Рисунок 20 – ключ для 'Вольеры'

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

PRAGMA foreign_key_list('Животные');

Grid view | Form view

Total rows loaded: 2

	id	seq	table	from	to	on_update	on_delete	match
1	0	0	Сотрудники	Сотрудник ответственный за животное	ID Работника	NO ACTION	NO ACTION	NONE
2	1	0	Вольеры	Название	Животное в вольере	NO ACTION	NO ACTION	NONE

Рисунок 21 – ключ для 'Животные'

PRAGMA foreign_key_list('Посетители');

Total rows loaded: 2

	id	seq	table	from	to	on_update	on_delete	match
1	0	0	Билеты	Фамилия	Фамилия	NO ACTION	NO ACTION	NONE
2	1	0	Билеты	Имя	Имя	NO ACTION	NO ACTION	NONE

Рисунок 22 – ключ для 'Посетители'

PRAGMA foreign_key_list('Сотрудники');

Grid view | Form view

Total rows loaded: 1

	id	seq	table	from	to	on_update	on_delete	match
1	0	0	Животные	ID Работника	Сотрудник ответственный за животное	NO ACTION	NO ACTION	NONE

Рисунок 23 – ключ для 'Сотрудники'

3. Разработка и реализация АИС

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;
```

	ID	Фамилия Имя Отчество	Возраст	Зарплата	Стаж работы	Должность
1	1	Мошков Тимофей Дмитриевич	29	30000	7	Охранник
2	2	Рысев Андрей Сергеевич	30	35000	5	Ветеринар
3	3	Бушуев Александр Анатольевич	55	30000	35	Ветеринар
4	4	Петров Никита Андреевич	27	25000	3	Экскурсовод

Рисунок 24 – представление таблицы 'Сотрудники'

	ID	Название	Семейство	Класс	Возраст	Пол	Страна обитания	Масса	Высота	Продолжительность жизни	Сотрудник ответственный за животное
1	1	Тигрица	Кошачьи	Белый тигр	31	Самка	Россия	95	150	15	2
2	2	Лев	Кошачьи	Лев обычный	5	Самец	Африка	50	15	100	1
3	3	Пингвин	птицы	Императорский пингвин	4	Самец	Антарктида	20	70	20	3

Рисунок 25 – представление таблицы 'Животные'

Filter data | Total rows loaded: 4

	ID	Услуги	Суммарная	Дата покупки	Имя	Фамилия
1	3	Пропуск в зоопарк, Экскурсия	650	2024-11-13 15:56:04	NULL	NULL
2	4	Пропуск в зоопарк, Экскурсия	650	2024-11-21 13:32:11	Тимофей	Мошков
3	6	Пропуск в зоопарк, Корм для уток	550	2024-11-27 15:10:32	Даниил	Бубнов
4	7	Пропуск в зоопарк, Экскурсия, Корм для уток	700	2024-11-27 15:12:34	Алексей	Мокоронин

Рисунок 26 – представление таблицы 'Билеты'

3.1. Создание SQL-запросов

SELECT

[Вид_вольера] AS 'Вид вольера',

[Корм нужный в вольере] AS 'Корм',

(SELECT ФИО FROM Сотрудники WHERE [ID Работника] = Вольеры.[Сотрудник ответственный за вольер]) AS 'ФИО Сотрудника',

[Животное_в_вольере] AS 'Животное'

FROM Вольеры";

Данный запрос выбирает столбцы в таблице Сотрудники и Вольеры с помощью команды FROM. При помощи ключевого слова AS присваивает им название и выводит их на экран.

	Вид вольера	Корм	ФИО Сотрудника	Животное
1	Хищник	Мясо	Мошков Тимофей Дмитриевич	Барсик
2	Птицы	Рыбы	Бушуев Александр Анатольевич	Пин
3	Хищник	Мясо	Бушуев Александр Анатольевич	Тигруля
4	Птицы	рыба насекомые	Петров Никита Андреевич	Рома
5	Хищник	мясо кости	Бушуев Александр Анатольевич	Пушок

Рис. 27 – Результат выполнения запроса на таблице 'Вольеры'

3.1.2 Запрос для вставки в таблицу INSERT:

"INSERT INTO Билеты (Услуги, [Суммарная стоимость], [Дата покупки], Имя, Фамилия) " + "VALUES (@services, @totalCost, @purchaseDate, @firstName, @lastName); " + "SELECT last_insert_rowid();"

В приведённом SQL-запросе выполняются две последовательные операции: вставка данных в таблицу и получение ID последней вставленной строки с использованием функции last_insert_rowid(). Результат показан на рисунке 28.

Добавить животное | Вольеры | Сотрудники | Услуги | Билеты | Архив билетов

№	Услуги	Суммарная стоимость	Дата покупки	Сотрудник, выдавший билет	Имя покупателя	Фамилия покупателя
1	Пропуск в зооп...	650	2024-11-13 15:5...	Андрей Рысев	Тимофей	Мошков
2	Пропуск в зооп...	550	2024-11-27 15:1...	Андрей Рысев	Данил	Бубнов
3	Пропуск в зооп...	700	2024-11-27 15:1...	Андрей Рысев	Алексей	Мокронин
4	Пропуск в зооп...	655	2024-12-12 20:2...	Мошков Тимоф...	Анастасия	Абрамова
5	Пропуск в зооп...	700	2024-11-27 15:1...	Мошков Тимоф...	Алексей	Мокронин

Рис. 28 – Результат выполнения запроса на таблице 'Билеты'

3.1.3 Запрос для обновления записей UPDATE:

```
UPDATE Животные
SET
    Кличка = @name,
    Семейство = @family,
    Класс = @species,
    Возраст = @age
WHERE
    [Номер ветеринарной карты] = @id";
```

Оператор UPDATE используется для изменения значений существующих строк в таблице. В данном случае обновляются данные в таблице Животные. В блоке SET указываются столбцы, которые нужно обновить, и новые значения для этих столбцов.

Новые значения передаются через параметры (@name, @family, @species, @age), что безопасно для предотвращения SQL-инъекций. Оператор WHERE определяет, какие строки должны быть обновлены. Без WHERE все строки таблицы будут обновлены.

3.1.4 Запрос для удаления записей (DELETE)

"DELETE FROM Билеты WHERE [ID Билета] = @ticketId";

Этот SQL-запрос предназначен для удаления строки из таблицы Билеты на основе условия, которое определяется значением параметра @ticketId.

3.2. Руководство пользователя

Запуск приложения и первое окно. После запуска приложения на экране появляется главное окно, содержащее два варианта входа:

Кнопка "Войти как администратор" (синяя кнопка): позволяет войти в режим администратора.

В этом режиме пользователю предоставляется доступ к расширенному функционалу:

- Управление данными о билетах и пользователях.
- Редактирование, добавление и удаление записей.
- Просмотр отчётов и архивов.

Для входа требуются дополнительные данные (логин и пароль).

Кнопка "Войти как пользователь" (зелёная кнопка):

Позволяет войти в режим обычного пользователя. В этом режиме доступны функции просмотра информации: просмотр доступных услуг, информация о билетах, покупках и других доступных данных. В режиме пользователя функции редактирования и удаления данных ограничены. Окно с вариантами входа показано на рисунке 29.

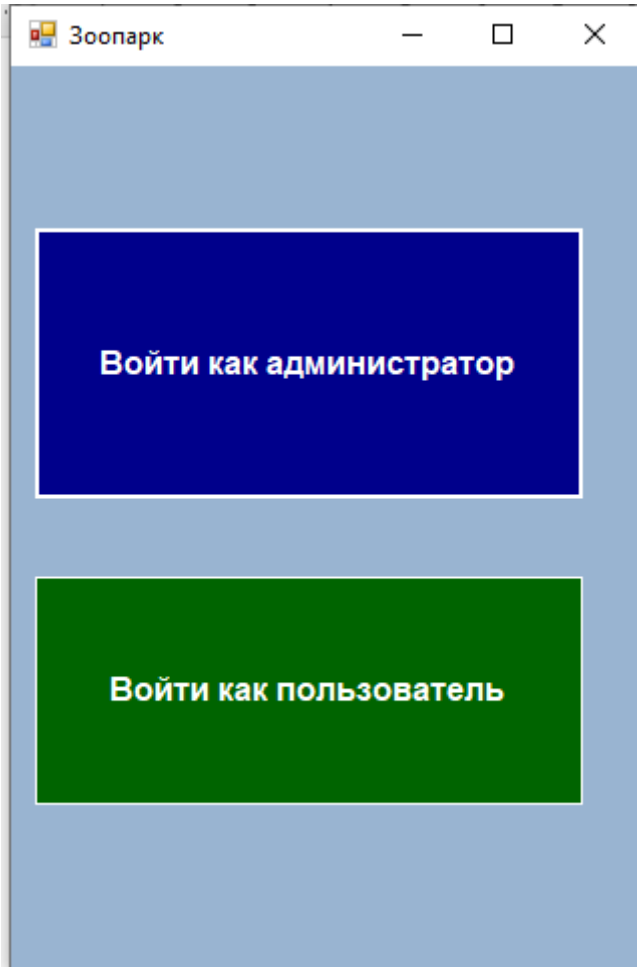


Рисунок 29 – Окно входа

Окно авторизации предназначено для входа администратора в систему и проверки его учетных данных (логин и пароль). При вводе этих данных они сверяются с теми, что хранятся в таблице users в базе данных и производится допуск пользователя, если такие данные найдены.

Данное окно представляет собой простую форму с двумя полями для ввода данных и кнопкой для подтверждения.

Окно авторизации предназначено для ограничения доступа к административному функционалу. Оно обеспечивает проверку логина и пароля и позволяет входить только авторизованным пользователям. Интерфейс окна авторизации представлен на рисунке 30.

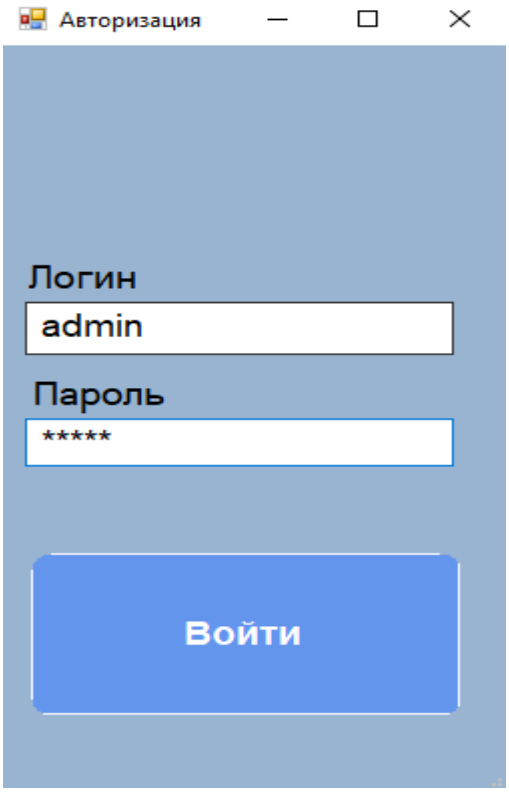


Рисунок 30 – Окно авторизации

Окно "Интерфейс администратора" позволяет управлять данными о животных, находящихся в системе. Это основное окно для работы с базой данных животных, включающее функционал добавления, изменения, поиска и удаления записей. Это окно показано на рисунке 31.

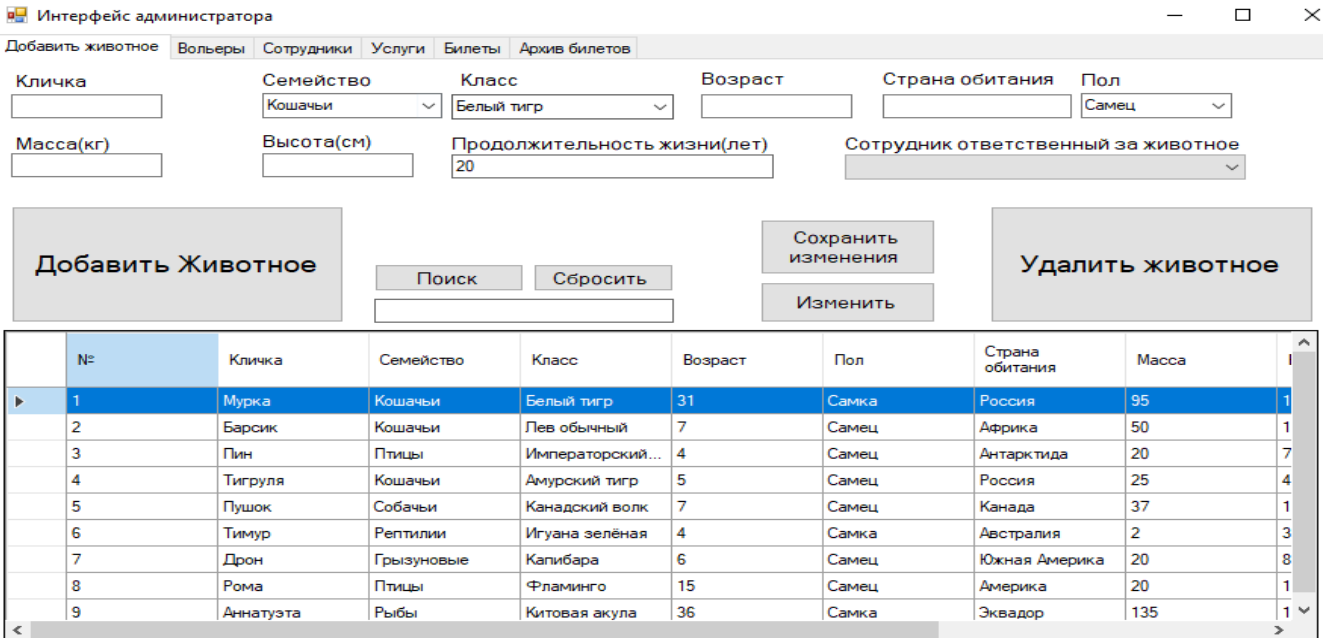


Рисунок 31 – Интерфейс администратора

Окно "Вольеры" предоставляет функционал для управления информацией о вольерах и связанных с ними животных. Администратор может добавлять, редактировать и удалять записи о вольерах, а также связывать животных и сотрудников с конкретными вольерами. Это окно показано на рисунке 32.

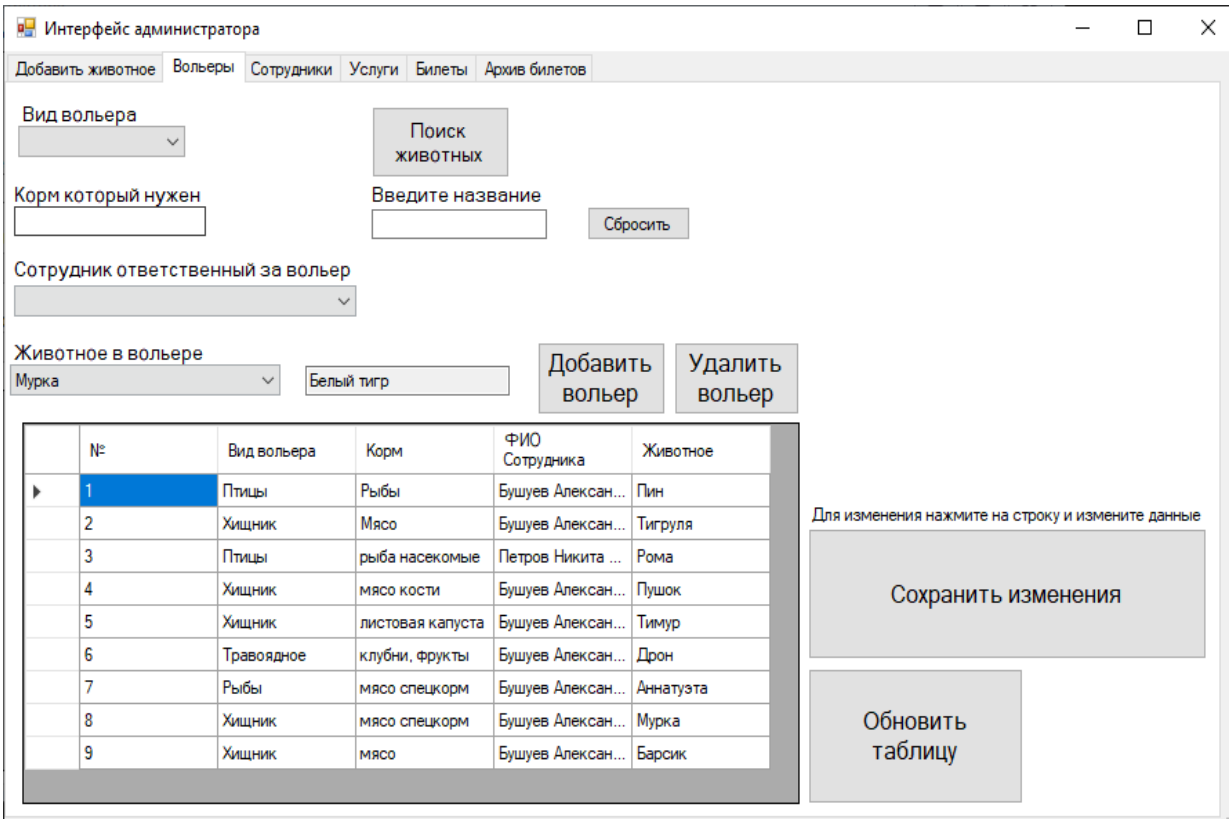


Рисунок 31 – Окно Вольеры

Окно "Сотрудники" предназначено для управления данными о сотрудниках. В нем предусмотрен функционал для добавления, изменения, удаления записей, а также для поиска сотрудников по фамилии.

Окно "Сотрудники" предоставляет удобный интерфейс для управления данными о сотрудниках, обеспечивая полный набор функций для добавления, редактирования, поиска и удаления записей. Это позволяет поддерживать актуальную и структурированную базу данных сотрудников. Это окно представлено на рисунке 32.

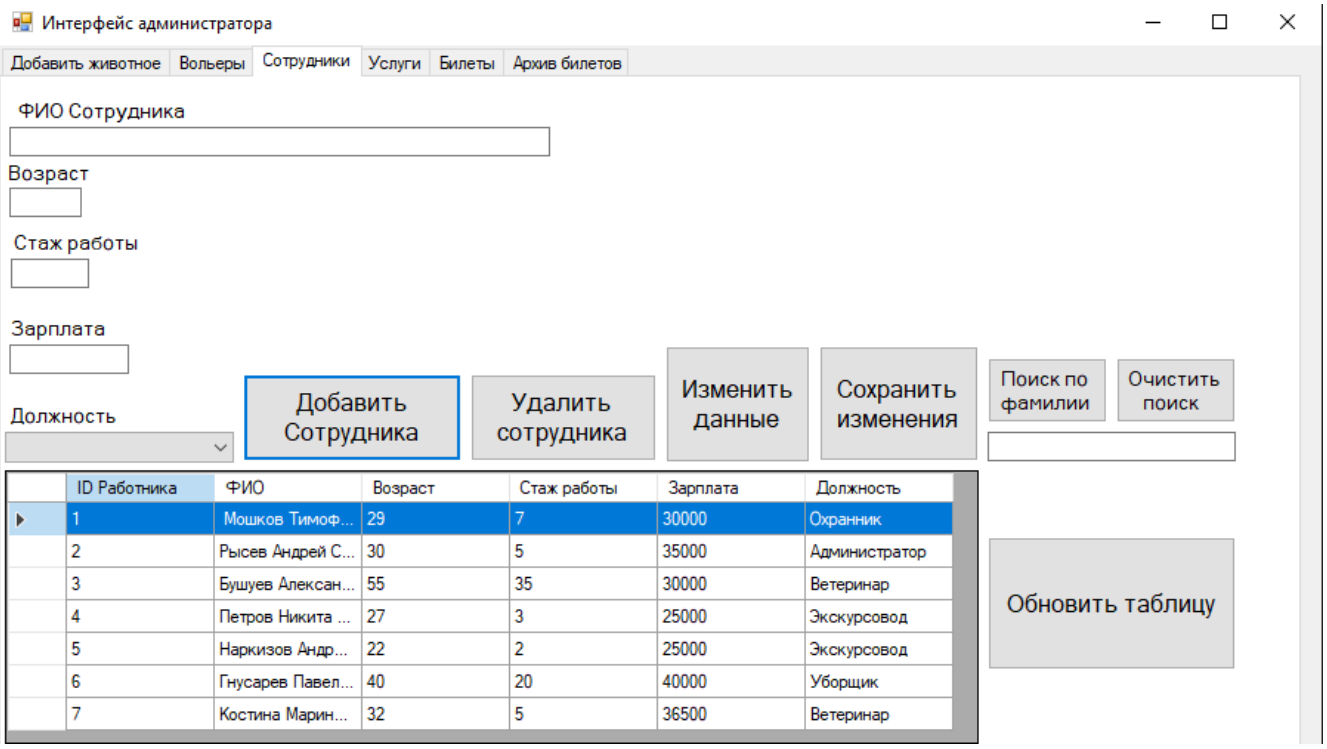


Рисунок 32 – Окно Сотрудники

Окно "Услуги" предназначено для управления данными о доступных услугах зоопарка. Оно позволяет добавлять новые услуги, редактировать, удалять, а также просматривать подробное описание услуг. Интерфейс показан на рисунке 33.

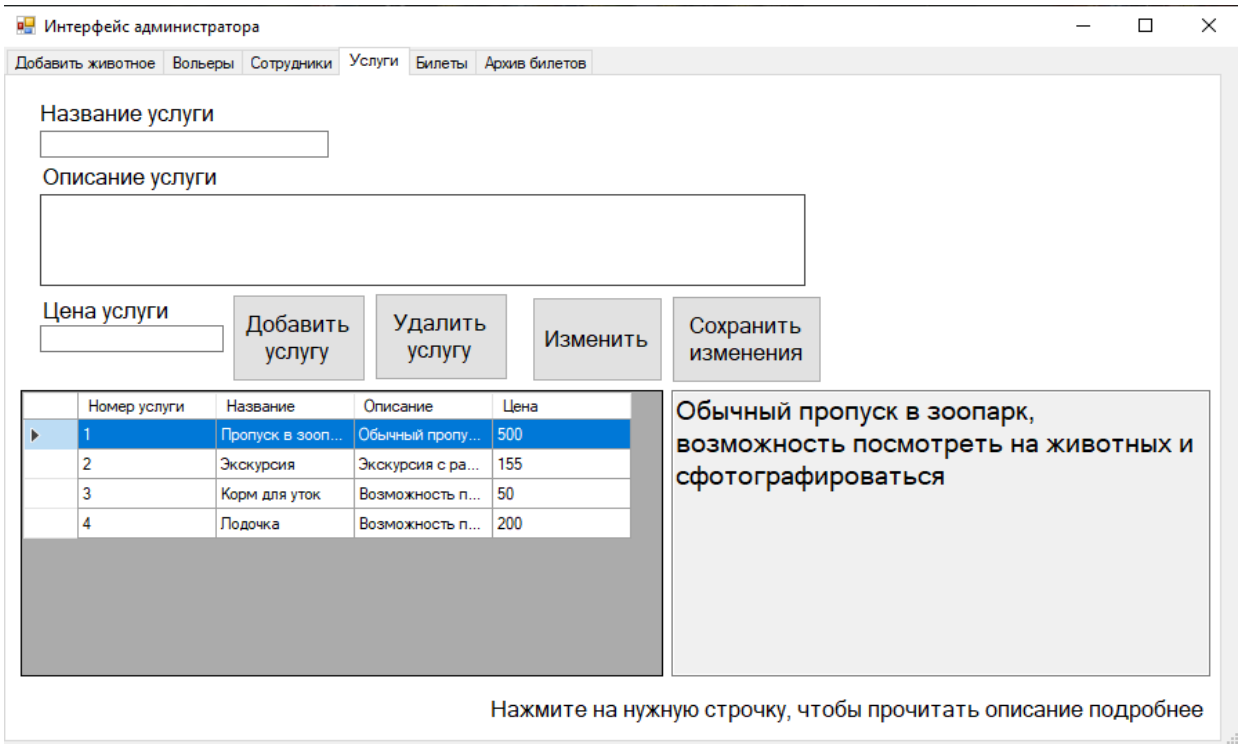


Рисунок 33 – Окно Услуги

Окно "Билеты" предназначено для управления данными о билетах зоопарка. Оно предоставляет возможность создавать новые билеты, редактировать существующие, удалять их, а также искать информацию по билетам и покупателям. Это окно показано на рисунке 34.

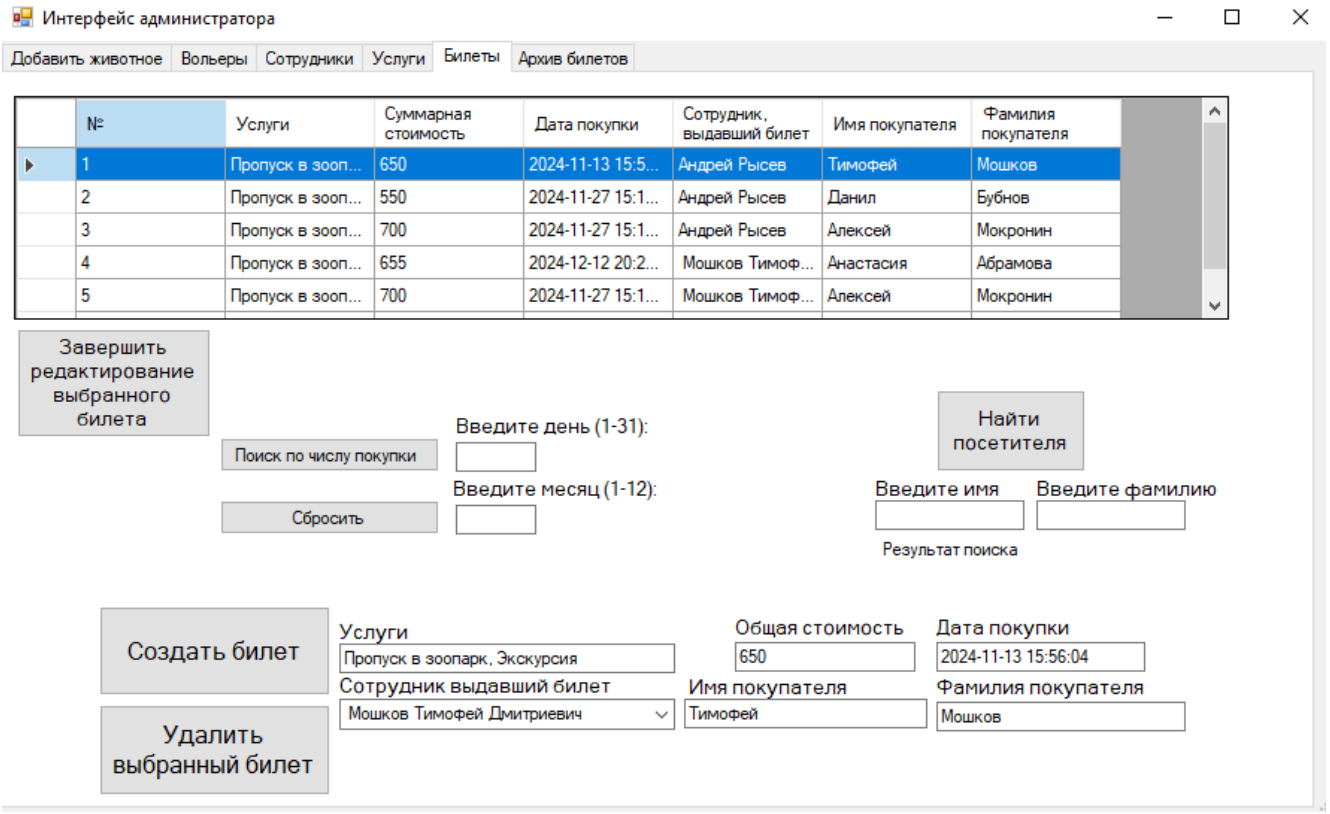


Рисунок 34 – Окно Билеты

Так же создан интерфейс для просмотра удаленных билетов, он показан на рисунке 35.

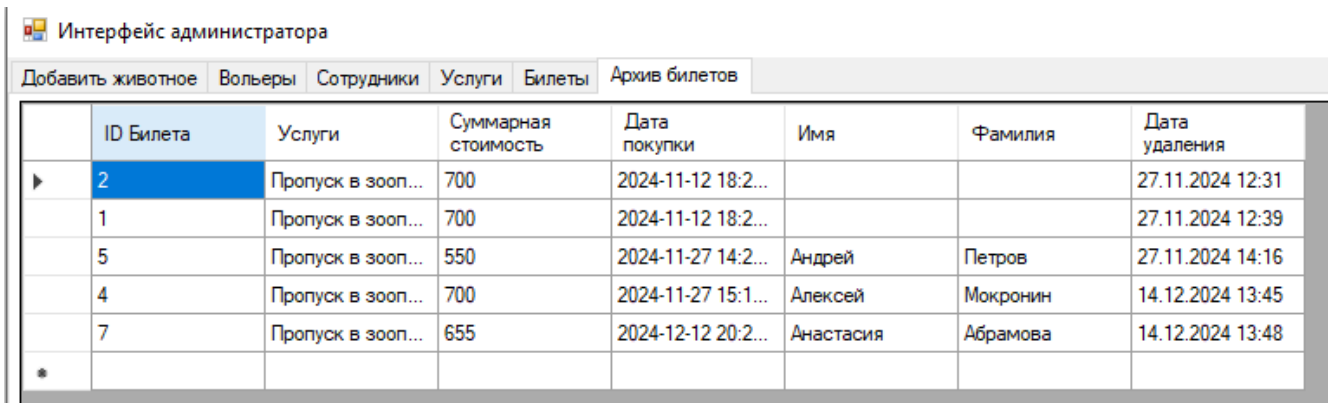


Рисунок 35 – Архив билетов

3.3. Руководство программиста

3.3.1 Введение

Приложение разработано на языке C# в среде разработки Visual Studio с использованием СУБД SQLite. Программа предоставляет функционал для управления данными зоопарка, включая билеты, услуги, сотрудников, вольеры и архив. Приложение состоит из нескольких форм, каждая из которых отвечает за определённый функционал.

3.3.2 Установка и настройка

Необходимые компоненты:

1) Visual Studio 2019 или новее.

2) SQLite NuGet Package (System.Data.SQLite).

3) SQLite Manager (для управления базой данных, опционально).

Установка:

1. Откройте Visual Studio.
2. Откройте проект "АИС зоопарк".
3. Проверьте, установлен ли пакет System.Data.SQLite через Управление NuGet-пакетами. Если его нет, добавьте вручную:
4. Перейдите в Инструменты > Управление NuGet-пакетами.
5. Найдите и установите System.Data.SQLite.
6. Убедитесь, что файл базы данных (zoo.db) подключён к проекту. Он должен быть расположен в корневой папке проекта.

Запуск проекта:

1. Убедитесь, что файл базы данных SQLite доступен и путь к нему указан правильно в коде.
2. Соберите проект (Ctrl + Shift + B).
3. Запустите приложение (F5).

3.3.3 Структура приложения

Файлы проекта:

Form1.cs – Форма выбора пользователя.

Form2.cs – Форма авторизации.

Form3.cs – Форма пользователя.

Form4.cs – Форма администратора.

Program.cs – Точка входа в приложение.

Дополнительные файлы:

App.config: настройки конфигурации приложения.

zoo.db: файл базы данных SQLite.

3.3.4 Описание используемых методов.

1) Form1.cs

- Конструктор Form1: инициализирует форму авторизации и настраивает внешний вид кнопок.
- Метод admin_Click: обрабатывает нажатие кнопки "Войти как администратор".
- Метод user_Click: обрабатывает нажатие кнопки "Войти как пользователь".

2) Form2.cs

- Конструктор Form2: инициализирует форму, настраивает визуальный стиль текстовых полей и кнопки
- Метод buttonLogin_Click: обрабатывает нажатие кнопки входа: получает данные из текстовых полей (логин и пароль). Проверяет введённые данные с помощью метода AuthenticateUser. Если пользователь авторизован: определяет, является ли пользователь администратором (username ==

"admin"). Открывает соответствующую форму (администратор или пользователь) как модальное окно. Закрывает текущую форму. Если данные некорректны, выводит сообщение об ошибке.

- Метод AuthenticateUser: проверяет логин и пароль в базе данных SQLite: выполняет SQL-запрос: "SELECT COUNT (1) FROM users WHERE username = @username AND password = @password". Использует параметризованный запрос для защиты от SQL-инъекций. Возвращает true, если пользователь существует, и false в противном случае. Обрабатывает ошибки подключения к базе данных и выводит их в MessageBox.

3) Form3.cs

- Конструктор UserForm: инициализирует форму пользователя. Выполняет следующие действия: загружает доступные услуги в checkedListBoxServices с помощью метода LoadServices. Загружает списокживотных в ComboBox с помощью метода LoadAnimals. Подключает обработчик checkedListBoxServices_ItemCheck для изменения состояния услуг. Инициализирует общую стоимость с помощью метода UpdateTotalCost.
- Метод checkedListBoxServices_ItemCheck: обрабатывает событие выбора или снятия выбора услуги в checkedListBoxServices: использует BeginInvoke для обновления состояния после выбора услуги. Вызывает метод UpdateTotalCost, чтобы пересчитать общую стоимость.
- Метод LoadAnimals: загружает список животных из базы данных SQLite и добавляет их в ComboBox: выполняет подключение к базе данных. Выполняет SQL-запрос для получения данных о животных.Добавляет названия животных в выпадающий список.
- Метод buttonViewAnimalInfo_Click: обрабатывает нажатие кнопки просмотра информации о животном: получает выбранное животное из ComboBox.

- Выполняет SQL-запрос к базе данных для получения информации о животном.
- Отображает результат в текстовом поле или окне сообщений.
- Класс ServiceItem: служебный класс для хранения данных об услуге: Содержит свойства для названия услуги и её стоимости.Используется при загрузке услуг и расчёте общей стоимости.
- Метод UpdateTotalCost: пересчитывает общую стоимость выбранных услуг: итеративно проверяет, какие услуги выбраны в checkedListBoxServices. Суммирует стоимость выбранных услуг. Отображает общую стоимость в текстовом поле или метке.
- Метод LoadServices: загружает список доступных услуг из базы данных SQLite: выполняет подключение к базе данных. Выполняет SQL-запрос для получения списка услуг. Добавляет услуги в checkedListBoxServices как элементы с названиями и их стоимостью.
- Метод buttonPurchase_Click: обрабатывает нажатие кнопки покупки: проверяет, выбраны ли услуги и животное.Формирует текст с данными о покупке (услуги, стоимость, выбранное животное). Вызывает метод SaveTicketInfoToFile для сохранения информации о билете. Выводит подтверждение об успешной покупке.
- Метод SaveTicketInfoToFile: сохраняет информацию о билете в файл: принимает строку ticketInfo с информацией о покупке. Записывает эту строку в файл на диске, включая дату и время покупки. Уведомляет пользователя о том, что билет успешно сохранён.

4) Form4.cs

- Метод InitializeGenderComboBox: инициализирует ComboBox для выбора пола, добавляя значения "Male" и "Female".
- Метод LoadWorkers: загружает данные о работниках и отображает их в соответствующем элементе управления (например, ComboBoxWorker).

- Метод LoadAnimals: загружает список животных из базы данных или другого источника и заполняет таблицу dataGridViewAnimals.
- Метод LoadEmployees: загружает список сотрудников из базы данных или другого источника и заполняет таблицу dataGridViewEmployees.
- Метод InitializeEnclosureTypeComboBox: инициализирует ComboBox для выбора типа вольера с возможными значениями: "Cage", "Pen", "Aquarium".
- Метод LoadEnclosures: загружает данные о вольерах и отображает их в таблице dataGridViewEnclosures.
- Метод LoadServices: загружает данные об услугах и отображает их в таблице dataGridViewServices.
- Метод LoadTickets: загружает список билетов и отображает их в таблице dataGridViewTickets.
- Метод LoadArchivedTickets: загружает данные об архивных билетах и отображает их в таблице dataGridViewArchive.
- Метод LoadPositions: загружает доступные должности сотрудников и заполняет ComboBoxPosition.

Обработчики событий:

- Метод textBoxAnimalName_KeyPress: проверяет ввод текста в поле имени животного, запрещая вводить цифры или другие недопустимые символы.
- Метод textBoxAge_KeyPress: проверяет ввод возраста, разрешая только числовые значения.
- Метод textBoxCountry_KeyPress: проверяет ввод страны, запрещая использование цифр.
- Метод textBoxEmployeeName_KeyPress: проверяет ввод имени сотрудника, исключая ввод цифр.
- Метод txtSearchDate_KeyPress: проверяет корректность ввода даты поиска.

- Метод textBoxWeight_KeyPress: проверяет ввод веса, разрешая только числовые значения.
- Метод textBoxHeight_KeyPress: проверяет ввод роста, разрешая только числовые значения.
- Метод textBoxLifetime_KeyPress: проверяет ввод продолжительности жизни, разрешая только числовые значения.
- Метод txtFirstName_KeyPress: проверяет ввод имени клиента, исключая ввод недопустимых символов.
- Метод txtLastName_KeyPress: проверяет ввод фамилии клиента, исключая ввод недопустимых символов.
- Метод textBoxEmployeeAge_KeyPress: проверяет ввод возраста сотрудника, разрешая только числовые значения.
- Метод textBoxEnclosureFood_KeyPress: проверяет ввод количества еды для вольера, разрешая только числовые значения.
- Метод textBoxEmployeeExperience_KeyPress: проверяет ввод опыта сотрудника в годах, разрешая только числовые значения.
- Метод textBoxEmployeeSalary_KeyPress: проверяет ввод заработной платы, разрешая только числовые значения.
- Метод dataGridViewEnclosures_SelectionChanged: обрабатывает событие выбора строки в таблице вольеров.
- Метод dataGridViewServices_SelectionChanged: обрабатывает событие выбора строки в таблице услуг.
- Метод Form_Load: выполняет действия при загрузке формы, такие как заполнение данных, настройка элементов управления.
- Метод LoadAnimals: загружает список животных из базы данных и заполняет соответствующие элементы управления.
- Метод InitializeEnclosureTypeComboBox: инициализирует ComboBox для выбора типа вольера с возможными значениями (например, "Cage", "Pen", "Aquarium").

- Метод LoadEnclosures: загружает данные о вольерах и отображает их в таблице dataGridViewEnclosures.
- Метод ClearServiceFields: очищает все поля, связанные с услугами.
- Метод UpdateRowNumbersEnclosures: обновляет номера строк в таблице вольеров для корректного отображения данных.
- Метод LoadEmployees: загружает список сотрудников из базы данных и отображает их в таблице dataGridViewEmployees.
- Метод buttonAddAnimal_Click(object sender, EventArgs e): обрабатывает нажатие кнопки "Добавить животное", добавляя новое животное в базу данных.
- Метод buttonAddEmployee_Click(object sender, EventArgs e): обрабатывает нажатие кнопки "Добавить сотрудника", добавляя нового сотрудника в базу данных.
- Метод AddAnimalToDatabase: добавляет информацию о животном в базу данных (статический метод).
- Метод AddEmployeeToDatabase: добавляет информацию о сотруднике в базу данных (статический метод).
- Метод ClearAnimalFields: очищает все поля, связанные с данными животного.
- Метод ClearEmployeeFields: очищает все поля, связанные с данными сотрудника.
- Метод buttonDeleteAnimal_Click: обрабатывает нажатие кнопки "Удалить животное", удаляя выбранное животное из базы данных.
- Метод buttonDeleteEmployee_Click: обрабатывает нажатие кнопки "Удалить сотрудника", удаляя выбранного сотрудника из базы данных.
- Метод LoadServices: загружает список услуг из базы данных и отображает их в таблице dataGridViewServices.
- Метод buttonAddEnclosure_Click: обрабатывает нажатие кнопки "Добавить вольер", добавляя новый вольер в базу данных.

- Метод LoadTickets(): загружает данные о билетах и отображает их в таблице dataGridViewTickets.
- Метод UpdateRowNumbers: обновляет номера строк в таблицах для корректного отображения данных.
- Метод buttonAddService_Click: обрабатывает нажатие кнопки "Добавить услугу", добавляя новую услугу в базу данных.
- Метод buttonDeleteService_Click: обрабатывает нажатие кнопки "Удалить услугу", удаляя выбранную услугу из базы данных.
- Метод LoadArchivedTickets: загружает данные об архивных билетах и отображает их в таблице dataGridViewArchive.
- Метод buttonDeleteTicket_Click(object sender, EventArgs e): обрабатывает нажатие кнопки "Удалить билет", удаляя выбранный билет из базы данных.
- Метод buttonSearch_Click(object sender, EventArgs e): выполняет поиск по введённым данным и отображает результат в таблице.
- Метод buttonUpdate_Click(object sender, EventArgs e): обрабатывает нажатие кнопки "Обновить", выполняя действия по обновлению данных.
- Метод buttonSaveChanges_Click(object sender, EventArgs e): сохраняет изменения, внесенные в текущие данные.
- Метод ResetSearch_Click(object sender, EventArgs e): сбрасывает параметры поиска и обновляет таблицу с исходными данными.
- Метод buttonEditEmployee_Click(object sender, EventArgs e): инициирует процесс редактирования данных сотрудника.
- Метод buttonSave_Click(object sender, EventArgs e): обрабатывает нажатие кнопки "Сохранить", выполняя действия по сохранению данных.
- Метод buttonSearchEmploye_Click(object sender, EventArgs e): выполняет поиск сотрудников по введённым данным.

- Метод ResetEmployeeSearch_Click(object sender, EventArgs e): сбрасывает параметры поиска сотрудников и обновляет таблицу с исходными данными.
- Метод buttonFinishEditing_Click(object sender, EventArgs e): завершает процесс редактирования данных и сохраняет изменения.
- Метод dataGridViewTickets_CellClick(object sender, DataGridViewCellEventArgs e): обрабатывает событие клика по ячейке таблицы dataGridViewTickets.
- Метод SearchByDate_Click(object sender, EventArgs e): выполняет поиск данных по указанной дате.
- Метод ClearInputFields: очищает все вводимые поля формы.
- Метод buttonCreateTicket_Click(object sender, EventArgs e): обрабатывает нажатие кнопки "Создать билет", добавляя новый билет в базу данных.
- Метод btnSearchAnimal_Click(object sender, EventArgs e): выполняет поиск животного по введённым данным.
- Метод buttonResetFilters_Click(object sender, EventArgs e): сбрасывает применённые фильтры.
- Метод buttonreset_Click(object sender, EventArgs e): сбрасывает введённые данные в определённой форме.
- Метод buttonRefreshArchive_Click(object sender, EventArgs e): обновляет данные архива и отображает их в таблице.
- Метод IsValidName(string name): проверяет корректность имени, исключая недопустимые символы.

4. Тестирование АИС

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.

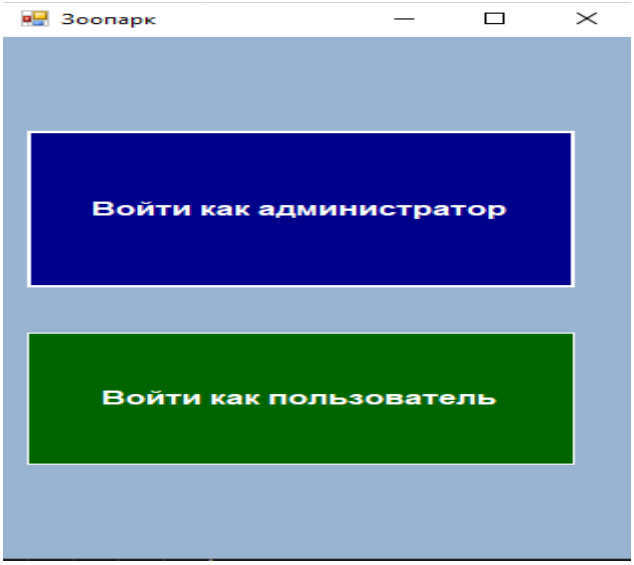


Рисунок 36 – форма 1

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

Далее если выбрать “Администратор” выдастся окно для ввода логина и пароля.

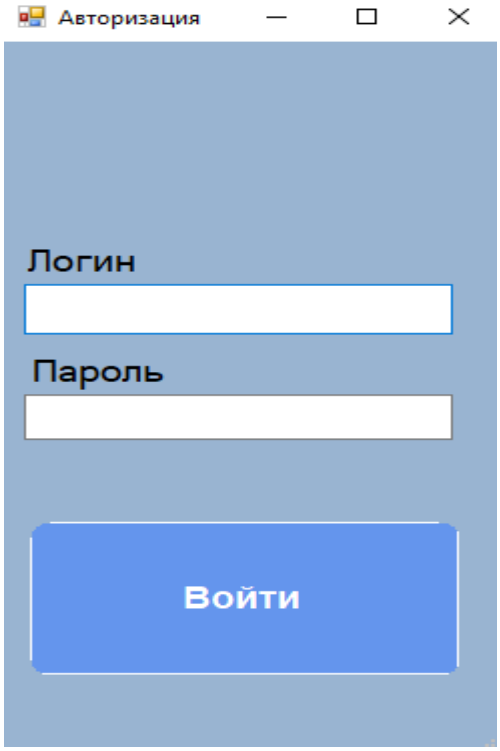


Рисунок 37 – форма 2

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

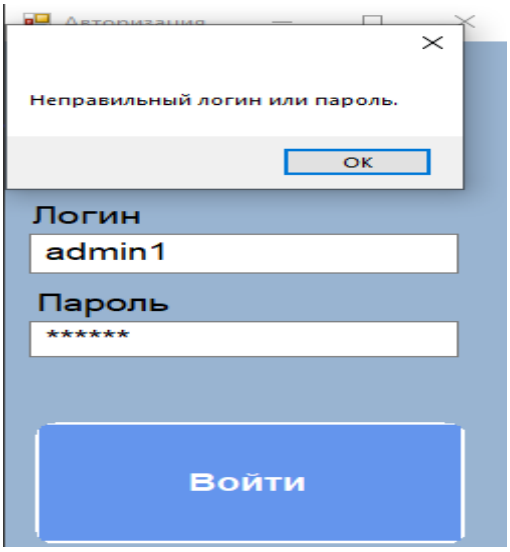


Рисунок 38 – обработчик ошибок для формы 2

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

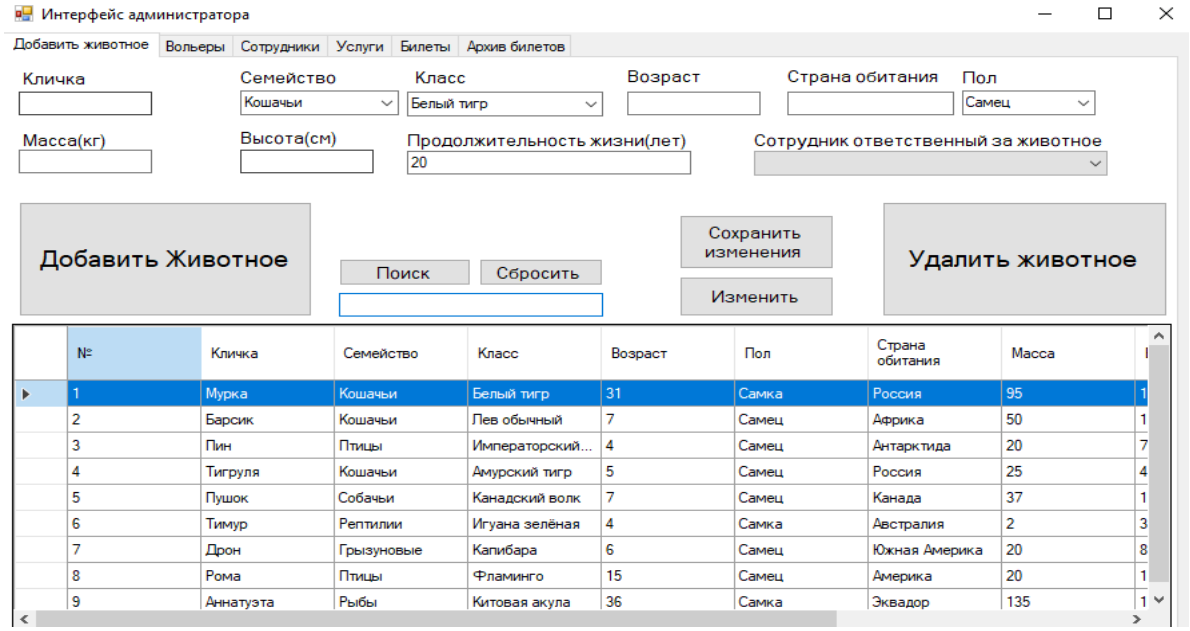


Рисунок 39 – интерфейс для администратора

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.

Начнем с первой вкладки, которая называется “Добавить животное”. На ней администратор может добавлять и удалять животных, так же реализована функция поиска по названию и редактирования существующих записей. При вводе несуществующей клички животного появится сообщение что такое животное не найдено.

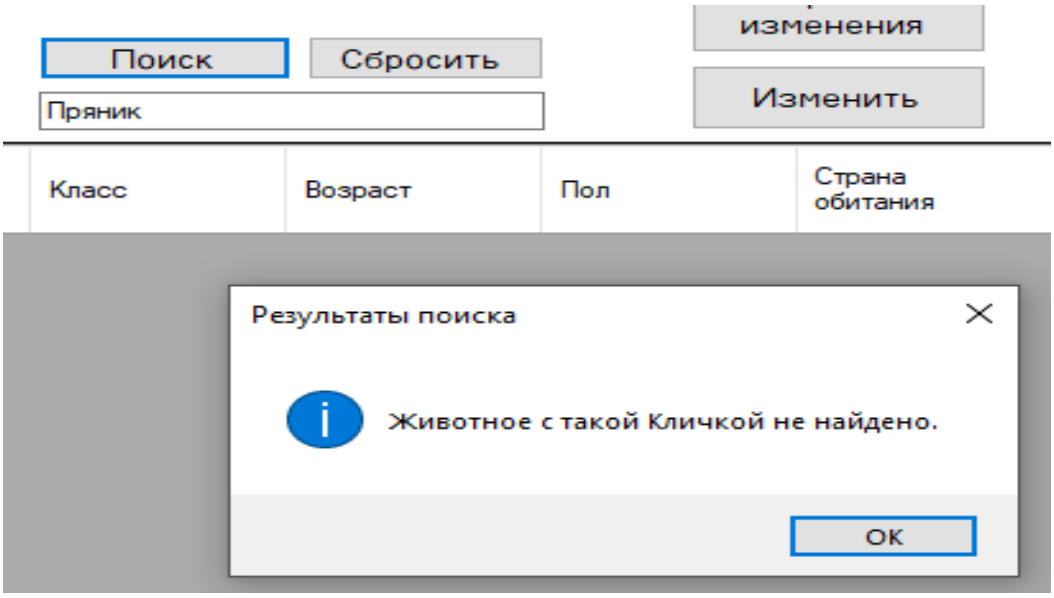


Рисунок 40 – запись не найдена

Для добавления животного в таблицу необходимо заполнить все поля, если поля не заполнены предусмотрен обработчик событий, который показан на рисунке 41.

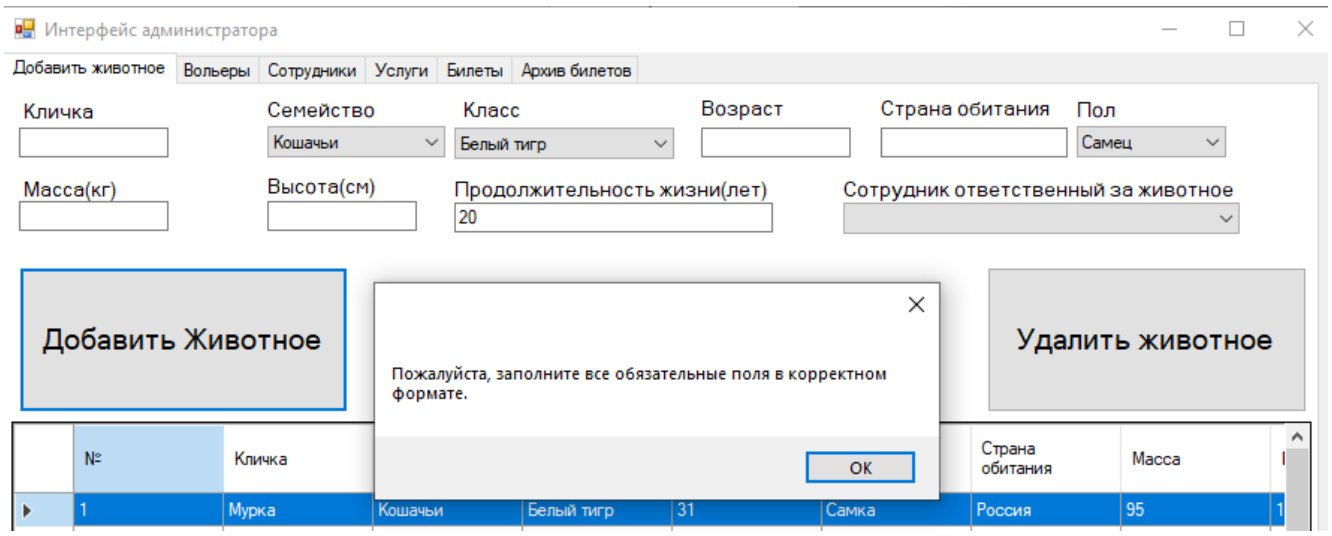


Рисунок 41 –корректность данных

Так же для удобства пользования предусмотрена следующая логика: при выборе определенного семейства автоматически определяются возможные классы по словарю, это показано на рисунке 42.

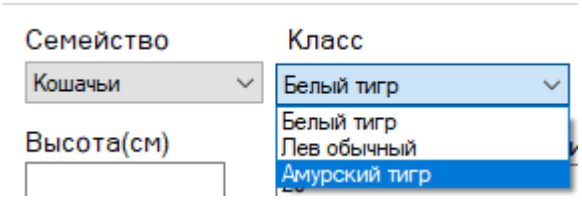


Рисунок 42 – ограничение выбора

Далее так же для удобства при выборе определенного класса, автоматически выбирается его продолжительность жизни.

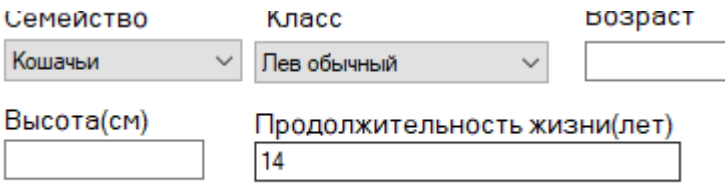


Рисунок 43 – автоматизация

На вкладке “Вольеры” реализован поиск по названию животных, если ввести несуществующее получим сообщение.

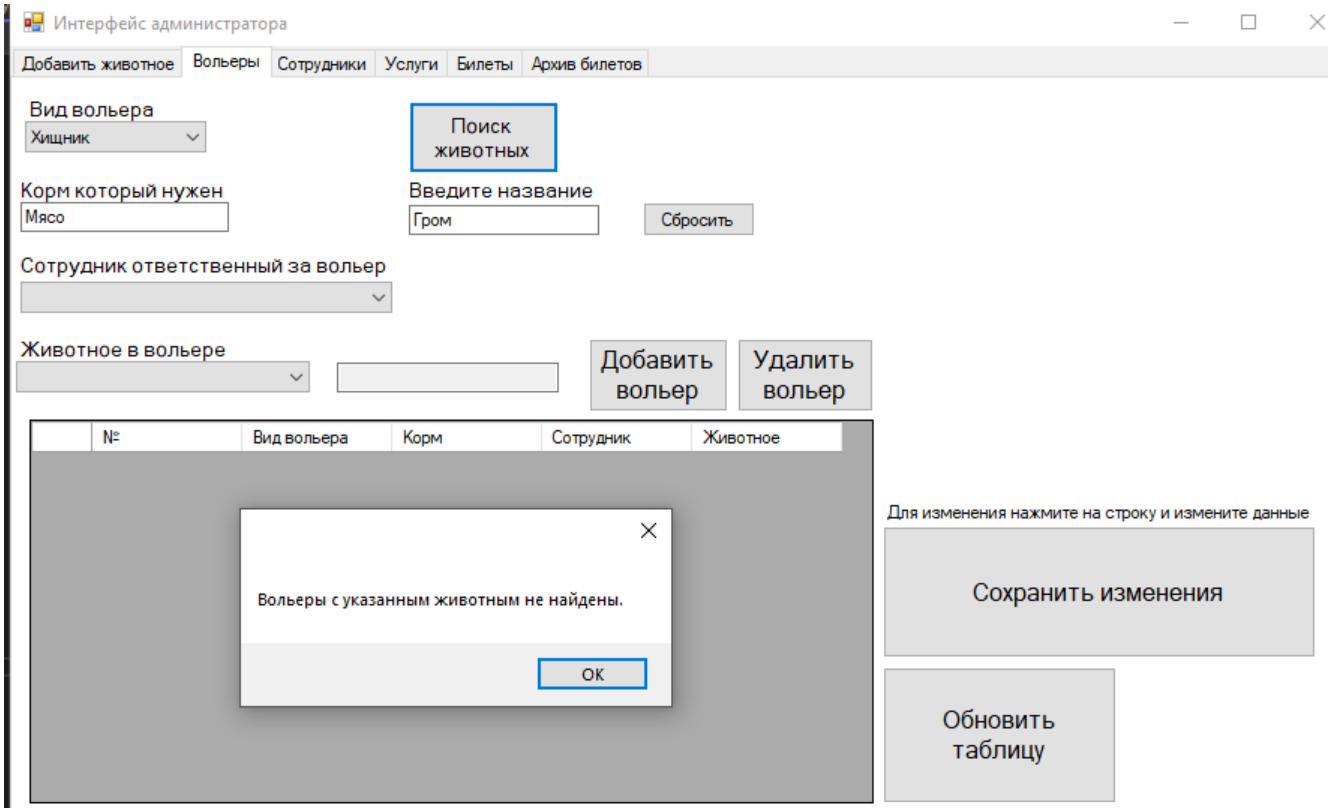


Рисунок 44 – поиск вольеров

Так же реализован вывод сообщения при успешном сохранении изменений в таблице, который показан на рисунке 45.

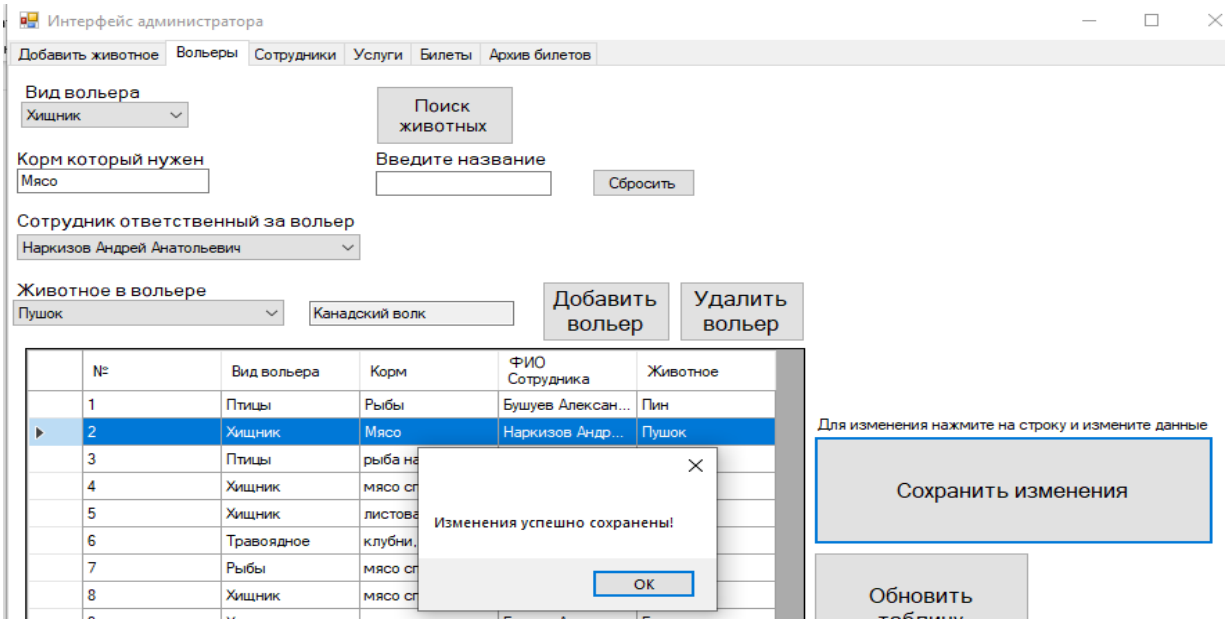


Рисунок 45 – изменения вольеров

На вкладке “Сотрудники” тоже реализован поиск, только по фамилии сотрудника, при вводе несуществующей фамилии появляется окно.

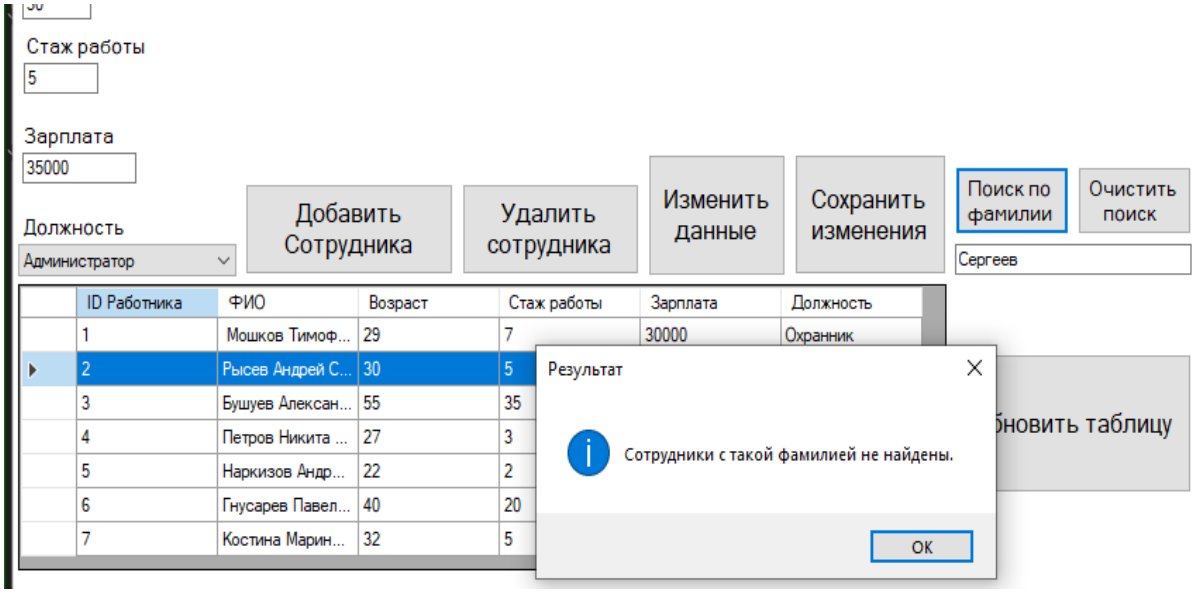


Рисунок 46 – поиск сотрудника

На вкладке услуги реализована немного другая логика, пользователю позволено вводить некорректные данные, но сохранить в таком формате он их не сможет, появится ошибка с просьбой заполнить все корректно

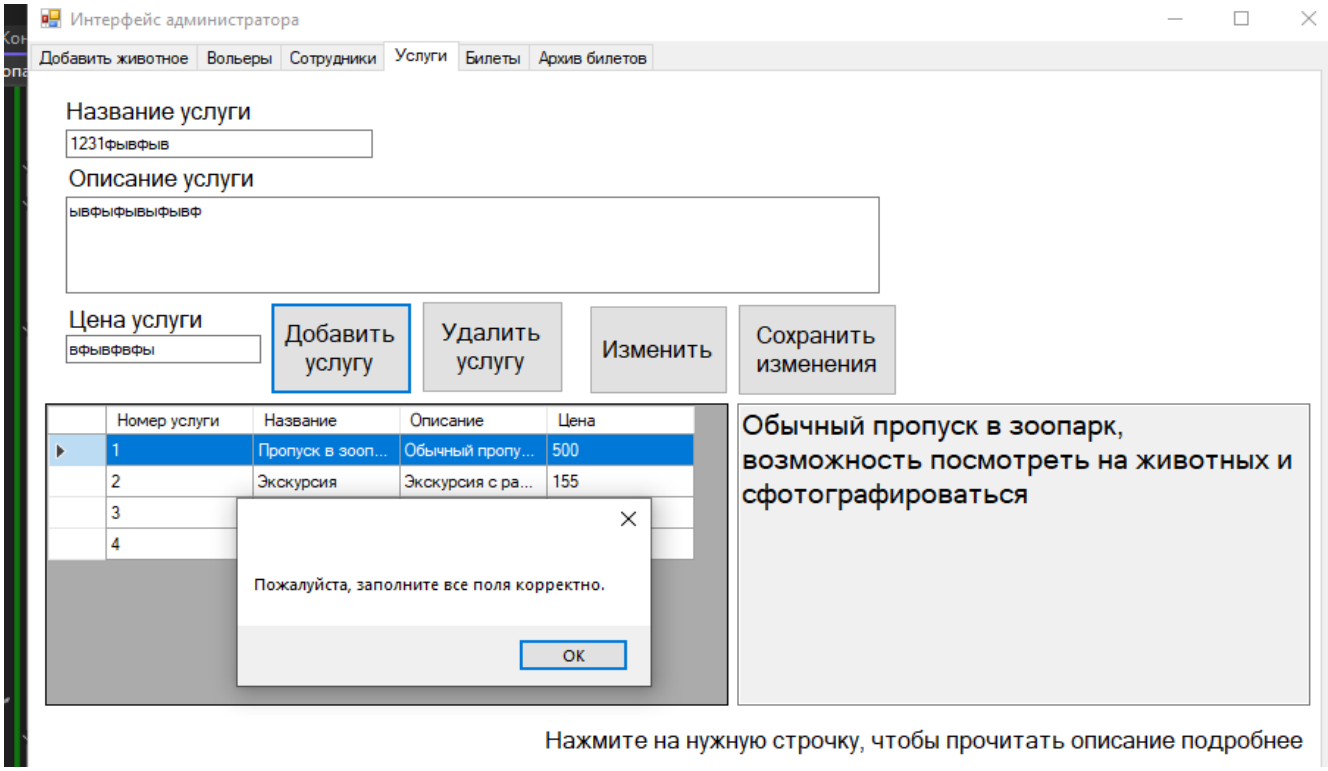


Рисунок 47 – создание услуги

На вкладке “Билеты” реализован обработчик некорректного ввода даты поиска билета и для некорректного ввода имени или фамилии посетителя для поиска. Эти обработчики показаны на рисунках 46–47.

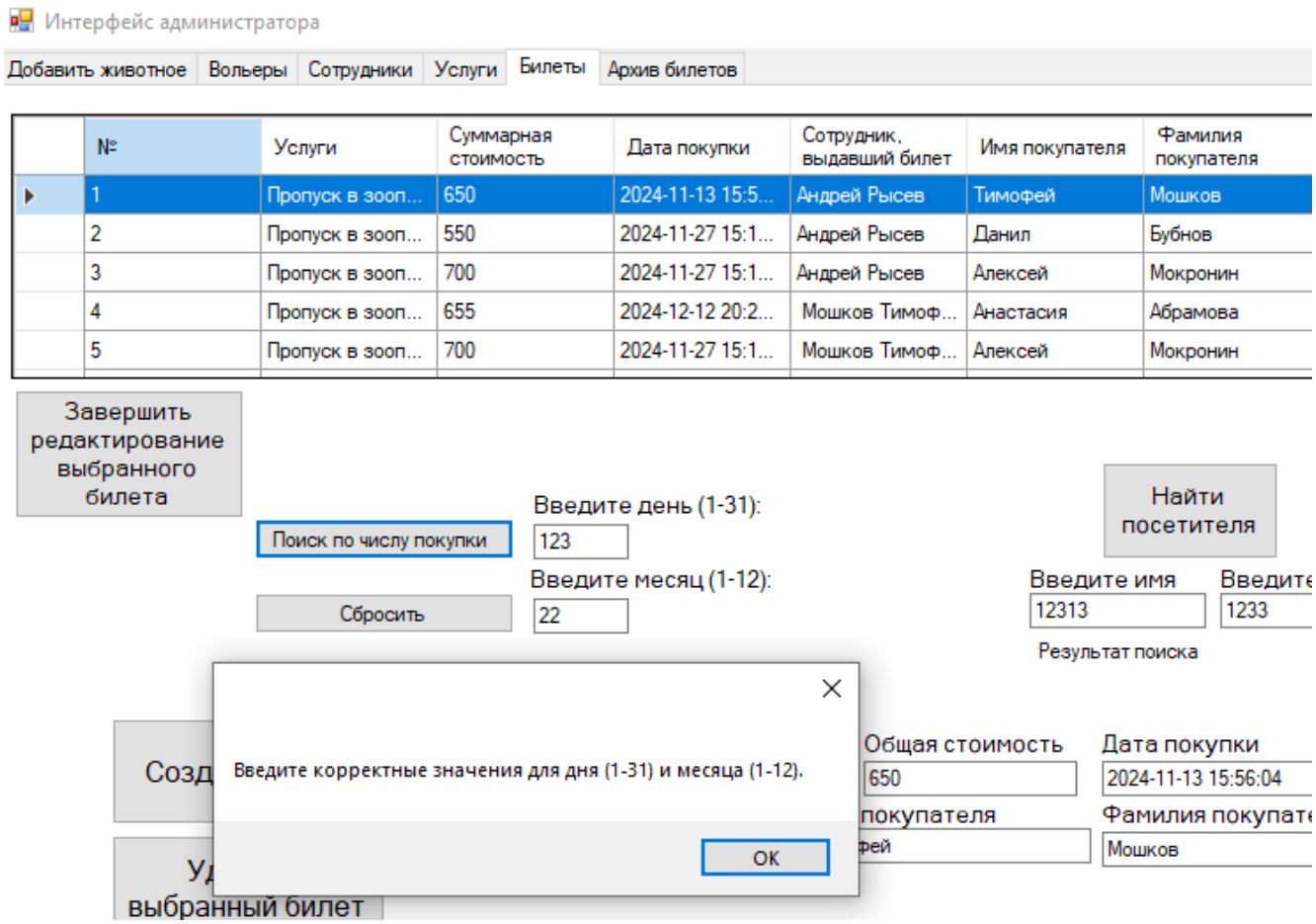


Рисунок 48 – поиск по дате

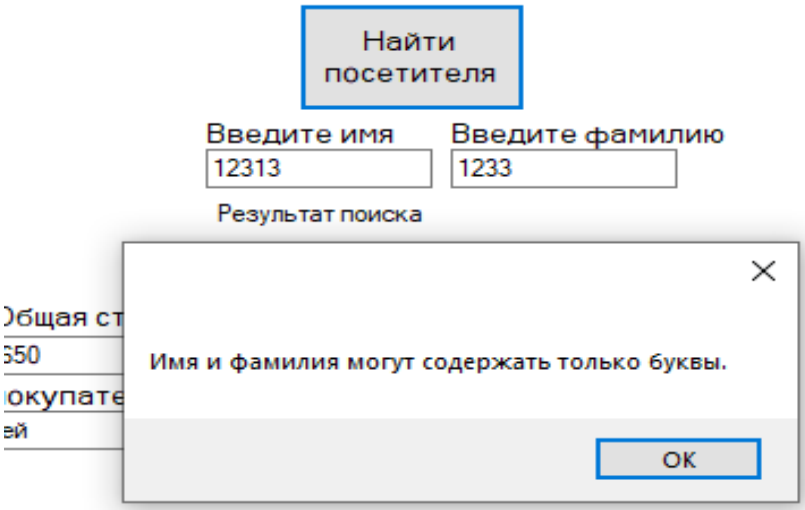


Рисунок 49 – поиск посетителя

Далее для пользователя предусмотрено всего 2 сценария, успешная покупка билета и ошибка при вводе данных, визуальное оформление этой логики показано на рисунках 50–51.

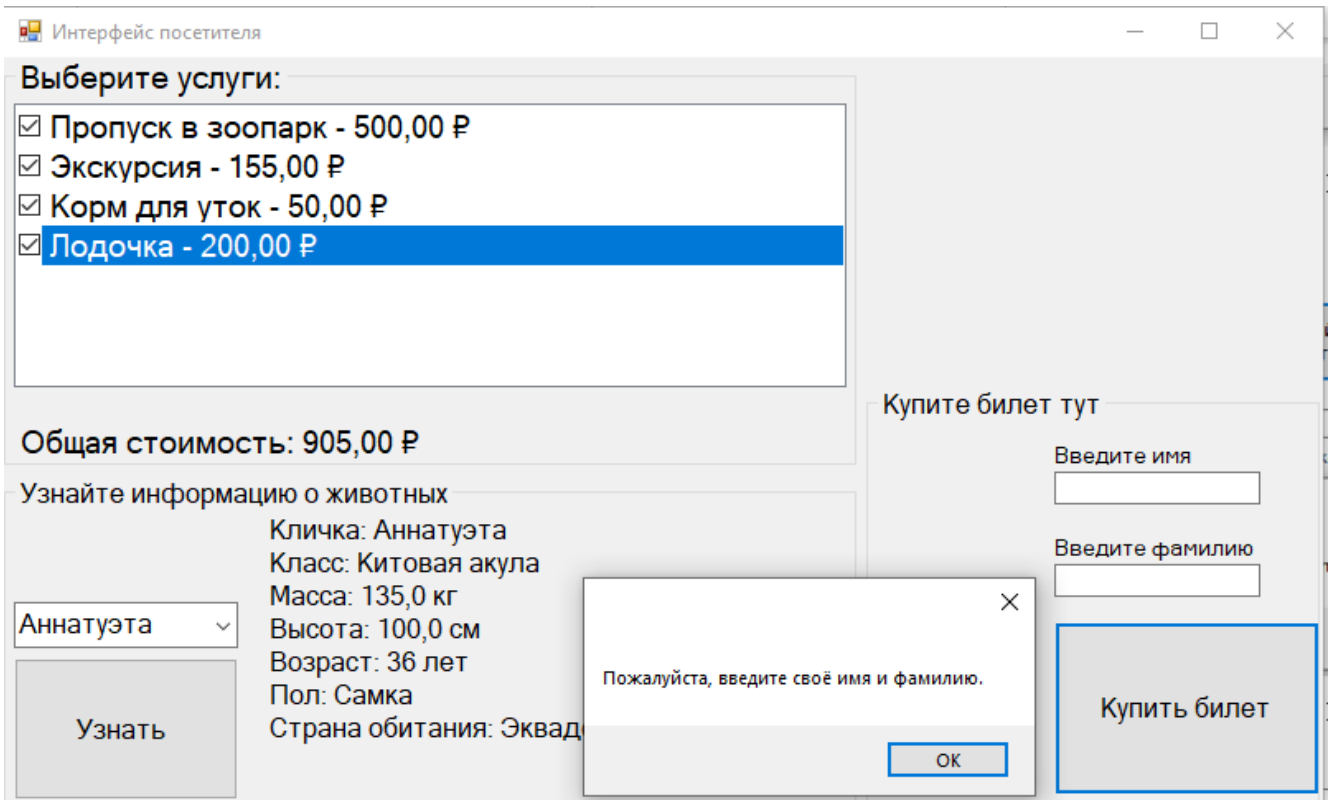


Рисунок 50 – ошибка при покупке

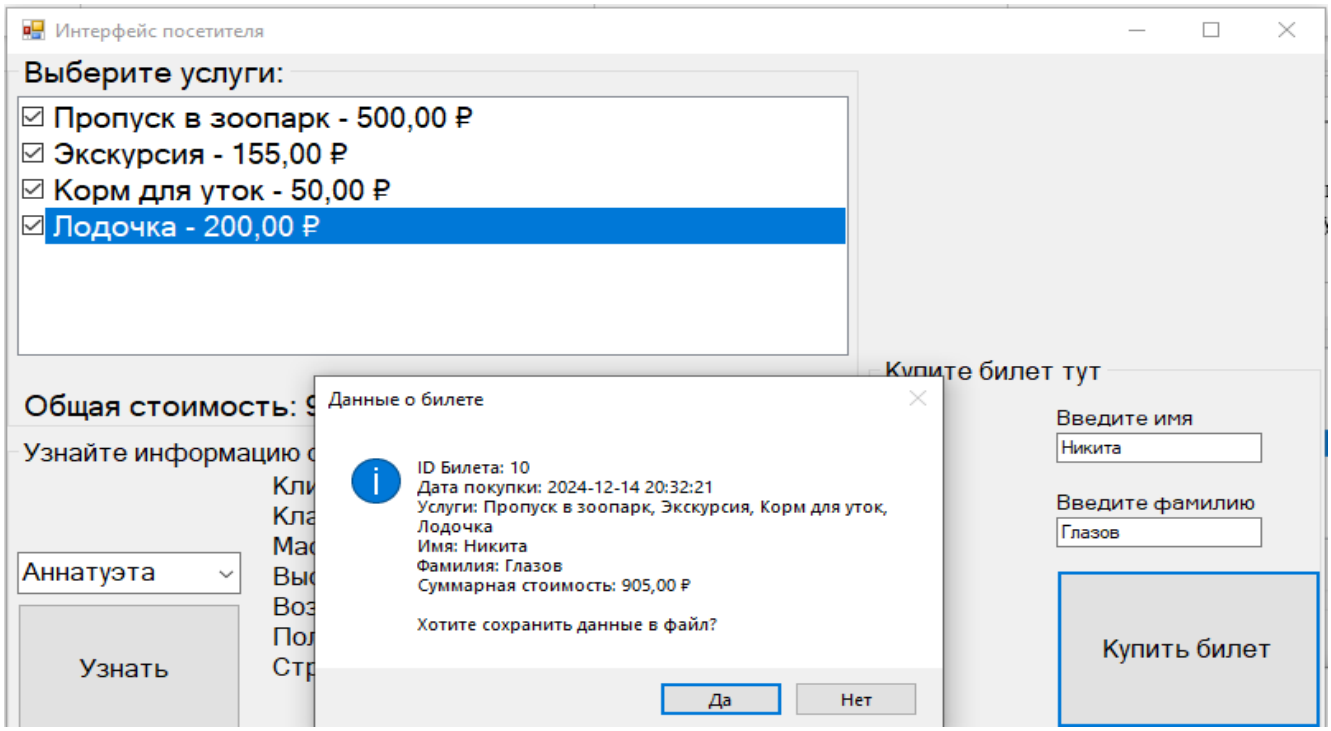


Рисунок 51 – успешная покупка

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.

Список литературы

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

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

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

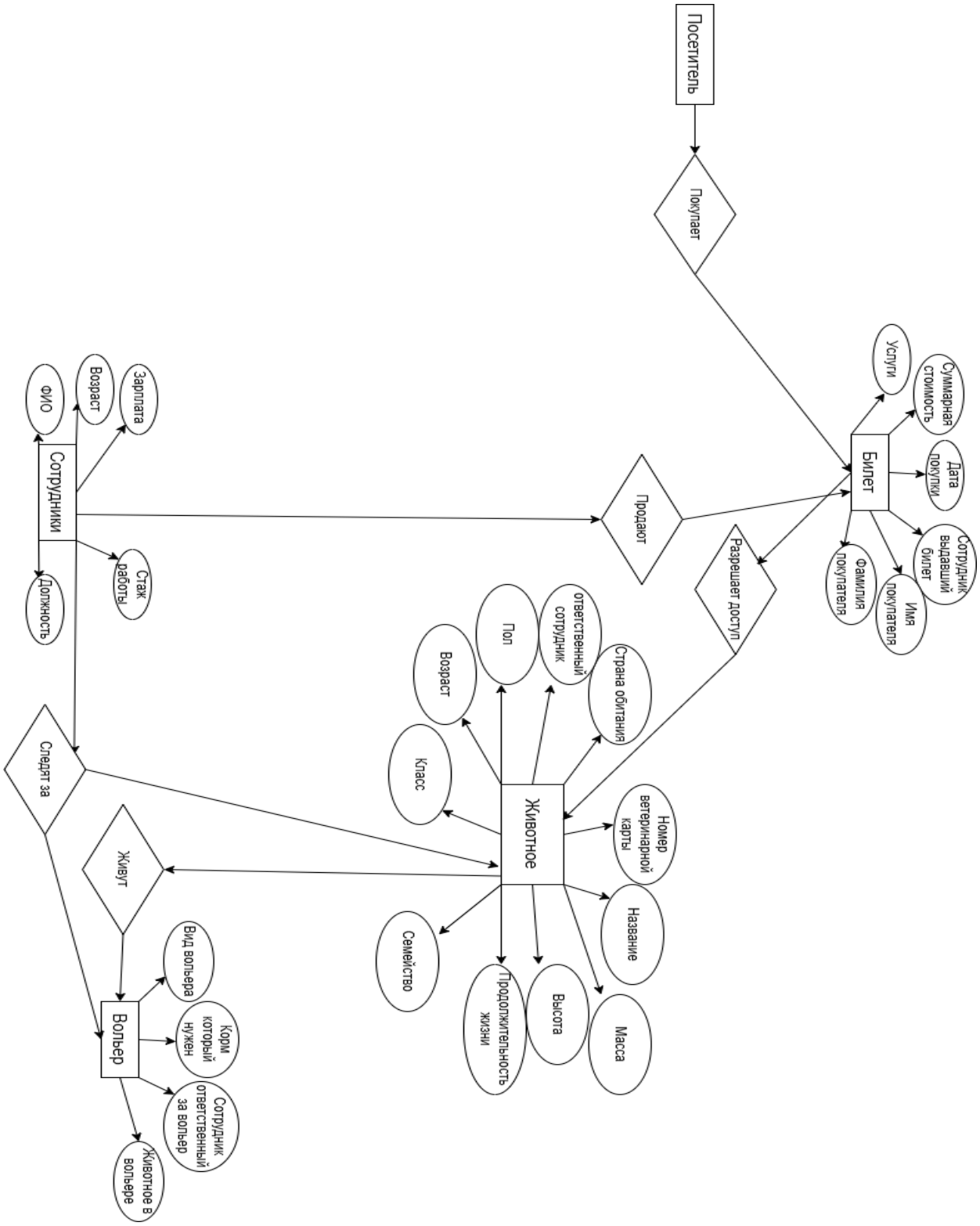
4) Соколова М. В., Павлов И. А. "Основы зоотехники: содержание и учет животных". — СПб.: Профессия, 2017. — 208 с.

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

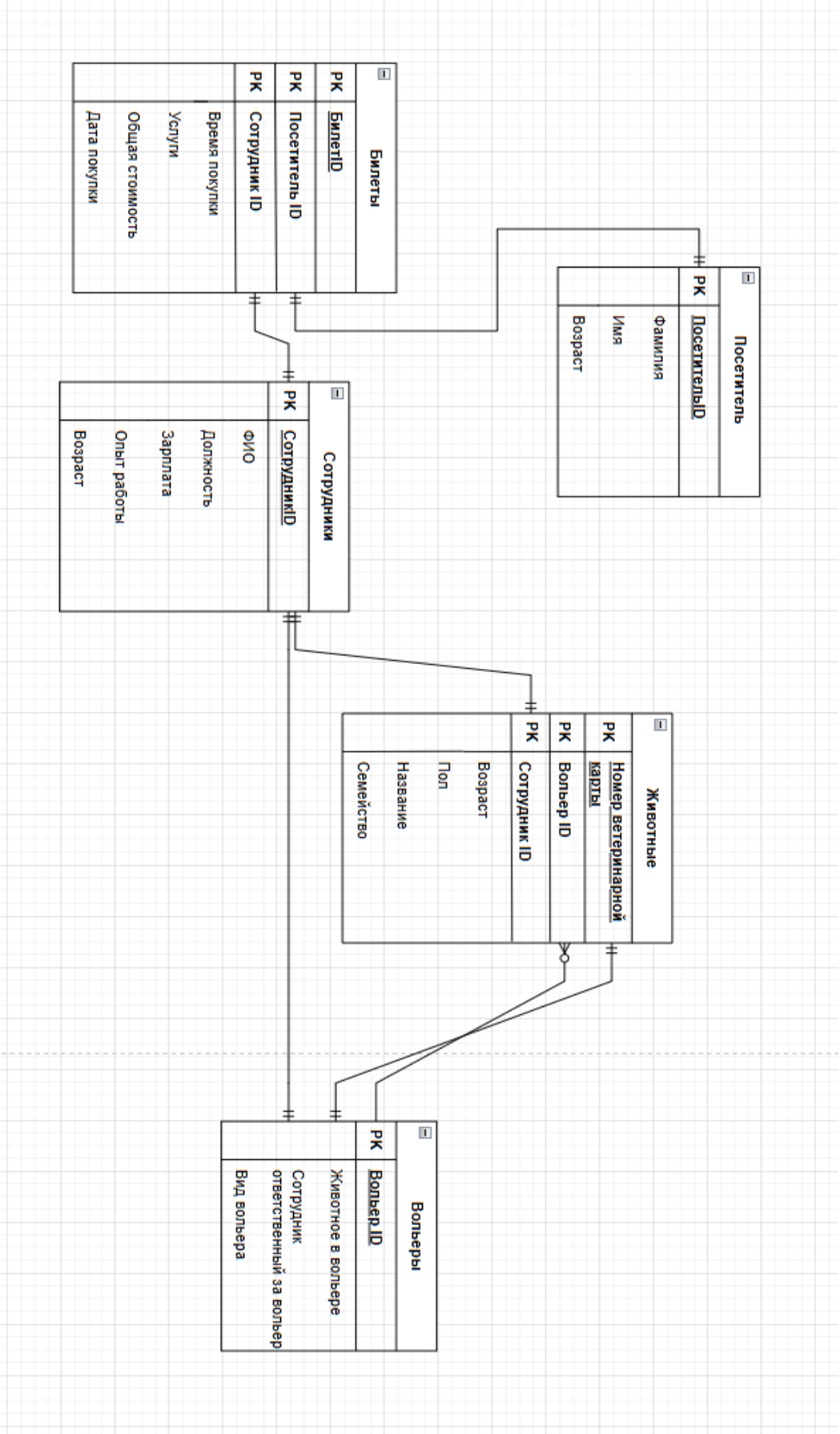
6) Ульман Д. Джеффри, Уидом Дж. "Основы систем баз данных". — 7-е изд. — М.: Вильямс, 2019. — 1248 с.

Приложение А: Модели данных

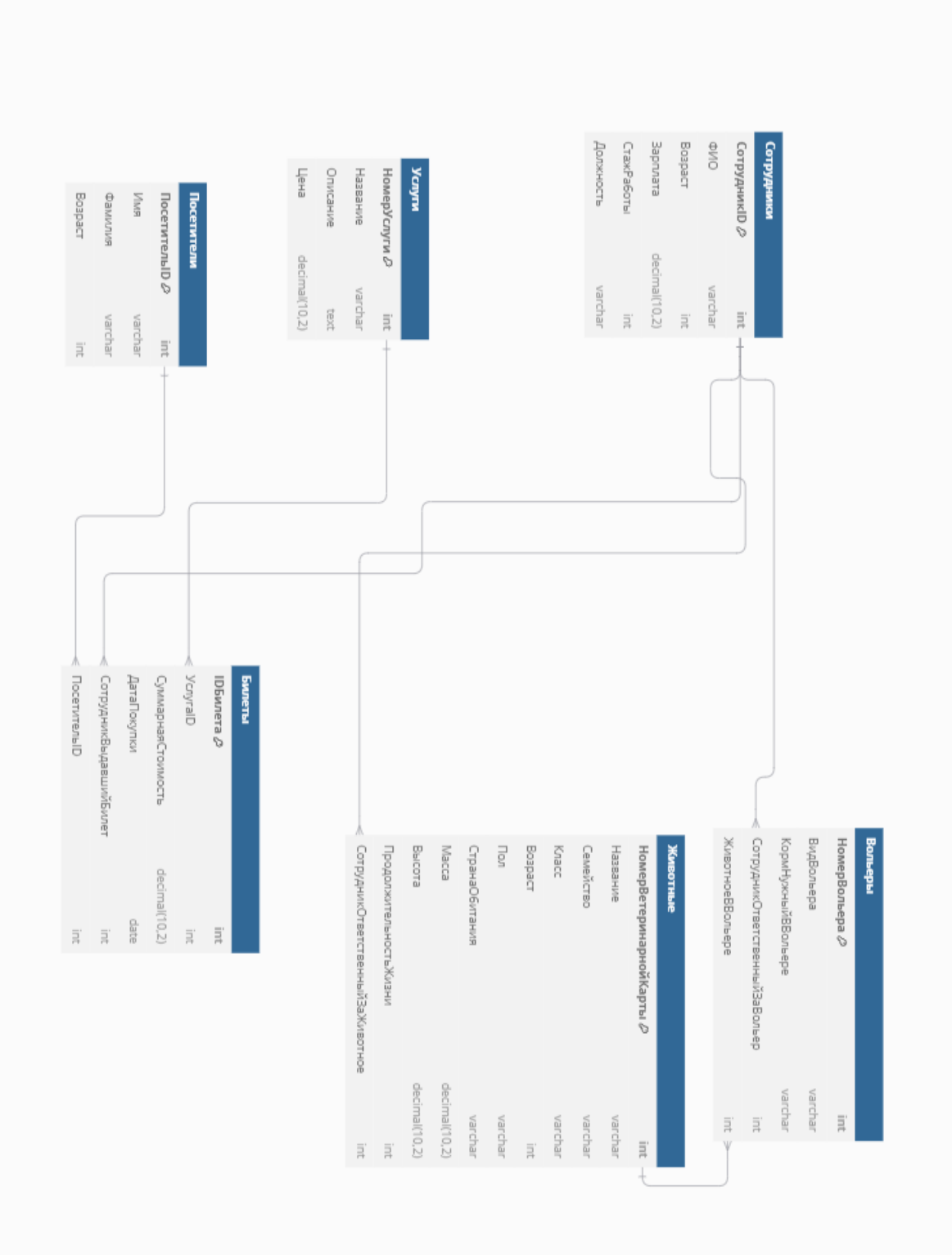
Концептуальная модель.



Логическая модель.



Физическая модель.



Приложение Б: «Ссылка на GitHub»
Исходный код представлен по ссылке:

https://github.com/BirdManl/-zoo-