

Федеральное агентство морского и речного транспорта

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Государственный университет морского и речного флота имени адмирала С.О. Макарова»

Институт водного транспорта

Кафедра вычислительных систем и информатики

**Курсовая работа**

по дисциплине «Технологии программирования»

Тема: «Объектно-ориентированный подход к программированию и технология .NET /ADO.NET(С++ или C#)»

Вариант №27

«Разработка приложения для предметной области «Учёт домашних

животных в фермерском хозяйстве»»

|  |  |
| --- | --- |
|  | **Выполнил**: студент гр. ИС-23  Перфильева О.К., Дель А.А.  **Руководитель**: доц. каф. ВСИ  к.г.н. Балса Алдрин Раульевич |
|  |  |

Санкт-Петербург

2022

**СОДЕРЖАНИЕ**

**[ВВЕДЕНИЕ](#_Toc30593701)** [2](#_Toc30593701)

[1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОБЪЕКТНО-ОРЕНТИРОВАНОГО ПРОГРАММИРОВАНИЯ В DOT.NET C# 3](#_Toc30593702)

[1.1 Особенности язык С# 4](#_Toc30593703)

[1.2 Особенности NET Framework 6](#_Toc30593704)

[1.3 Особенности технологи ADO.NET 7](#_Toc30593705)

[1.4 Особенности SQL 8](#_Toc30593706)

**[2. РЕАЛИЗАЦИЯ ПРОЕКТА](#_Toc30593707)** [11](#_Toc30593707)

[2.1. Постановка задачи 11](#_Toc30593708)

[2.2. Описание архитектуры программы с использованием UML диаграммы. 13](#_Toc30593709)

[2.3. Руководство пользователя 21](#_Toc30593710)

[2.4. Руководство администратора 24](#_Toc30593711)

**[ЗАКЛЮЧЕНИЕ](#_Toc30593712)** [33](#_Toc30593712)

**[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ](#_Toc30593713)** [34](#_Toc30593713)

**[ПРИЛОЖЕНИЕ А](#_Toc30593714)** [35](#_Toc30593714)

**ПРИЛОЖЕНИЕ Б**................................................................................................36

**ВВЕДЕНИЕ**

Приложения в век информационных технологий стали незаменимыми спутниками человека. Их используют все, у кого есть техника, будь то переносная или стационарная. На данный момент приложения можно скачивать уже не только на смартфон и компьютер. Сейчас и в часы можно встроить различные калькуляторы, прогнозы погоды, курсы валют, конвертеры и так далее.  
 Приложения бывают разные: предустановленные и скачиваемые. Первые разработчики по умолчанию устанавливают в заводскую комплектацию устройства. Второй тип приложений же надо устанавливать вручную через специализированные магазины приложений, которые, к слову, предустановлены на устройствах, поддерживающих установку дополнительного ПО.  
 Но зачем нужны приложения? Приложения созданы для того, чтобы предоставлять их пользователю дополнительные возможности. С помощью них можно просматривать, сохранять и воспроизводить информацию, взятую из интернета, делать заметки, общаться с другими пользователями, производить подсчеты и многое другое.   
 Приложения становятся все более востребованными в связи с ростом объёма информации в интернете. Пользователи хотят упрощенного взаимодействия с ней, как следствие появляется спрос, который нужно снижать предложением. Потому разработка приложений является неоспоримо важной составляющей деятельности любой крупной и не очень IT компании.  
 Целью курсовой работы является создание прототипа приложения «Учет домашних животных в фермерском хозяйстве» с возможностью хранения информации в базе данных устройства.

Задачами курсовой работы являются:

* + 1. Применить объектно-ориентированный подход к программированию;
    2. Изучить технологию .NET /ADO.NET;
    3. Разработать проектный план;
    4. Построить архитектуру приложения «Учет домашних животных в фермерском хозяйстве» с помощью диаграммы классов;
    5. Создать Windows-приложение «Учет домашних животных в фермерском хозяйстве» на основе Visual C#;

Для создания проекта были использованы следующие концепции программирования: объектно-ориентированное программирование (ООП), унифицированный язык моделирования (UML) и специализированный средства разработки программного обеспечения.

При разработке большинства программных проектов приоритетным стал объектно-ориентированный подход. Технология объектно-ориентированного программирования (ООП) - это дальнейшее развитие идей структурного и процедурного (модульного) программирования. Ее представляли в виде программных структур, прячущих в себя различные детали и тонкости реализации.

Применение визуальных объектно-ориентированных средств разработки позволило резко сократить время разработки благодаря использованию стандартных объектов API, ранее разработанных объектов и пр.

В качестве технологий для разработки прототипа приложения была выбрана платформа .NET (dotNet, произносят как «Дот-Нет») и языков программирования для неё - C# (C sharp, говорят «си шарп»), Delphi .NET и пр.

Для оформления были использованы диаграммы UML для графического представления и описания функционирования системы, разработанной средствами Window Forms на языке C#.

В дальнейшем будут даны точные определения всех рассмотренных терминов.  
 Данная курсовая работа состоит из 3 глав. В первой главе рассмотрена объектно-ориентированный подход к программированию, технология .NET и её объекты, особенности разработки приложений .NET и C#. Во второй главе унифицированный язык моделирования UML и создание с помощью него диаграммы классов. В третьей главе разрабатываем приложения с помощью перечисленных технологий по поставленным нам задачам.

1. **МОДЕЛИ И СПЕЦИФИКАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**1.1. Основные понятия ООП**

Основными понятиями ООП являются объекты и классы. Класс – это модель определенной физической системы, реального процесса или объекта реального мира. Класс является типом данных, определяемым пользователем. В классе задаются свойства и поведение какого-нибудь объекта в виде данных и функций (методов) для их обрабатывания [1].  
 Подобная технология подразумевает отказ от функций и процедур. В ООП программа представляется в виде совокупности объектов, обладающих аналогичными качествами и комплексом действий, которые возможно с ними производить.  
 Объект – это экземпляр соответствующего класса. Функции объекта, именуемые методами или функциями-членами, обычно специализированы для доступа к данным объекта. Доступ к данным исполняется с помощью вызова соответственного метода, который осуществляет чтение и возвращает спрашиваемое значение. Это говорит о том, что данные скрыты от внешнего воздействия и защищены от случайного изменения.  
 Отдельные объекты, что можно изменять без влияния на остальные части программы, упрощают вдобавок и внесение в программу изменений без опасности испортить их.  
 ООП характеризует 4 ключевых принципа:  
1) Абстракция – это процесс определения имеющийся характеристик определенной сущности, отличающих ее от другой сущности и значимых в рамках определенной задачи. Классы и объекты являются абстракциями.  
2) Инкапсуляция (encapsulation) – это скрытие элементов реализации, несуществующей с точки зрения рассматриваемой абстракции. Инкапсуляция имеет своё происхождение от деления модулей в некотором языке программирования на две части (на интерфейс и реализацию).  
Интерфейсная секция модуля описывает все объявления функций и процедур о возможных типах данных за границами самого модуля. Подобные функции и процедуры являются методом оказания услуг наружным клиентам. В другой секции модуль реализации включает программный код, который определяет конкретные способы реализации функций и процедур, объявленных в интерфейсной части.  
3) Наследование – это принцип на основе познаний преимущественно общей категории, который разрешен к применению для более приватной категории.  
4) Полиморфизм – это возможность объектов с подобной спецификацией обладать разной реализацией.  
Таким образом, ООП сильно упрощает ход организации и создания структуры программы.

**1.2. Унифицированный язык моделирования UML**

Написание кода на языке программирования - не единственный этап создания программы. Помимо программирования, важно заняться постановкой задачи, разработкой интерфейса, составлением алгоритма и созданием документации.

С этой целью используется унифицированный язык моделирования (UML) - это графический язык, включающий в себя множество различных диаграмм, помогающих специалистам по системному анализу создавать алгоритмы, а программистам - разбираться в принципах работы программы [3].

Словарь UML включает три вида строительных блоков:

* Диаграммы.
* Сущности.
* Связи.

Сущности – это абстракции, которые являются основными элементами модели, связи соединяют их между собой, а диаграммы группируют представляющие интерес наборы сущностей.

Таким образом, UML является мощным инструментов, позволяющим сделать процесс программирования более легким и эффективным.

* + 1. **Диаграмма классов**

Диаграмма классов занимает основное место в проектировании объектно-ориентированной системы. Нотация классов применяется на различных этапах проектирования и строится с разной степенью детализации. Язык UML используется не только для проектирования, однако и с целью документирования проекта. Диаграмма – это графическое представление комплекта элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей). Диаграммы классов демонстрируют набор классов, интерфейсов, а также их связи. Они предназначены для статического представления системы.  
 Ход проектирования приложения естественно предполагает как минимум присутствие схемы классов, которые будут созданы, и представление их взаимодействия. Среда Visual Studio обладает областью проектирования ClassDesigner, в коей возможно сформировать диаграмму классов. Далее в классы можно добавлять поля, свойства и методы, а вдобавок и устанавливать связи между ними.  
 Большинство элементов UML располагают уникальной и прямой графической нотацией, которая дает зрительное представление особенно значительных аспектов элемента. Образец диаграммы классов для нашей системы учета домашних животных будет представлен в следующей главе.

# **Платформа .NET**

Платформа .NET - это совокупность программных средств, обеспечивающих разработку приложений на базе промежуточного кода и их выполнение в специализированной операционной среде . NET Framework. .NET возник как инициативный проект фирмы Microsoft, охватывающий комплекс технологий, программных средств, стандартов и средств разработки, сосредоточенный на обеспечении создания единого информационного пространства в Интернете и соединяющий или согласующий между собой современную вычислительную технику и программное обеспечение [2] Язык C# и платформа . NET впервой были презентованы в 2002 году и предназначались для обеспеченья более мощной, гибкой и простой модели программирования по сравнению с предыдущей моделью программирования COM. Платформа . NET предоставляет следующий список ключевых средств:

1. Возможность взаимодействия с существующим кодом; Эта возможность является очень хороша, поскольку дозволяет сочетать существующие бинарные единицы СОМ (т. обеспечивать их взаимодействие) с более свежими бинарными единицами . NET и наоборот. С выходом версии . NET 4. 0 эта возможность стала выглядеть еще проще, вследствие добавлению ключевого слова dynamic.
2. Поддержка множества языков программирования;

Приложения . NET возможно создавать с через практически любой язык программирования (C#, VisualBasic. S# и т.п Причем в . NET код, написанный на любом языке компилируется в код на промежуточном языке (IntermediateLanguage - IL) Общий исполняющий механизм, делимый всеми поддерживающими . NET языками).

1. Языковая интеграция; В . NET поддерживается межъязыковое наследование, межъязыковая обработка исключений и межъязыковая отладка. При этом . NET применяет общий исполняющий механизм, генеральным аспектом которого является неплохо определенный комплект типов, который способен определять каждый, поддерживающий . NET язык.
2. Обширный набор базисных классов; Данная библиотека дает более тысячи первоначально назначенных типов, кои дозволяют создавать библиотеки кода, несложные оконечные приложения, графические настольные приложения и веб-сайты уровня предприятия.
3. Упрощенная модель развертывания. Так же, в . NET был всецело переделан метод деления кода промеж приложений за счет добавления понятия сборки (assembly - англ.) заместо классических библиотек DLL. Сборки обладают формальными средствами для управления версиями и допускают одновременное существование рядом нескольких различных версий сборок. Набор частей . NET, известный под общим именем ADO. NET, дозволяет обретать лучший доступ к реляционным базам данных и многим другим источникам данных. Вдобавок предлагаются компоненты, позволяющие получать доступ к файловой системе и каталогам. В частности, в . NET встроена поддержка XML, дозволяющая управлять данными, импортируемыми и экспортируемыми на платформы, отличные от Windows.
   * 1. **Структура .NET**

Платформа . NET располагает тремя прикладными направлениями: первое ориентировано на пользователей и разработчиков технических и программных средств, другое - на разработчиков-профессионалов информационных технологий и третье - на бизнесменов.  
Структура платформы . NET складывается из нескольких частей (Рис.1.1)  
1)ОС .  
2)Платформа разработки и исполнения приложений;  
3)Сетевые сервисы и серверные продукты;  
4)Средства разработки приложений.

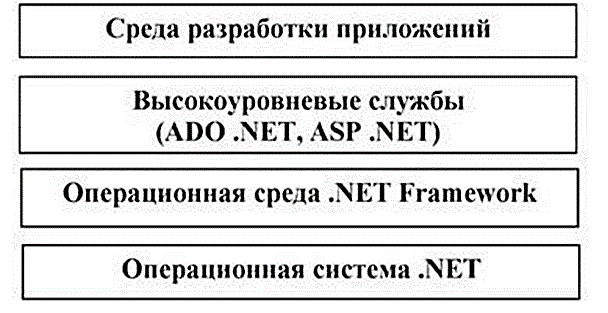


Рис. 1.1 Структура .NET

**1.2.1. Технология .NET**

Первоначально программист избирает программную среду разработки, программу-компилятор коей обеспечивает создание переходного кода на языке MSIL. При помощи инструментов Microsoft разработчик может работать с языками С#, C , Visual Basic.  
Итак, исполнитель выбрал набор инструментов-сред разработки, создал некое приложение и откомпилировал его. В результате компиляции создается код приложения на промежуточном коде MSIL, который не интерпретируется в машинные команды. Следовательно, приложение. NET становится самостоятельным от конкретных реализаций операционной системы и аппаратной платформы.  
Готовое приложение. NET складывается из сборок. Сборка — это один или несколько файлов, в которых помимо собственно кода MSIL приложения вдобавок интегрированы метаданные — разнообразная служебная информация о самом приложении. Как результат-отпадает потребность в регистрации приложения в системном реестре, подобно приложениям СОМ, как-никак вся необходимая информация доступна вместе с приложением. Здесь же, к примеру, возможно добавить сведения о версии приложения и т.  
Готовая программа должна воспроизводиться на компьютере, если на нем поставлена операционная среда. NET Framework. Код приложения взаимодействует только с операционной средой, абстрагируясь от уровня операционной системы. Все упоминаемые в дальнейшем инструменты и механизмы представляются ее составными частями.  
При открытии разработанного на платформе .NET приложения запускается среда выполнения приложения Common Language Runtime (или сокращенно CLR), что при помощи загрузчика загружает сборки приложения и обеспечивает его выполнение. Но для сего нужно перевести код MSIL в машинные команды процессора. (рис. 1.2).

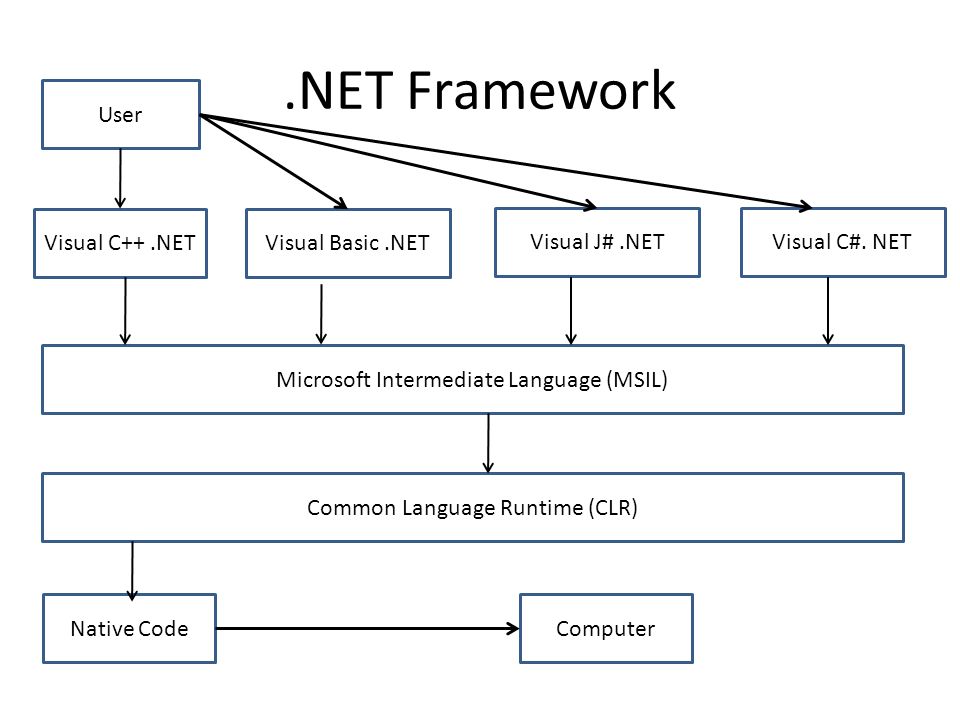


Рис. 1.2 Технология .NET

Абсолютные превосходства подобного подхода — абстракция кода приложения от реализации функций операционной системы и право реализовать "интеллектуальное выполнение" кода, распоряжаясь интерпретируемым кодом. Создатели . NET постарались решить проблему действенности интерпретируемого кода. Для этого в составе . NET Framework присутствует JustInTime (JIT) компилятор, кой проделывает переделку кода MSIL в машинные указания по мере вызова подпрограмм. Как только в коде видится вызов подпрограммы, загрузчик CLR находит и загружает нужный фрагмент кода в компилятор JIT. В результате излишний в данный момент участок кода приложения возможно будет вообще не откомпилирован. Откомпилированный в ходе одного сеанса исполнения приложения код можно использовать многократно, CLR побеспокоится о том, для сохранения его на протяжении сеанса воспроизведения приложения. Помимо этого, CLR может хранить однажды откомпилированный код подпрограмм на жёстком диске и вызывать его при последующих обращениях к приложению. На выходе компилятора образовывается так называемый управляемый код, кой дальше выполняется в среде CLR. Такой код называется управляемым, т. к. он содержит метаданные, дозволяющие CLR, помимо явной функции управлением компиляцией, исполнять и многие другие операции, к коим причисляются контроль безопасности, проверка прав, контроль версий, сборка мусора и так далее.

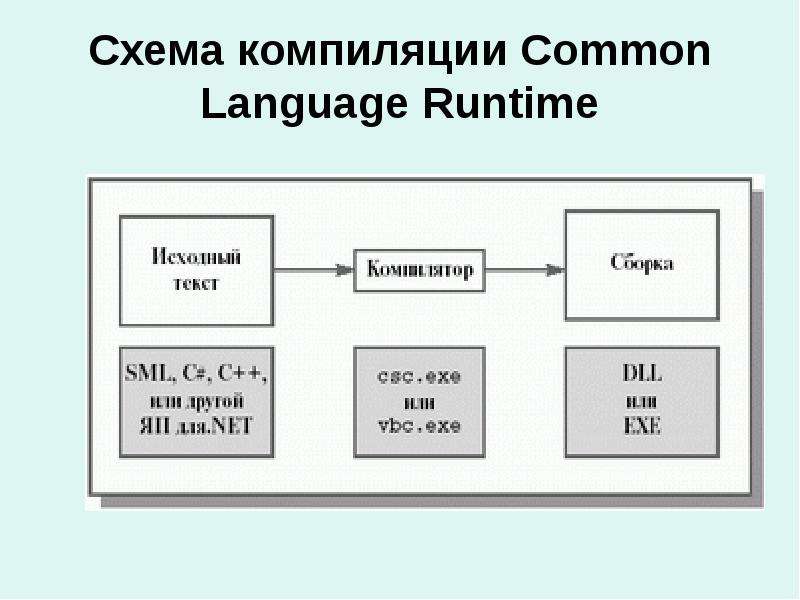
.

Рис. 1.3 Схема компиляции CLR

Таким образом, можно сделать вывод о том, что разработка приложений .NET выполняется в специализированных средах, а выполнение подразумевает использование операционной среды .NET Framework.

# **Связь языка C# и .NET**

C# создавался как язык компонентного программирования, и в этом одно из главных достоинств языка, сориентированное на право вторичного применения разработанных ранее компонентов. Из других справедливых факторов заметим следующие. C# формировался синхронно с каркасом Framework . Net и в полной мере предусматривает весь его потенциал - как FCL, так и CLR. Он является всецело объектно-ориентированным языком, где аж типы, интегрированные в язык, изображены классами. Язык C# и связанную с ним среду . NET Framework впору вне всякого приукрашивания называть самой значительной из предлагаемых в настоящее время технологий для разработчиков. Среда . NET является такой средой, какая была созданадабы в ней возможно было разрабатывать почти любое приложение для запуска в Windows, а C# является языком программирования, кой был специально создан для использования в . NET Framework. Например, с использованием C# и . NET Framework становится возможным создавать динамические веб-страницы, приложения Windows Presentation Foundation, веб-службы XML, составляющие для распределенных приложений, составляющие для доступа к базам данных, классические настольные приложения Windows и даже клиентские приложения нового интеллектуального типа, располагающие возможностями для работы в оперативном и автономном режимах.

# **Создание Windows-приложений**

Для создания графических интерфейсов используя платформы . NET используются различные технологии - WinForms, WPF, приложения для магазина Windows Store. Впрочем наиболее легкой и комфортной платформой до сих пор остается WinForms или формы. Класс Form – по файту это контейнер, в кой вмещаются остальные элементы приложения, определяющие далее всю функциональность и работоспособность созданного приложения [4]. При создании нового приложения появляется пустая форма, представляющая собой окно дизайнера форм. Конструктор дозволяет работать с формой: помещать компоненты в форму, удалять и выделять компоненты, перетаскивать их с одного места на другое, редактировать характеристики формы, обрабатывать ее события и методы.Конструктор дает программисту различные возможности для работы с функциональностью и пользовательским интерфейсом приложения.

1. **РЕАЛИЗАЦИЯ ПРОЕКТА**
   1. **Постановка задачи**

Разработать приложение, позволяющее собирать и накапливать сведения по учёту домашних животных, содержащихся в фермерском хозяйстве. Структура приложения обязательно должна включать следующие классы, созданные студентом: домашнее животное, вид домашнего животного, порода, класс животных, корм, карточка учёта животных, владелец и др.

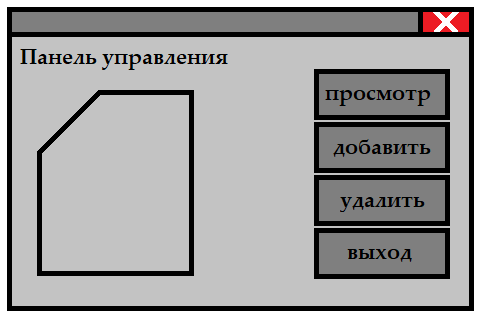
****

Рис. 2.1 Внешний вид панели учета и управления

Система учета хранит определенное ограниченностью памяти количество записей. Панель управления осуществляет доступ к любой записи о животном напрямую из базы данных MySql. У пользователя есть возможность просматривать уже созданные записи, создавать новые или удалять старые. Исполнителем должна быть разработана схема базы данных для хранения данных о учете домашних животных в фермерском хозяйстве. Интерфейс с пользователем осуществляется при помощи экранного меню. При помощи кнопки «Вход» осуществляется вход в аккаунт, для доступа к панели управления. С помощью кнопок «Просмотр», «Добавить» и «Удалить» есть возможность манипуляции записями о животных напрямую через программу. Также есть кнопки «Раскрыть всё» и «Скрыть всё» для удобного просмотра созданных записей. После добавления нового животного появляется соответствующее окно, уведомляющее об успешности операции. Также, подобное окно появляется и при выполнении операции удаления животного из базы данных. Возможен просмотр сохраненных данных с целью их учета благодаря кнопке «Просмотр». При нажатии на нее откроется отдельное окно просмотра. В нем в виде дерева узлов удобно представлены все накопленные сведения. При нажатии на кнопку «Удалить» открывается окно с соответствующим режимом. В нем можно удалять записи о животных, зная их ID-уникальный идентификатор.

# **2.2. Проектный план до реализации**

Проектный план до реализации состоит из функциональных требований к разработке приложения, оценки сложности реализации и статуса их выполнения (рис. 2.2).



Рис. 2.2 Функциональные требования

Далее был разработан проектный план работы, распределены обязанности, назначены сроки выполнения работы и оценены риски, связанные с эксплуатацией приложения (рис. 2.3).

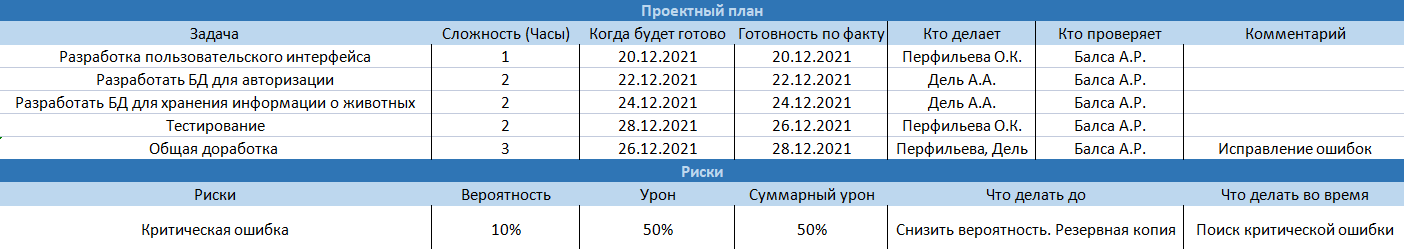


Рис. 2.3 Проектный план

**2.3 Руководство пользователя**

Данное руководство служит пользователю инструкцией по использованию приложения.

При запуске программы появляется главное окно программы (рис. 2.4)

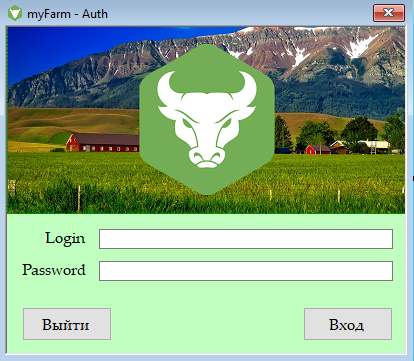


Рис. 2.4 Главное окно программы

Чтобы получить доступ к панели администрирования необходимо ввести логин и пароль, заданный ранее в базе данных. После успешной авторизации пользователю открывается доступ к панели управления (рис. 2.5).

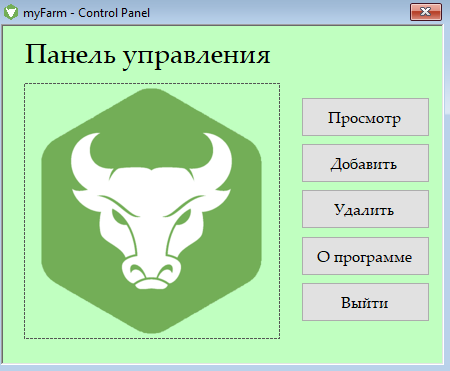


Рис. 2.5 Панель управления

В этом окне в полной мере раскрывается управление записями о животных, содержащихся на ферме.

При нажатии на кнопку «Просмотр» открывается окно (рис. 2.6) просмотра всех записей, хранящихся в базе данных приложения. Стоит отметить, что панель управления при этом не закрывается, что позволяет дополнительно открыть другие органы управления БД.

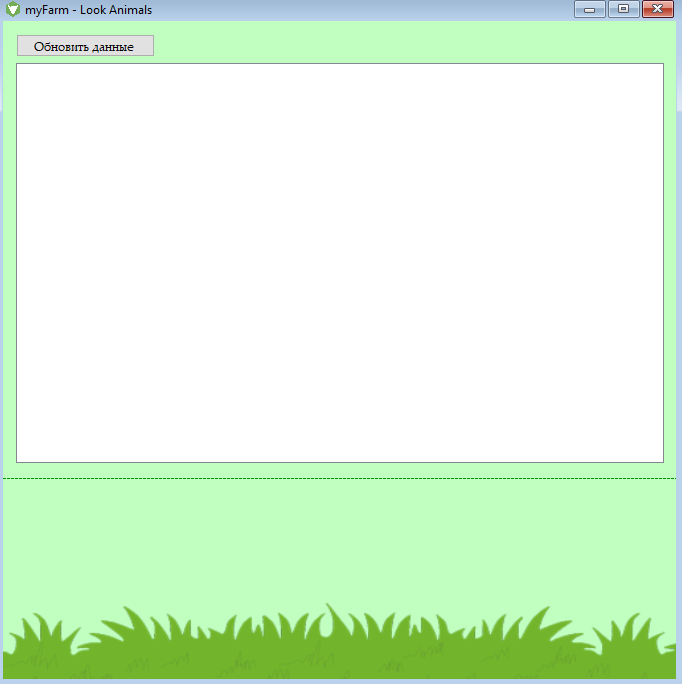


Рис. 2.6 Окно просмотра записей о животных

Используя кнопку «Обновить данные» можно получать актуальные записи, в случае сомнений их актуальности.

С помощью панели управления также есть возможность добавлять новые записи используя кнопку «Добавить». Откроется окно (рис. 2.7) с 7ю полями-характеристиками животного. Заполнять поля необязательно.

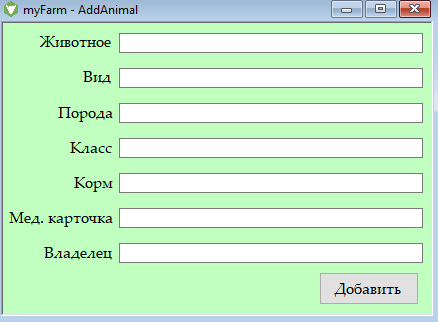


Рис. 2.7 Окно просмотра записей о животных

После нажатия кнопки «Добавить» заполненные поля отправляются в базу данных и впоследствии могут быть обнаружены в окне «Просмотр».

С помощью панели управления также можно удалять накопленные данные о животных. Достаточно перейти с помощью кнопки «Удалить» в специально предназначенное для этого окно (рис. 2.8).

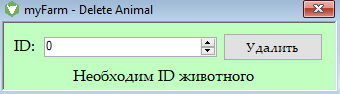


Рис. 2.8 Окно удаления записей о животных

Для удаления записи необходимо знать ее уникальный идентификатор (который далее будем называть ID). Просмотреть ID записи о животном можно с помощью окна «Просмотр» на панели управления.

Что немаловажно – составлено окно «О программе» (рис. 2.9), которое можно открыть всё с того же окна «Панель управления».

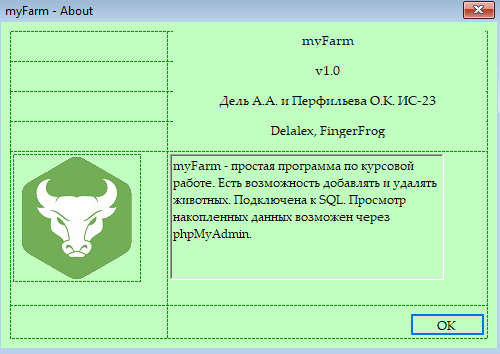


Рис. 2.9 Окно «О программе»

В этом окне можно просмотреть информацию о разработчиках, краткое описание проекта и его версию. Двукратным нажатием на значок приложения можно увидеть технологию, которая была применена в ходе составления программы.

# **2.4 Проектный план после реализации**

В ходе работы были распределены ресурсы для удобства реализации интерфейса приложения (рис. 2.10).

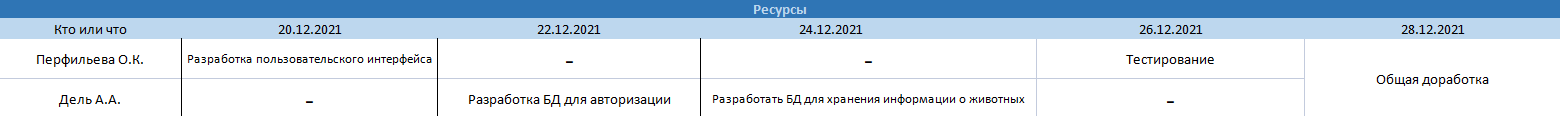


Рис. 2.10 Ресурсы

Разработанное приложение было протестировано в соответствие с заявленными требованиями, были исследованы различные сценарии работы приложения (рис. 2.11).

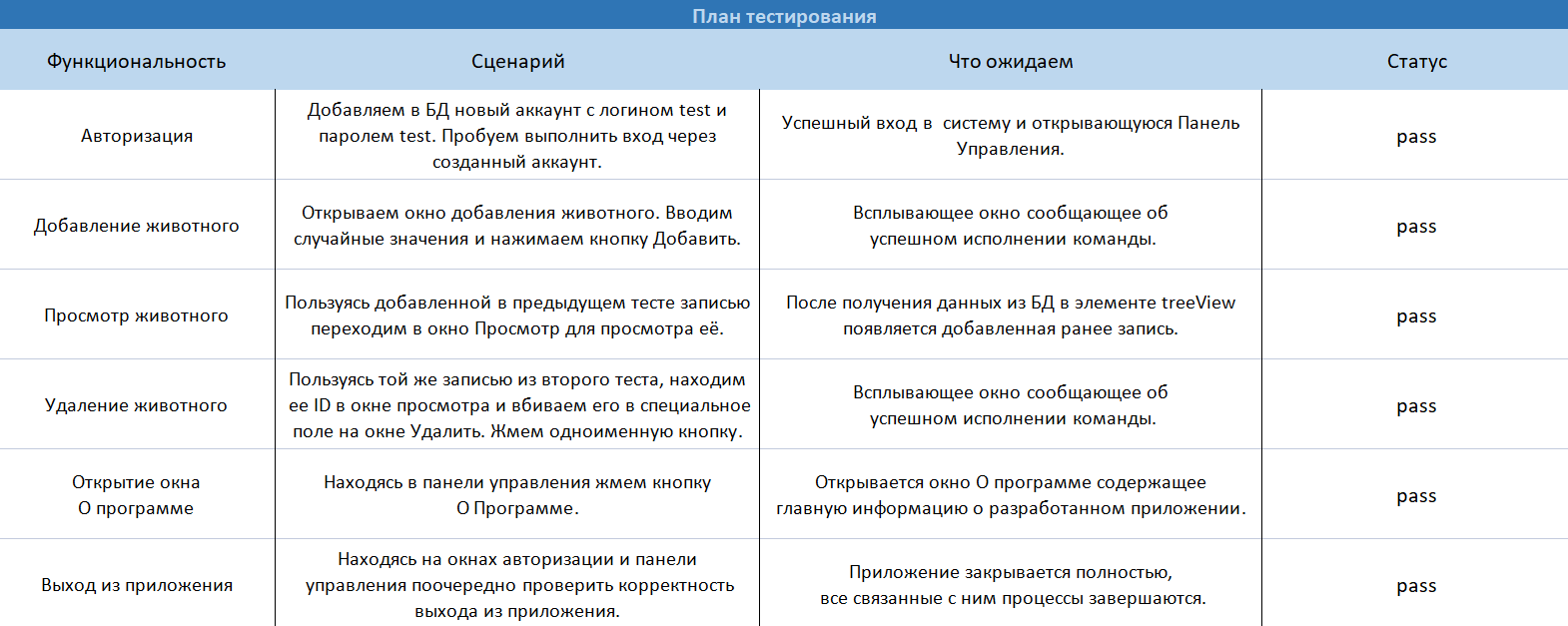


Рис. 2.11 План тестирования

## **ЗАКЛЮЧЕНИЕ**

В результате курсовой работы был разработан интерфейс диктофона и реализована база данных для хранения сообщений в памяти устройства.

Были решены следующие задачи, поставленные для достижения цели:

1. Рассмотрен объектно-ориентированный подход к программированию;
2. Изучена технология .NET /ADO.NET;
3. Разработан проектный план;
4. Построена архитектура приложения «Учет домашних животных в фермерском хозяйстве» с помощью диаграммы классов;
5. Создано Windows-приложение «Учет домашних животных в фермерском хозяйстве» на основе Visual C#;

В дальнейшем рассматривается возможность хранения данных на удалённом сервере, а также добавления дополнительных функций меню в виде «переименования» уже существующих записей о животных.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Евсеева О.Н. Основы языка C# 2005 : учеб. / О.Н. Евсеева, А.Б. Шамшев. – Ульяновск :УлГТУ, 2008. – 132 с.
2. Лафоре Р. Объектно-ориентированное программированиев С++ / Р. Лафоре. – 4-е изд. –СПб.: Питер, 2004 – 922 с.

## Троелсен Э. Язык программирования C# 6.0 и платформа .NET 4.6.

1. Э. Троелсен, Ф. Джеликс ; пер. с англ. Ю.Н. Артеменко. – 7-е изд. – Москва [и др.]: Вильямс, 2017. - 1438 с.  
   Давыдов В.Г. Технология Программирования С++ / В.Г. Давыдов. – СПб.: БХВ-Петербург, 2005. – 672 с.: ил.
2. Пахомов Б.И. С/С++ и MSVisualC++ 2010 для начинающих / Б.И. Пахомов. – СПб.: БХВ-Петербург, 2011. – 736 с.: ил. + дистрибутив (на DVD).

[Электронные ресурсы]

1. UML - диаграммы классов <https://prog-cpp.ru/uml-classes/>
2. Studfiles - <https://studfile.net/preview/14502449/>
3. Основы языка С# 2005 - <http://window.edu.ru/resource/863/58863/files/23.pdf>

# Учебное пособие для студентов специальности 051312 по дисциплине «Высокоуровневые методы инорматики и программирования» - <https://rudocs.exdat.com/docs/index-501640.html?page=3>