Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра комплексной информационной безопасности электронных вычислительных систем (КИБЭВС)

НАКОРМИ ОБЕЗЬЯНКУ (ТАМАГОЧИ)

Курсовая работа по дисциплине «Основы программирования»

Пояснительная записка

Студентка гр. 712-2:
_____Е.Е. Глазырина
«__»_____2024 г.
Руководитель
Преподаватель кафедры КИБЭВС
_____ _____ Б.С. Лодонова
Оценка «__»_____2024 г.

Томск 2024

РЕФЕРАТ

Курсовая работа содержит 55 страниц пояснительной записки, 28 рисунков, 10 источников, 4 приложения.

ЯЗЫК ПРОГРАММИРОВАНИЯ C#, WINDOWS FORMS, VISUAL STUDIO, SQL SERVER MANAGEMENT STUDIO, БАЗЫ ДАННЫХ, ПРОЕКТИРОВАНИЕ.

Программа: «Накорми обезьянку. Тамагочи».

Цель работы: необходимо предоставить пользовательский интерфейс для авторизации и регистрации пользователя, спроектировать и разработать программу «Накорми обезьянку. Тамагочи», реализующую взаимодействие пользователя с виртуальным питомцем.

Разработка программы проводилась на языке программирования C#.

Курсовая работа выполнена в текстовом редакторе Microsoft Word 2013.

Пояснительная записка оформлена согласно ОС ТУСУР 01-2021 [1].

the abstract

The course work contains 55 pages of explanatory notes, 28 drawings, 10 sources, 4 appendices.

C# PROGRAMMING LANGUAGE, WINDOWS FORMS, VISUAL STUDIO, SQL SERVER MANAGEMENT STUDIO, DATABASES, DESIGN.

Program: "Feed the monkey. Tamagotchi."

The purpose of the work: it is necessary to provide a user interface for user authorization and registration, design and develop a program "Feed the monkey. Tamagotchi", which implements the user's interaction with a virtual pet.

The program was developed in the C# programming language.

The course work was done in the Microsoft Word 2013 text editor.

The explanatory note is issued in accordance with OS TUSUR 01-2021 [1].

Министерство науки и высшего образования РФ

ФГБОУ ВО «Томский государственный университет систем управления и радиоэлектроники»

Кафедра комплексной информационной безопасности электронных вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ

Зав.кафедрой КИБЭВС

_____А.А. Шелупанов

«___»_____2023 г.

Задание

на курсовую работу по дисциплине «Основы программирования» студентке группы 712-2 факультета безопасности Е.Е. Глазыриной.

Тема работы: «Накорми обезьянку (тамагочи)».

Цель работы: Получение навыков программирования на языке высокого уровня и применения подхода объектно-ориентированного программирования на практике.

Срок сдачи студентом законченной работы: «___» февраля 2024 г.

Исходные данные к работе: тамагочи - это виртуальное электронное устройство, представляющее собой мобильную игрушку, разработанную в Японии. Она позволяет пользователям заботиться о виртуальном существе, известном как «тамагочи» (от японского слова "тамаго", что означает "яйцо", и "учи", что означает "помещение"). Пользователь отвечает за удовлетворение физических и эмоциональных потребностей своего тамагочи, включая кормление, чистку, игру и межличностные взаимодействия. Тамагочи обладает базовыми элементами искусственного интеллекта и имеет различные состояния, которые зависят от того, как пользователь ухаживает за ним.

Задание: разработать программу, позволяющую пользователю с помощью клавиатуры и мыши взаимодействовать с виртуальным питомцем на примере элементарных команд: кормить, спать, играть.

Требования к используемым технологиям: язык программирования C#, .Net Framework, СУБД SQL Server Management Studio 19.

СОГЛАСОВАНО	ПРИНЯЛ К ИСПОЛНЕНИЮ
Преподаватель кафедры КИБЭВС	Студент гр. 712-2
_______Б.С. Лодонова	_____Е.Е. Глазырина
«___»_____ 2023 г.	«___» _____ 2023 г

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

Накорми обезьянку

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

На 11 листах

СОГЛАСОВАНО	РАЗРАБОТЧИК
Преподаватель кафедры КИБЭВС	Студент гр. 712-2
_______Б.С. Лодонова	_____Е.Е. Глазырина
«___»_____ 2023 г.	«___» _____ 2023 г

Томск 2023

1 Общие сведения

1.1 Полное наименование системы и её условное обозначение

Полное наименование системы: “Накорми обезьянку (тамагочи)”.

1.2 Заказчик

Заказчиком является Томский государственный университет систем управления и радиоэлектроники, кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС).

1.3 Исполнитель

Исполнителем является студентка группы 712-2 Глазырина Елизавета Евгеньевна.

1.4 Основания для разработки

Основанием для разработки является задание на выполнение курсовой работы по дисциплине «Основы программирования» для студентов направления 10.03.01 «Информационная безопасность».

2 Назначение и цель создания системы

2.1 Назначение системы

Система предназначена для реализации игры под названием «Накорми обезьянку».

2.2 Цели создания системы

Целью разработки является создание пользовательского приложения для игры «Накорми обезьянку».

3 Характеристика объектов автоматизации

3.1 Объект автоматизации

Объектом автоматизации является процесс последовательных действий необходимых для игры «Накорми обезьянку».

4 Требования к системе

4.1 Перечень подсистемы, их назначение и основные характеристики

В системе предлагается выделить следующие функциональные подсистемы:

1) Подсистема графического интерфейса, для более комфортного использования;

2) Подсистема изменения состояния питомца по истечению установленного времени;

3) Подсистема проверки состояния питомца (величины жизненных показателей);

4) Подсистема регистрации;

5) Подсистема авторизации.

4.2 Требования к надежности

При возникновении сбоев в аппаратном обеспечении, включая разряд аккумулятора устройства, информационная система восстанавливает свою работоспособность после устранения сбоев и корректного перезапуска аппаратного обеспечения (за исключением случаев повреждения рабочих носителей информации с исполняемым программным кодом).

4.3 Требования к безопасности

Все технические решения, использованные при создании системы, а также при определении требований к аппаратному обеспечению, соответствует действующим нормам и правилам техники безопасности, пожарной безопасности, а также охраны окружающей среды, при эксплуатации или утилизации.

4.4 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению

Для эксплуатации разрабатываемой информационной системы необходимы следующие условия:

1) Компьютер под управлением операционной системы Windows 7 и более, и MacOS;

2) Предустановленный .Net Framework 4.7.2

3) Питание компьютера от сети или батареи;

4) Наличие периферийного устройства – мышь.

4.5 Требования к защите информации от несанкционированного доступа

Доступ к работе с интерфейсом системы имеют только авторизованные пользователи.

4.6 Требования к функциям разработчика

Роль разработчика заключается в обновлении и пополнении системы новыми функциями, а также исправление возможных ошибок в функционировании системы.

4.7 Требования к функциям пользователя

Пользователь может использовать все функции, которыми обладает система.

4.8 Описание процессов и функций работы с системой

Процессы и функции, выполняемые при эксплуатации системы, приведены в разбивке по подсистемам: подсистема графического интерфейса, подсистема изменения состояния питомца по истечению установленного времени, подсистема проверки состояния питомца, подсистема авторизации, подсистема регистрации. Процессы, реализованные под управлением различных подсистем, реализуются на основе системных процедур, которые являются составной частью функции системы. Системные процедуры группируются в соответствии с их назначением:

1) Графический интерфейс пользователя;
2) Проверка столбцов/строк;
3) Изменение состояния питомца;
4) Проверка состояния питомца;
5) Регистрация пользователя;
6) Авторизация пользователя.

4.9 Требования к информационному обеспечению системы

Компоненты системы должны активно взаимодействовать с системой управления базой данных (СУБД). Обмен информацией с СУБД должен происходить автоматически. Уровень хранения данных в системе должен быть построен на основе реляционных или объектно-реляционных СУБД. Доступ к данным должен быть предоставлен только авторизованным пользователям.

4.10 Требования к программному обеспечению

1) ОС Windows 7, 8, 10 и 11 или MacOS;
2) Язык программирования C#;
3) .Net Framework 4.7.2;
4) Установлено ПО.

5) СУБД MySQL.

5 Порядок контроля и приемки системы

5.1 Перечень этапов испытаний и проверок

Этапы испытаний подразделяются на предварительные и приемочные. Предварительные испытания проводятся на стадии тестирования разработчиком. Приемочные испытания проводятся во время сдачи проекта разработчиком совместно с заказчиком. Все подсистемы испытываются одновременно на корректность взаимодействия подсистем, влияние подсистем друг на друга, то есть испытания проводятся комплексно. Во время приемочных испытаний оценивается:

1) Полнота и качество реализации функций, указанных в настоящем техническом задании;

2) Демонстрация объектно-ориентированного подхода при реализации функций, указанных в настоящем техническом задании;

3) Выполнение каждого требования, относящегося к интерфейсу системы;

4) Полнота действий доступных пользователю:

- Ввод данных пользователя;
- Возможность начать и остановить игру;
- Возможность увеличивать жизненные показатели питомца;
- Сохранение результата;
- Выход авторизованного пользователя из приложения.

Приемка результатов должна осуществляться в сроки, установленные заказчиком. Результаты проектирования системы и её тестирования предоставляются в электронном виде с помощью ЭИОС sdo.tusur.ru.

5.2 Общие требования к приемке работы

Приемка осуществляется представителями Заказчика и Исполнителя. Все создаваемые в рамках настоящей работы программные изделия передаются Заказчику, как в виде готовых модулей, так и в виде исходных кодов, представляемых в сохраненных программах C#.

6 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

Для обеспечения готовности объекта к вводу системы в действие провести комплекс мероприятий:

1) Загрузка файлов приложения;

2) Проведение предварительных испытаний;

3) Проверка приемочных испытаний.

7 Требования к документированию

Состав программной документации:

1) Задание на курсовую работу
2) Техническое задание (ТЗ);
3) Пояснительная записка (ПЗ);
4) Документация к системе в электронном виде;

Документация должна быть оформлена с использованием:

1) ГОСТ 34.602-89;
2) ОС ТУСУР 01-2021 для технического задания;
3) ОС ТУСУР 01-2021 для пояснительной записки

Содержание

Введение

Целью данной курсовой работы является получение навыков разработки алгоритмов и их реализации на языке программирования высокого уровня в виде программного обеспечения для персонального компьютера.

1 ОБЗОР ТЕМЫ

Изначально игра тамагочи шла для портативной консоли «Game Boy» от «Nintendo». Картридж был оборудован EEPROM-памятью, процессором не шибко мощным, динамиком и часами. Фактически виртуальный питомец жил сам по себе в это время внутри картриджа. Пока хозяин не подключал его к консоли на протяжении какого-то времени, питомец мог успеть испражниться, заболеть или вовсе умереть, о чём сигнализировал встроенный динамик. Но, чтобы оказать необходимый уход, нужно иметь при себе «Game Boy», который был не у всех, а питомца хотелось многим. Потому появление массовой версии был вопросом времени [3].

Продавать игру начали с 1996 года. Ее создателем считается Маита Аки. Идея изобретательнице пришла после того, как она просмотрела видеоролик о том, как мальчик пытается пронести питомца в школу. После чего и возникла мысль о «зверьке в кармане». Когда девайс вышел на мировые рынки, продажи выросли до 40 миллионов копий.

Собственно, оригинальный «Тамагочи» был оснащён тремя кнопками: выбор одного из восьми пунктов в меню, подтверждение и отмена. И, в отличие от дешёвых китайских аналогов, владелец игрушки не получал никакого «зоопарка» в своё распоряжение — лишь одного абстрактно выглядящего питомца. Однако, по аналогии с возникшими в тот же год «Покемонами», пришелец мог эволюционировать в одно из нескольких существ. Какое — зависело от действий игрока. Скажем, если сильно перехваливать питомца, тот может вырасти избалованным, а если не проявлять должного ухода — он станет обиженным на жизнь существом, а то и вовсе погибнет. Идеальный сценарий предполагал минимум сладостей в рационе, своевременное оказание внимания (корм, лечение от болячек) и, конечно же, уборку продуктов жизнедеятельности подопечного. Даже свет, когда питомец засыпал, следовало выключать — в этом случае пришелец

вырастал здоровым, радостным и проживал максимально положенный ему срок.

Хотя с момента появления «Тамагочи» на свет прошло уже более 20 лет, ключевые механики этой игрушки по-прежнему находят своё отражение как в мини-играх, так и отдельных проектах, многие из которых добились огромного успеха — как, например, та же популярная игра My Talking Tom [4].

Тамагочи - это не только игрушка, но и символ эпохи, когда сначала начиналось виртуальное воспитание и забота о виртуальном существе. Он показал, что путешествие "воспитания" не требует реального питомца, и может быть совершено с помощью цифровых устройств. Тамагочи остается одним из самых знаменитых игрушек, которая оставила неизгладимый след в истории игровой индустрии и культуре.

2 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОГО ПРИЛОЖЕНИЯ

2.1 Обоснование выбранных технологий

Перед тем, как начать реализовывать выбранную игру, необходимо определиться с набором технологий, с помощью которых будет написано приложение. Необходимо определиться со следующим набором: язык программирования, среда разработки, СУБД.

Игра «Накорми обезьянку (тамагочи)» будет выполнена на объектно-ориентированном языке программирования C# в интегрированной среде разработки Visual Studio.

Объектно-ориентированных языков программирования на сегодняшний момент достаточно много. Но остановимся на двух языках программирования C# и Python. Python – универсальный язык и используется во многих сферах. Python пригодится в создании компьютерных и мобильных приложений, его применяют в работе с большим объемом информации, при разработке web-сайтов и других разнообразных проектов, используют в машинном обучении [5]. Язык C# чаще используют для создания игр и веб-приложений, к тому же, данный язык программирования уже изучался во втором семестре первого курса на дисциплине «Основы программирования».

В качестве среды разработки была выбрана программа «Visual Studio». Visual Studio предоставляет разработчикам широкие возможности среды разработки для эффективного и совместного разработки высококачественного кода [6]. Кроме того, для сравнения была взята достаточно новая среда разработки Project Rider. Project Rider – это среда от JetBrains для работы с платформой .NET. Выпущена в прошлом году, но уже приобрела много поклонников. Но у данной IDE все же есть небольшие минусы, один из них – молодость (часть функциональности программы еще

находится в разработке, поэтому не все стартовые баги в ней исправлены) [7].

При выборе базы данных внимание больше всего привлекла СУБД (система управления базами данных) SQL Server Management Studio (далее – SSMS). Среда SSMS предоставляет собой единую служебную программу, которая сочетает в себе обширную группу графических инструментов с разными многофункциональными редакторами скриптов для доступа к SQL Server для разработчиков и администраторов баз данных всех уровней [8].

2.2 Описание алгоритмов приложения

Алгоритм А «Основное окно игры»:

A0. Начало.

A1. Создать класс form2 в пространстве имен "курсовая".

A2. Установить форму form2 как частичную форму (public partial class form2 : form).

A3. Создать публичное поле sleep типа int и инициализировать его значением 0.

A4. Создать только для чтения поле imagelist1 типа object и присвоить ему пустое значение.

A5. Создать конструктор form2 без параметров.

A6. Внутри конструктора вызвать метод initializecomponent().

A7. Присвоить game.image значение imagelist1.images[3].

A8. Если нажата кнопка button4_click – переход к шагу A9, иначе – переход к шагу A12.

A9. Присвоить game.image значение imagelist1.images[3].

A10. Присвоить переменной sleep значение 0.

A11. Включить кнопки button1.enabled и button2.enabled.

A12. Если нажата кнопка button1_click – переход к шагу A13, иначе – переход к шагу A16.

A13. Если значение progressbar3.value меньше 100, то переход к шагу A14, иначе – переход к шагу A15.

A14. Увеличить значение progressbar3.value на 100.

A15. Присвоить game.image значение imagelist1.images[2].

A16. Если нажата кнопка button2_click, переход к шагу A17, иначе – переход к шагу A19.

A17. Проверить, если значение progressbar2.value больше значения progressbar2.maximum минус 10, то отключить свойство button2.enabled, иначе – переход к шагу A18.

A18. Проверить, если значение progressbar2.value меньше значения progressbar2.maximum минус 10, то увеличить значение progressbar2.value на 10 и присвоить game.image значение imagelist1.images[1], иначе – переход к следующему шагу.

A19. Если нажата кнопка button3_click – переход к шагу A20, иначе переход к шагу A23.

A20. Присвоить game.image значение imagelist1.images[4].

A21. Присвоить переменной sleep значение 1.

A22. Отключить свойства button1.enabled и button2.enabled.

A23. Создать метод timer1_tick без параметров.

A24. Проверить, если значение progressbar2.value меньше значения progressbar2.maximum минус 10 и sleep равно 0, то включить свойство button2.enabled, иначе – переход к шагу A25.

A25. Проверить, если значение progressbar2.value равно 1, то присвоить game.image значение imagelist1.images[5], переход к шагу A26, иначе – переход к шагу A27.

A26. Отключить свойства happy.enabled, button1.enabled, button2.enabled, button3.enabled, button4.enabled и soon.enabled, переход к шагу A34.

A27. Уменьшить значение progressbar2.value на 1.

A28. Создать метод timer2_tick без параметров.

A29. Внутри метода timer2_tick проверить, если sleep равно 1, то проверить, если значение progressbar1.value не равно 100, то увеличить значение progressbar1.value на 1, иначе – переход к шагу A30

A30. Присвоить game.image значение imagelist1.images[3] и уменьшить значение progressbar1.value на 1.

A31. Проверить, если значение progressbar3.value равно 1, то присвоить game.image значение imagelist1.images[5] и отключить свойства happy.enabled, button1.enabled, button2.enabled, button3.enabled, button4.enabled и soon.enabled и переход к шагу A34, иначе – переход к шагу A33.

A32. Проверить, если значение progressbar1.value равно 1, то присвоить game.image значение imagelist1.images[5] и отключить свойства happy.enabled, button1.enabled, button2.enabled, button3.enabled, button4.enabled и soon.enabled и переход к шагу A34, иначе – переход к шагу A33.

A33. Уменьшить значение progressbar3.value на 1. Переход к шагу A8.

A34. Конец.

Блок-схема к алгоритму А представлена в приложении А.

Алгоритм В «Регистрация нового пользователя»:

B0. Начало.

B1. Создать новый экземпляр класса "database".

B2. Создать новый экземпляр формы "form3".

B3. Инициализировать компоненты формы "form3".

B4. Если нажата кнопка "button1", переход к шагу B5, иначе – переход к шагу B17.

B5. Проверить пользователя с помощью функции "checkuser()". Проверка пройдена – к шагу B6, не пройдена – к B17.

B6. Взять логин и пароль из текстовых полей "textbox1" и "textbox2".

B7. Сформировать строку запроса для добавления нового пользователя в базу данных.

B8. Создать объект "sqlcommand" с заданными запросом и подключением к базе данных.

B9. Открыть соединение с базой данных.

B10. Если выполнение команды добавления пользователя успешно (возвращает 1) и переход к шагу B11, иначе – переход к шагу B15.

B11. Показать сообщение об успешном создании аккаунта.

B12. Создать новый экземпляр формы "авторизация".

B13. Скрыть текущую форму "form3".

B14. Показать форму "авторизация".

B15. Показать сообщение о неудачном создании аккаунта.

B16. Закрыть соединение с базой данных.

B17. Конец.

Блок-схема к алгоритму B представлена в приложении Б.

Алгоритм C к функции "checkuser()":

C0. Начало.

C1. Взять логин и пароль из текстовых полей "textbox1" и "textbox2".

C2. Создать объект "sqldataadapter".

C3. Создать объект "datatable".

C4. Сформировать строку запроса для выборки пользователей с заданным логином и паролем.

C5. Создать объект "sqlcommand" с заданным запросом и подключением к базе данных.

C6. Назначить команду выборки объекту "sqldataadapter".

C7. Заполнить объект "datatable" данными из адаптера.

C8. Если в таблице есть хотя бы одна строка – переход к шагу C9, иначе – переход к шагу C11.

C9. Показать сообщение о существующем пользователе.

C10. Вернуть значение true. Переход к шагу C12.

C11. Вернуть значение false.

C12. Конец.

Блок-схема к алгоритму C представлена в приложении B.

Алгоритм D «Авторизация пользователя»:

D0. Начало.

D1. Создаем класс с именем "авторизация" в namespace "Курсовая".

D2. Внутри класса объявляем переменную "database" типа "database" и инициализируем ее новым объектом "database()".

D3. Создаем пустой конструктор "авторизация", внутри которого вызываем метод "initializecomponent()".

D4. Если нажата кнопка "button1_click" с параметрами "sender" и "eventargs" – переход к шагу D5, иначе – переход к шагу D23.

D5. Получаем значения из текстовых полей "textbox1" и "textbox2" и сохраняем в переменные "loginuser" и "passuser" соответственно.

D6. Создаем объекты "adapter" типа "sqldataadapter" и "table" типа "datatable".

D7. Формируем запрос к базе данных, выбирающий запись с заданным логином и паролем: string querystring = $"select id, login, password from пользователи where login = '{loginuser}' " + $"and password = '{passuser}'";

D8. Создаем объект "command" типа "sqlcommand" с заданным запросом и подключением к базе данных.

D9. Присваиваем свойству "selectcommand" объекта "adapter" значение объекта "command".

D10. Заполняем таблицу "table" данными, используя метод "fill" объекта "adapter".

D11. Если в таблице есть только одна запись, то переход к шагу D12, иначе – переход к шагу D16

D12. Выводим сообщение об успешной авторизации с помощью метода "show" класса "messagebox".

D13. Создаем объект "form2" типа "form2".

D14. Открываем форму "form2" с помощью метода "showdialog".

D15. Скрываем текущую форму, вызывая метод "hide".

D16. Выводим сообщение о неверном логине или пароле с помощью метода "show" класса "messagebox", переход к шагу D23.

D17. Объявляем метод "авторизация_load" с параметрами "sender" и "eventargs".

D18. Внутри метода "авторизация_load" устанавливаем максимальную длину текстовых полей "textbox1" и "textbox2" равной 50.

D19. Объявляем метод "linklabel2_linkclicked" с параметрами "sender" и "linklabellinkclickedeventargs".

D20. Внутри метода "linklabel2_linkclicked" создаем объект "form3" типа "form3".

D21. Открываем форму "form3" с помощью метода "show".

D22. Скрываем текущую форму, вызывая метод "hide".

D23. Конец.

Блок-схема к алгоритму D представлена в приложении Д.

2.3 Определение временной сложности

Алгоритмическая сложность (Вычислительная сложность) – понятие, обозначающее функцию зависимости объема работы алгоритма от размера обрабатываемых данных [9].

Алгоритмическая сложность игры "Тамагочи" зависит от различных факторов, таких как: количество, тип действий, способ хранения данных и их обработки.

Если упростить игру и рассмотреть только основные действия, такие как питание, игра с питомцем и сон, то алгоритмическая сложность может быть оценена как O(1), то есть постоянная сложность. В этом случае для каждого действия или команды будет небольшое число операций, не зависящее от размера игрового мира или разнообразия действий.

2.4 Проектирование структуры программы приложения

Перед написанием программного кода, было составлена UML-диаграмма (диаграмма классов). С помощью UML-диаграммы можно отразить визуальное представление всех классов и их взаимосвязь друг с другом. Ниже представлена диаграмма классов (рисунок 2.4.1) для написания программы к игре «Тамагочи. Накорми обезьянку».

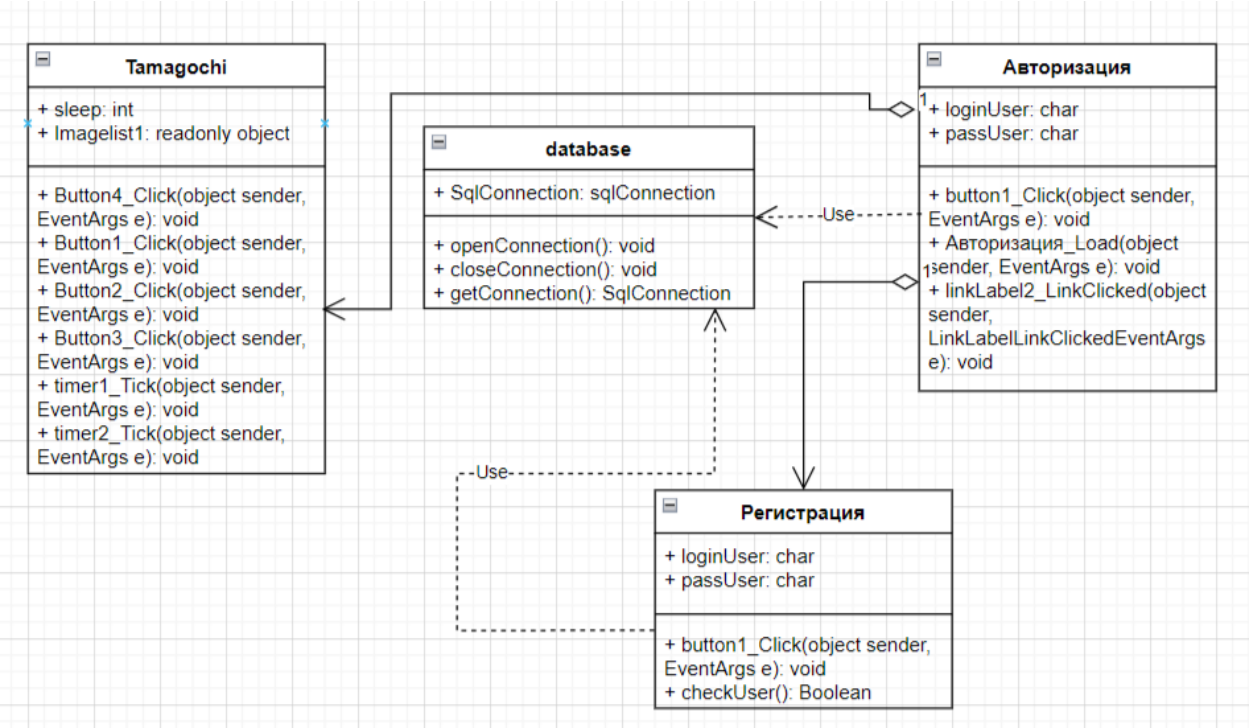


Рисунок 2.4.1 – UML-диаграмма

3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

Для выполнения поставленной задачи, отраженной в техническом задании, были созданы следующие формы:

— Форма «Авторизация» – оконная Windows-форма, которая отражает и обеспечивает интерфейс пользователя с программой. Данная форма предназначена для авторизации уже существующего пользователя или же для перехода к регистрации ещё не зарегистрированного пользователя.

— Форма «Регистрация» – оконная Windows-форма, которая обеспечивает интерфейс пользователя с программой. Форма предназначена для регистрации нового пользователя.

— Форма основного окна «Накорми обезьянку» – оконная Windows-форма, предоставляющая интерфейс взаимодействия пользователя с программой. Предназначена для взаимодействия пользователя с виртуальным питомцем с помощью мыши.

Кроме того, был разработан класс «База данных» – класс, обеспечивающий инициализацию базы данных, создающий таблицы для хранения данных обо всех пользователях приложения, имеющий методы проверки уже существующего пользователя и исполняющий функцию регистрации нового пользователя.

Полный код программы представлен в репозитории.

Рассмотрим основной функционал программы.

При запуске программы открывается форма «Авторизация» (рисунок 3.1).

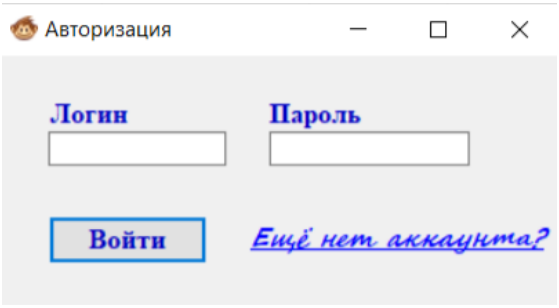


Рисунок 3.1 – Форма «Авторизация»

Заметим, что на форме присутствуют два текстовых поля для ввода логина и пароля. Далее в левом нижнем углу расположена кнопка «Войти», при нажатии которой, будет выполнен следующий фрагмент кода программы:

```
private void button1_Click(object sender, EventArgs e)
        {
            var loginUser = textBox1.Text;
            var passUser = textBox2.Text;
            SqlDataAdapter adapter = new SqlDataAdapter();
            DataTable table = new DataTable();
            string querystring = $"select ID, Login, Password from
Пользователи where Login = '{loginUser}' " +
                $"and Password = '{passUser}'";
            SqlCommand command = new SqlCommand(querystring,
database.getConnection());
            adapter.SelectCommand = command;
            adapter.Fill(table);
            if (table.Rows.Count == 1)
            {
                MessageBox.Show("Вы успешно вошли!", "Успешно!",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                Form2 form2 = new Form2();
                form2.ShowDialog();
                this.Show();this.Hide();
            }
            else
            {
                MessageBox.Show("Такого аккаунта не существует",
"Аккаунта не существует!", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
```

```
        }
    }
```

Данная обработка кода обращается к базе данных, и, в случае нахождения такого пользователя логина и пароля в таблице, выдает сообщение об успешной авторизации (рисунок 3.2).

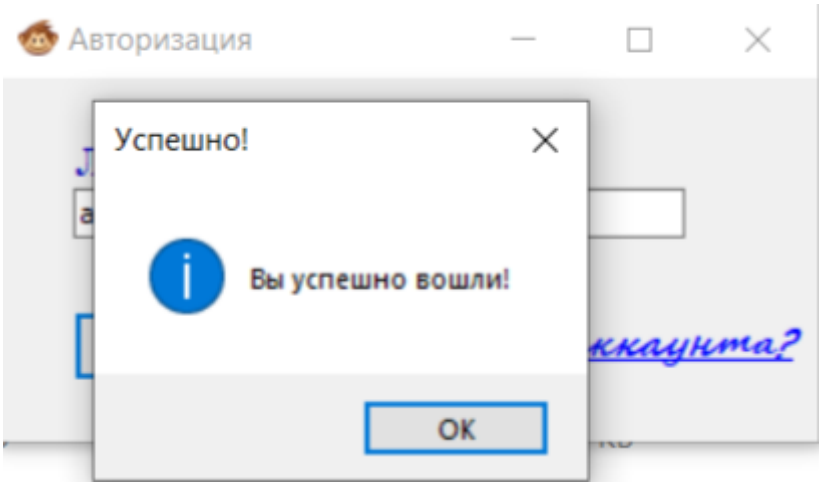


Рисунок 3.2 – Сообщение об успешной авторизации

После нажатия на кнопку «ОК» форма «Авторизация» закроется и откроется форма основного окна «Накорми обезьянку» (рисунок 3.3).

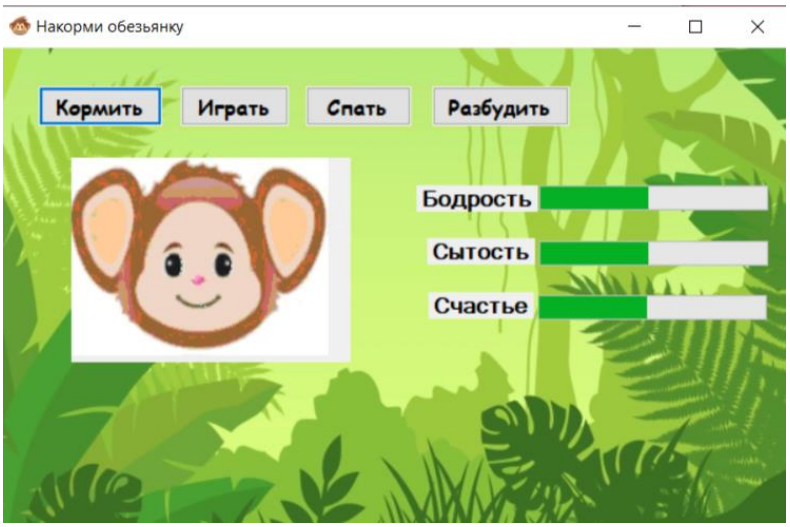


Рисунок 3.3 – Основное окно «Накорми обезьянку»

Если же пользователь еще не зарегистрирован, в форме «Авторизация» (рисунок 3.1) существует кликабельный текст «Еще нет аккаунта?», при нажатии на который будет отработан следующий фрагмент кода:

```
private void linkLabel2_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
    {
       Form3 form3 = new Form3();
       form3.Show();
       this.Hide();
    }
```

Данный код позволяет открыть форму «Регистрация» (рисунок 3.4).

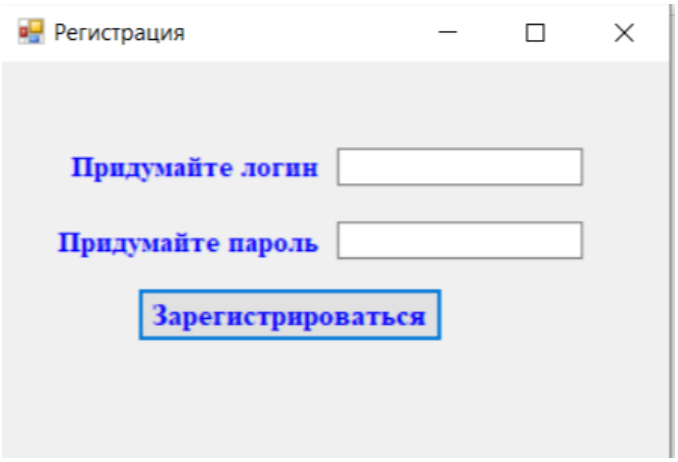


Рисунок 3.4 – Форма «Регистрация»

Высветившаяся форма «Регистрации» предлагает придумать логин и пароль и ввести их в текстовые поля соответственно. Ниже располагается кнопка «Зарегистрироваться», при нажатии на которую будет отработан следующий ряд событий:

```
private void button1_Click(object sender, EventArgs e)
    {
       if (checkUser())
       {
          return;
       }
       var login = textBox1.Text;
       var password = textBox2.Text;

       string querystring = $"insert into Пользователи(Login, Password)
values('{login}', '{password}')";
```

```
        SqlCommand command = new SqlCommand(querystring,
database.getConnection());

        database.openConnection();


        if (command.ExecuteNonQuery() == 1)
        {
           MessageBox.Show("Аккаунт успешно создан!", "Успех!
Спасибо за регистрацию!");
           Авторизация form1 = new Авторизация();
           this.Hide();
           form1.Show();
        }
        else
        {
           MessageBox.Show("Аккаунт не создан!");
        }
        database.closeConnection();

            }
```

После успешной регистрации появляется форма «Авторизация» и теперь новый пользователь может ввести свои логин и пароль, и форма пропустит его к игре.

Вернемся к основному окну (рисунок 3.3). Здесь появляется сама обезьянка в удовлетворительном состоянии, три строки состояния «Бодрость», «Сытость» и «Счастье». По умолчанию их начальные значения будут стоять на 50% у каждой, но они будут постепенно убывать, поскольку установлен таймер. Выше располагаются четыре кнопки для управления состоянием обезьянки. При нажатии на кнопку «Кормить» будет отрабатываться следующее событие:

```
public void Button1_Click(object sender, EventArgs e)
    {
       while (ProgressBar3.Value < 100)
       {
          ProgressBar3.Value =+100;
```

```
        }
        Game.Image = imageList1.Images[2];
    }
```

После нажатия на кнопку «Кормить» показатель «Сытость» вырос до 100% и ненадолго появляется счастливая обезьянка (рисунок 3.5).

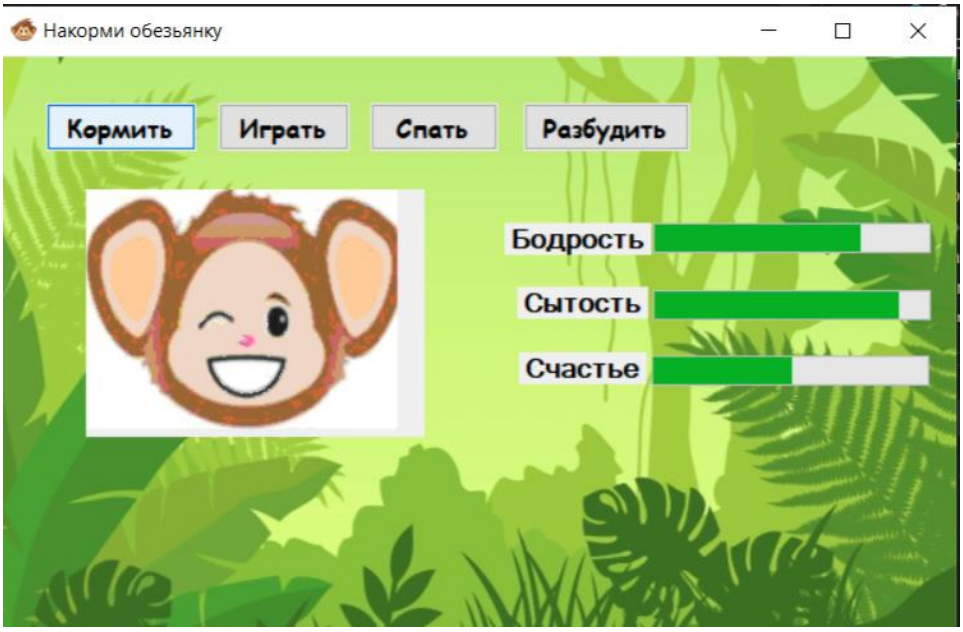


Рисунок 3.5 – Ситуация после нажатия на кнопку «Кормить»

Если нажать на кнопку «Играть» будет обработан следующий фрагмент кода:

```
public void Button2_Click(object sender, EventArgs e)
    {
        if (ProgressBar2.Value > ProgressBar2.Maximum - 10)
        {
            Button2.Enabled = false;
        }
        if (ProgressBar2.Value < ProgressBar2.Maximum - 10)
        {
            ProgressBar2.Value += 10;
            Game.Image = imageList1.Images[1];
        }
```

}

После отработки кода основное окно будет выглядеть следующим образом (рисунок 3.6). Заметим, что эмоция обезьянки сменила, а шкала «Счастье» немного повысилась.

Рисунок 3.6 – Окно игры после нажатия на кнопку «Играть»

Следующая кнопка «Спать». Когда пользователь нажмет на нее, программа приведет в действие фрагмент кода:

```
public void Button3_Click(object sender, EventArgs e)
    {
       Game.Image = imageList1.Images[4];
       sleep = 1;
       Button1.Enabled = false;
       Button2.Enabled = false;
    }
```

После отработки кода кнопки «Кормить» и «Играть» будут заблокированы и вернутся в доступ только, когда пользователь нажмет на четвертую кнопку «Разбудить». Также картинка поменяется, обезьянка уснет, а строка состояния, отвечающая за «Бодрость», начнет медленно расти (рисунок 3.7).

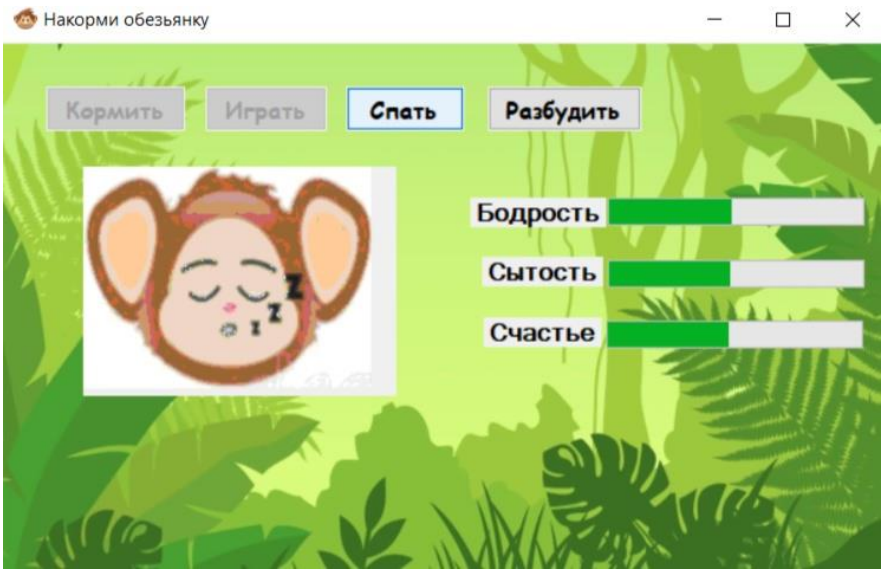


Рисунок 3.7 – Окно игры после нажатия на кнопку «Спать»

Когда будет нажата кнопка «Разбудить», будет отработан следующий фрагмент кода:

```
public void Button4_Click(object sender, EventArgs e)
    {
      Game.Image = imageList1.Images[3];
      sleep = 0;
      Button1.Enabled = true;
      Button2.Enabled = true;
    }
```

Кнопки «Кормить» и «Играть» станут снова доступны, обезьянка проснется, а показатель «Бодрость» вновь начнет падать (рисунок 3.8).

Рисунок 3.8 – Окно игры после нажатия на кнопку «Разбудить»

Если забыть про игру или не заметить, как одна из строк состояний достигла 1%, то будет отработан фрагмент кода (рисунок 3.9) ,и игра заканчивается. Высвечивается картинка, что обезьянка ушла, а кнопки управления блокируются (рисунок 3.10). При нажатии на крестик программа закрывается.

```
Game.Image = imageList1.Images[5];
Happy.Enabled = false;
Button1 .Enabled = false;
Button2 .Enabled = false;
Button3.Enabled = false;
Button4.Enabled = false;
Soon.Enabled = false;
```

Рисунок 3.9 – Фрагмент кода

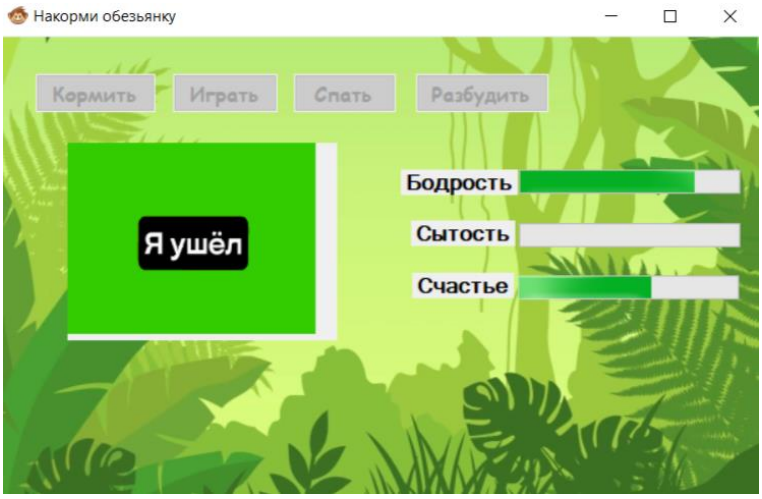


Рисунок 3.10 – Игра окончена

4 ТЕСТИРОВАНИЕ

В этом разделе будет представлено ручное и функциональное (NUnit) тестирование пользовательского интерфейса. Перед тем, как выполнять ручное тестирование, были составлены: тест план, чек-лист и Mind-Map.

Тест план:

1. Тестируется desktop приложение «Накорми обезьянку.Тамагочи» в рамках написания курсовой работы по дисциплине «Основы программирования»;

2. Операционная система: Windows;

3. В приложении необходимо протестировать следующие функции:

- Авторизация пользователя;
- Регистрация пользователя;
- Работа кнопок основного окна приложения.

4. Тестирование проводится в течение нескольких дней, по результатам тестирования предоставляется подробная отчетность и все тестовые документы, созданные в рамках проделанной работы;

5. Критерий начала тестирования – окончание составления нужной документации;

6. Критерий окончания тестирования – проведение всех основных действий пункта 7;

7. Необходимо провести ручное тестирование основного функционала согласно пункту 3. Такой тест позволит точно убедиться в работоспособности приложения. Для тестирования требуется написать тест-кейсы для функций, относящихся к основному функционалу. Также необходимо провести функциональное тестирование.

Тестовая стратегия – это то, как будет тестироваться продукт (рисунок 4.1.).

Название проекта		Introduction:	Функциональное решение для реализации приложения "Накорми обезьянку. Тамагочи"
Накорми обезьянку. Тамагочи		Objective:	Предоставление пользователям возможности управлять тамагочи при помощи компьютерной мыши
Дата завершения первого этапа	12.10.2023	Principles:	Выполнить работу в срок. Разобраться в реализации. Качественно выполнить проект и показать достойный результат
Дата сдачи проекта	26.01.2024		

In Scope		Entry Conditions(for test execution):		People		
1	Функциональное тестирование	1	Поставлена задача в рамках обучения	1	Менеджер-проекта	Глазырина Елизавета
2	Ручное тестирование	2	Собственное желание	2	Бизнес-аналитик	Глазырина Елизавета
Out of Score		**Exit Conditions(for test execution):**		3	Тестировщик	Глазырина Елизавета
1	Отсутствие прочих видов тестирования, т.к. представленных вполне достаточно	1	При тестировании не найдено крит.ошибок	**Test Environment**		
		2	Приложение работает на ОС Windows 10	1	Предполагается, что большинство пользователей будут работать в Windows-системах	
		3	Срок на выполнение задания подошел к концу	2	Предполагается, что большинство пользователей имеют компьютерную мышь	
		4	Все, включенное в план тестирования, завершено	**Timescales**		
		Risks		1	Планирование проекта	3 дня
		1	Отключение ПК по техническим причинам	2	Написание тест-кейсов, чек-листов	1 день
		2	Выход из строя компьютерной мыши	3	Выполнение тестов	2 дня
				4	Подведение итогов	2 дня

Рисунок 4.1 – Тестовая стратегия

Тест-кейс — это форма записи проверки, которую проводит тестировщик. По сути, это алгоритм действий, по которому предполагается тестировать уже написанную программу. В нём подробно прописаны шаги, которые нужно сделать для подготовки к тесту, сама проверка и ожидаемый результат [10]. Тест-кейс представлен на рисунках 4.2-4.4. Интеллект-карта (Mind-Map) отображена на рисунке 4.5. Чек-лист представлен на рисунке 4.6.

Проверка авторизации пользователя		
№	Действие	Результат
1	Ввести логин	В поле ввода "Логин" отображается admin1
2	Ввести пароль	В поле ввода "Пароль" отображается admin1
3	Войти через кнопку "Вход"	Открывается окно игры

Рисунок 4.2 – Тест-кейс «Проверка авторизации пользователя»

Проверка регистрации пользователя		
№	Действие	Результат
1	Открыть регистрацию через кнопку "Еще нет аккаунта?"	Открывается окно регистрации
2	Ввести логин	В поле ввода "Логин" отображается admin1
3	Ввести пароль	В поле ввода "Пароль" отображается admin1
4	Зарегистрироваться	Логин и пароль пользователя сохранены в базу данных

Рисунок 4.3 – Тест-кейс «Проверка регистрации пользователя»

Проверка работы игры		
№	Действие	Результат
1	Нажать на кнопку "Кормить"	Показатель "Сытость" увеличивается до 100%
2	Нажать на кнопку "Играть"	Меняется настроение обезьянки, показатель "Счастье" вырос на 10%
3	Нажать на кнопку "Спать"	Обезьянка спит, блокируются кнопки "Кормить" и "Играть", показатель "Бодрость" начинает расти
4	Нажать на кнопку "Разбудить"	Обезьянка просыпается, показатель "Бодрость" перестает расти и начинает убывать. Кнопки "Кормить" и "Играть" вновь функционируют
5	Закрыть программу через крестик	Закрывается основное окно, выход из программы

Рисунок 4.4 – Тест-кейс «Проверка работы игры»

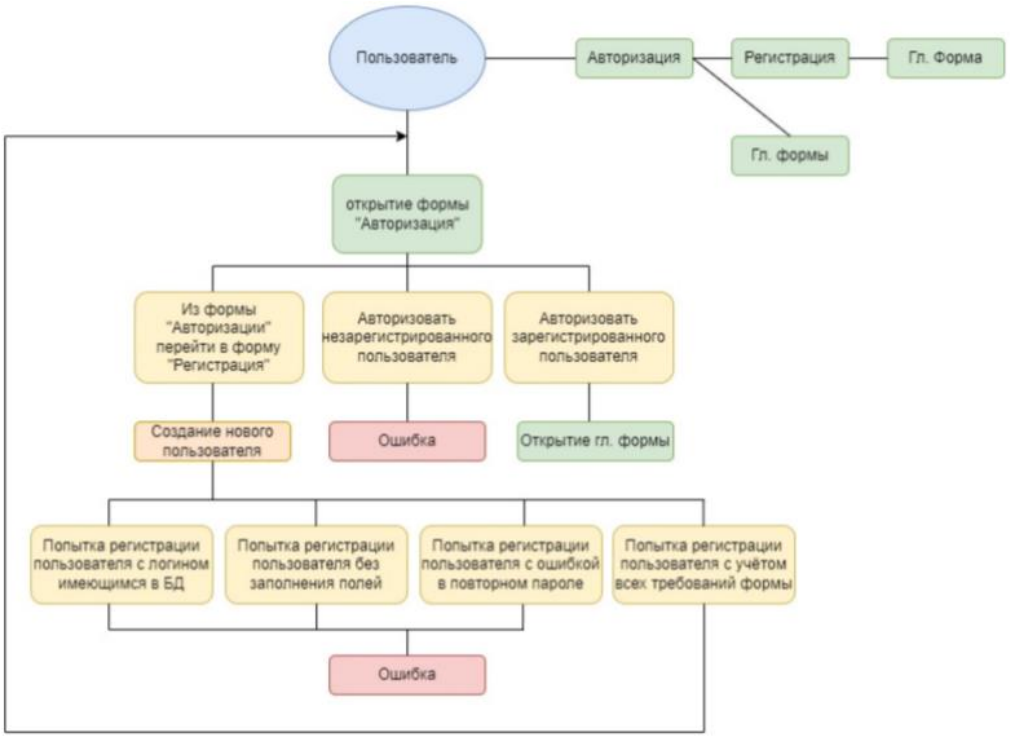


Рисунок 4.5 – Интеллект-карта

Проверка	Результат	
функция	Windows 10	Windows 7
Кнопка "Вход"	ОК	ОК
Ввод пароля в авторизации	ОК	ОК
Ввод логина в авторизации	ОК	ОК
Ввод логина в регистрации	ОК	ОК
Ввод пароля в регистрации	ОК	ОК
Кнопка "Зарегистрироваться"	ОК	ОК
Кнопка "Кормить"	ОК	ОК
Кнопка "Играть"	ОК	ОК
Кнопка "Спать"	ОК	ОК
Кнопка "Разбудить"	ОК	ОК

Рисунок 4.6 – Чек-лист

4.1 Функциональное тестирование

Согласно заданию нужно протестировать функционал основной игры, а именно, правильно ли работают кнопки и свою ли задачу они выполняют. Для этого данный класс переводим в режим доступа «Public». Кнопки

переводим также в общедоступный режим. На рисунке 4.4.1 представлено успешное прохождение автоматических тестов, написанных с использованием библиотека NUnit.

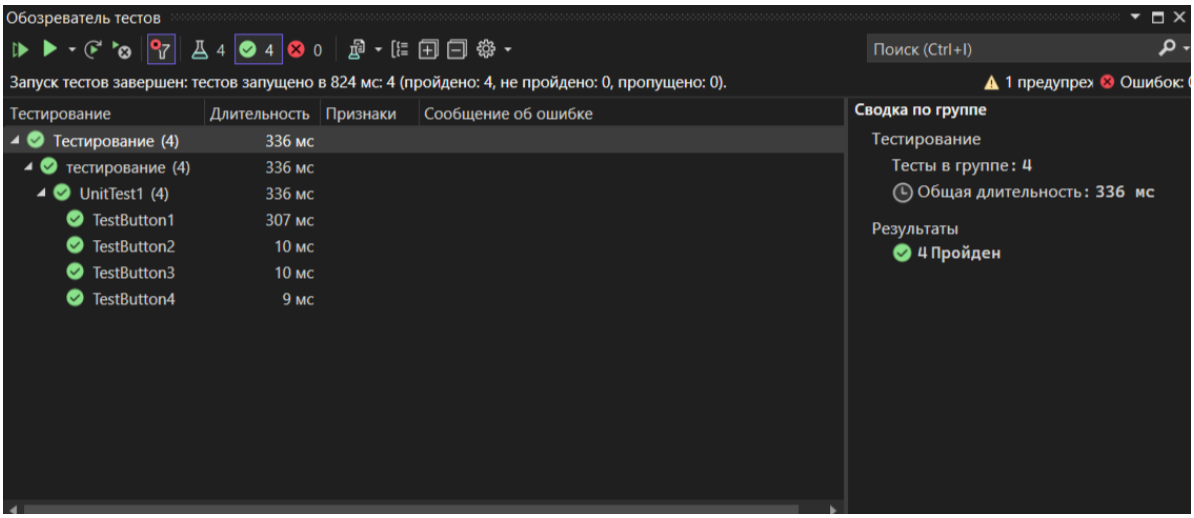


Рисунок 4.1.1 – Функциональное тестирование

4.2 Ручное тестирование

На данном этапе будем самостоятельно поэтапно проверять работоспособность программы с намеренным выполнением ошибок. На данном этапе будут проверяться формы «Авторизация» и «Регистрация», поскольку в основном окне присутствуют только кнопки. С помощью которых можно управлять питомцем, а по функциональному тестированию работают корректно, подробное описание каждой кнопки было рассмотрено в разделе «Реализация приложения» (стр.30).

Начнем с формы «Авторизация». Попробуем авторизоваться и войти в игру без каких-либо данных (рисунок 4.2.1). Следом заполним только одно из двух полей (рисунок 4.2.2) и потом попробуем авторизовать не зарегистрированного пользователя (рисунок 4.2.3).

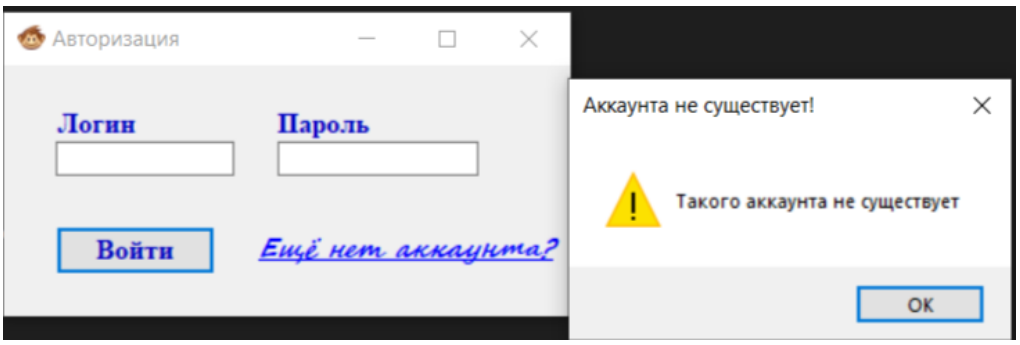


Рисунок 4.2.1 – Попытка авторизоваться с пустыми полями

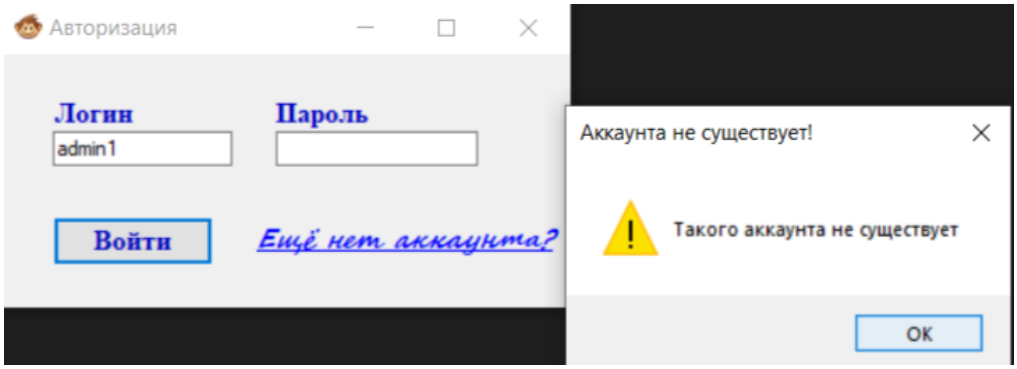


Рисунок 4.2.2 – Попытка авторизоваться с одним пустым полем

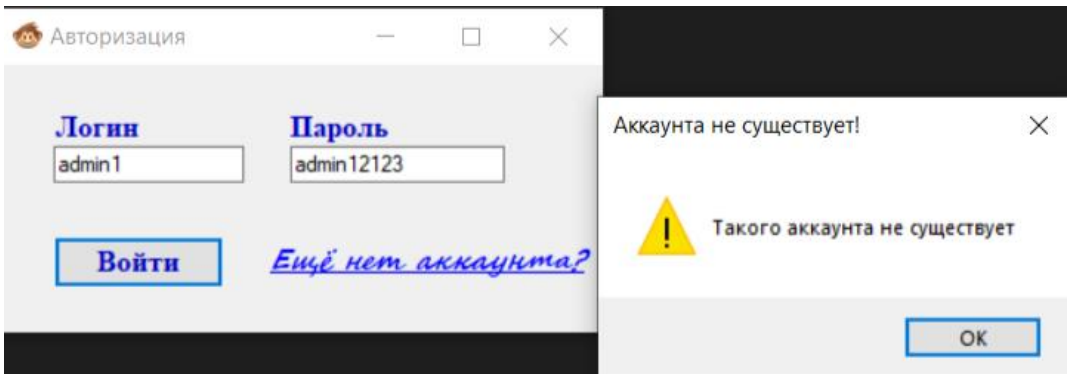


Рисунок 4.2.3 – Попытка авторизовать не зарегистрированного пользователя

Таким образом, можно заметить, что попасть в игру не зарегистрированный пользователь не может, все части кода программы работают корректно.

Пробуем провести тесты над формой «Регистрация», переключившись на данную форму через кнопку «Еще нет аккаунта?».

Необходимо будет протестировать следующие случаи: создание аккаунта с пустым логином и паролем, только с введенным логином, только с введенным паролем, а также попробуем повторно зарегистрировать уже зарегистрированного пользователя. Результаты данных тестов представлены на рисунках 4.2.4 – 4.2.7 соответственно.

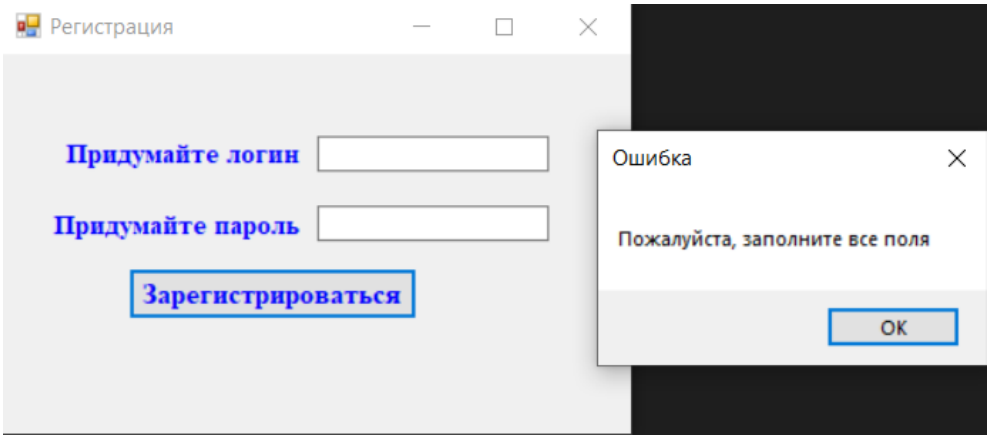


Рисунок 4.2.4 – Попытка зарегистрироваться с пустыми полями

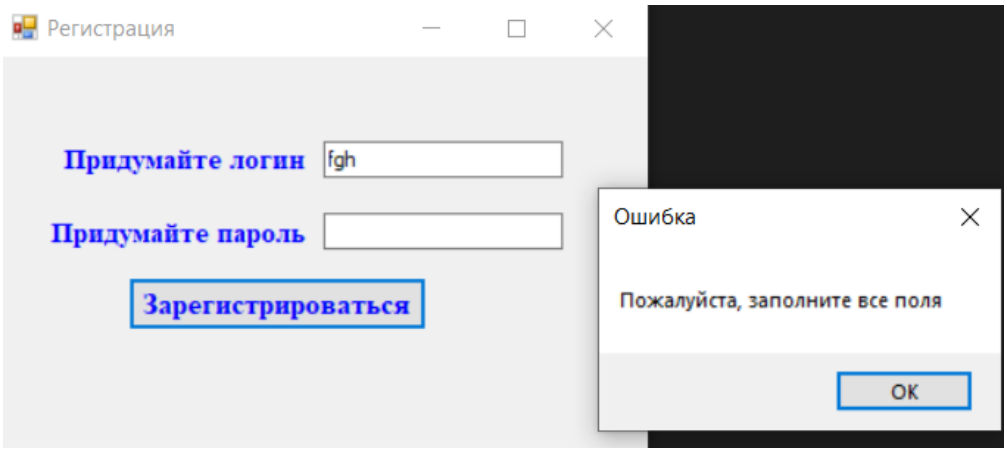


Рисунок 4.2.5 – Попытка зарегистрироваться с пустым паролем

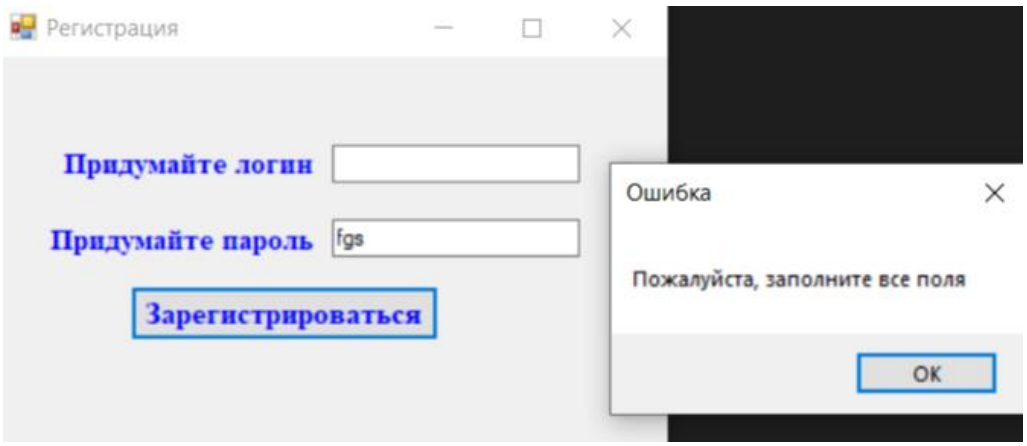


Рисунок 4.2.6 – Попытка зарегистрироваться с пустым логином

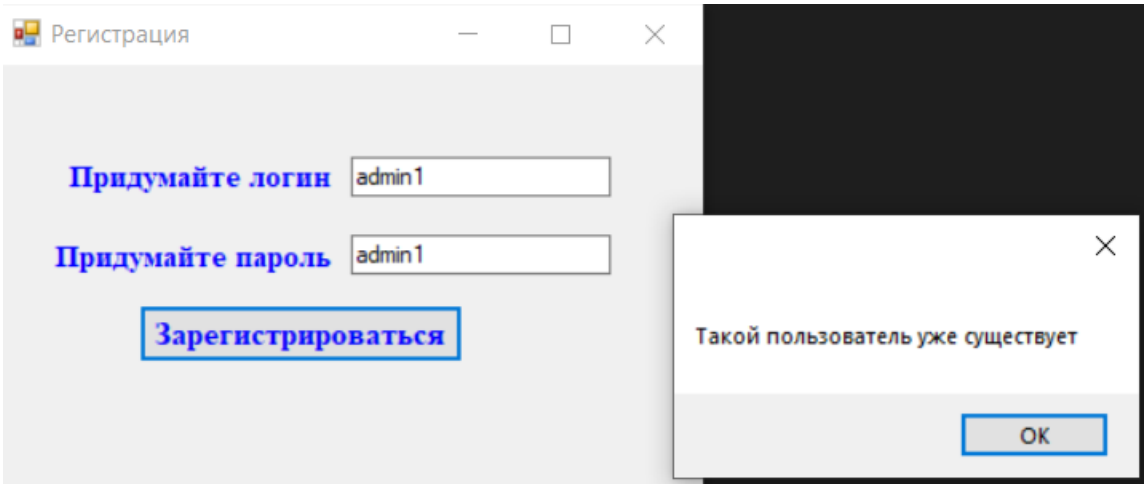


Рисунок 4.2.7 – Попытка зарегистрировать уже зарегистрированного пользователя

Видим, что все фрагменты кода в форме регистрации написаны верно, и создают нового пользователя только при условии ввода корректных данных.

Попробуем авторизовать только что зарегистрированного пользователя (рисунки 4.2.8-4.2.10).

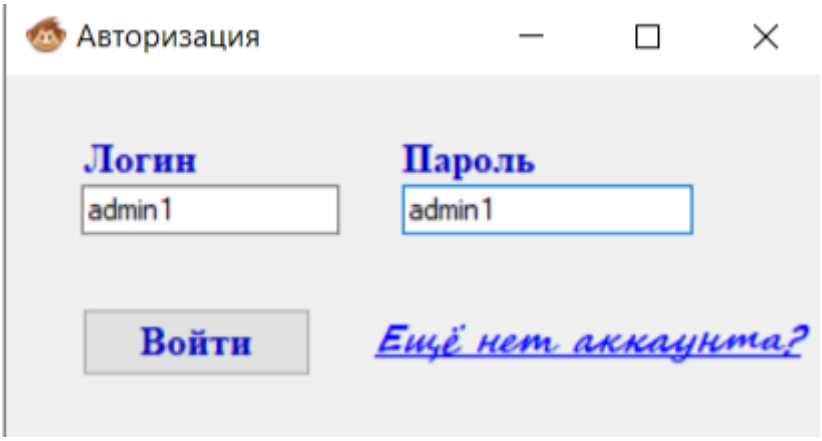


Рисунок 4.2.8 – Авторизация существующего пользователя

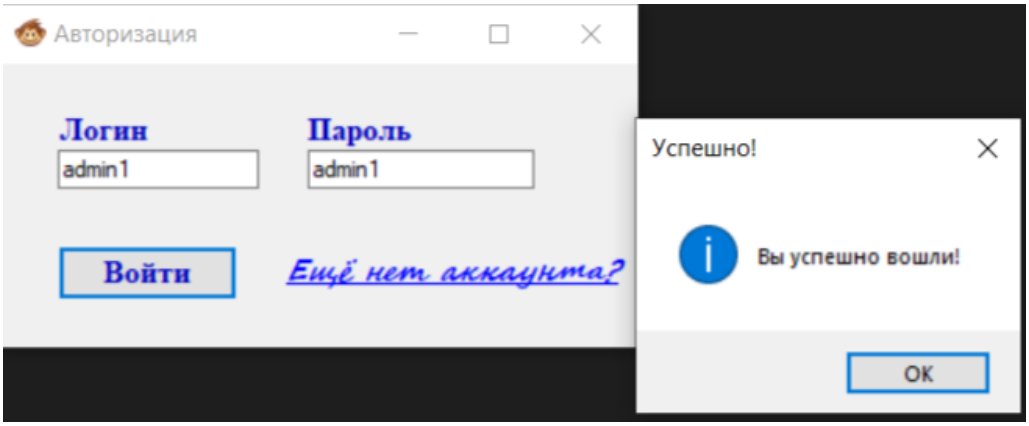


Рисунок 4.2.9 – Успешная авторизация существующего пользователя

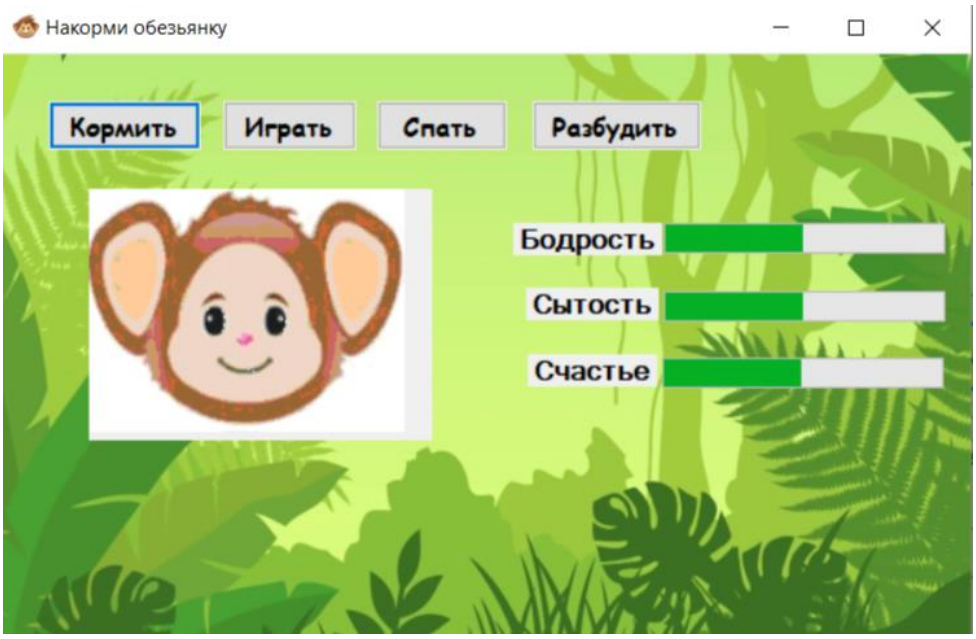


Рисунок 4.2.10 – Переход на окно игры

Заметим, что при соблюдении всех правил регистрации и авторизации можно успешно попасть на форму игры. Таким образом, программа проходит ручное тестирование.

Заключение

В ходе выполнения данной курсовой работы было разработано приложение, реализующее игру «Накорми обезьянку.Тамагочи» для взаимодействия пользователя с виртуальным питомцем. Данное приложение написано на языке программирования C# в среде разработки Microsoft Visual Studio Community 2022 (64-разрядная версия) – Current Версия 17.5.5 на платформе .NET Framework. Хранение данных было организовано с помощью подключения СУБД SQL Server Management Studio.

В ходе выполнения курсовой работы были освоены следующие навыки:

– навыки создания десктопных приложений на языке программирования C#;

– навыки работы с СУБД SQL Server Management Studio.

Для достижения поставленной цели были решены следующие задачи:

– анализ темы «Тамагочи»;

– описание алгоритма через словесное описание и создание блок-схемы алгоритма;

– определение временной сложности алгоритма;

– построение диаграммы классов;

– реализация механизма авторизации и регистрации пользователя;

– реализация механизма работы с БД;

– реализация механизма добавления пользователей;

– функциональное тестирование программы;

– ручное тестирование программы.

Исходный код проекта размещен на репозитории GitHub: https://github.com/ElizavetaGLZRN/kursovaya2024.

Список использованных источников

1. Образовательный стандарт вуза ОС ТУСУР 01-2021 [Электронный ресурс]: сайт ТУСУРа URL: https://regulations.tusur.ru/storage/150499/;

2. Харченко С.С. Основы программирования: учебно-методическое пособие по курсовой работе. – Томск: В-Спектр, 2019. – 48 с.;

3. https://habr.com/ru/companies/vdsina/articles/560174/;

4. https://www.igromania.ru/article/31952/Istoriya_Tamagochi-karmannoy_igrushki_kotoraya_svodila_s_uma_detey_v_90-e.html;

5. https://mchost.ru/articles/chto-takoe-python/;

6. https://learn.microsoft.com/ru-ru/visualstudio/get-started/;

7. https://gb.ru/blog/luchshie-ide-dlya-razrabotki-na-c/;

8. https://learn.microsoft.com/ru-ru/sql/ssms/;

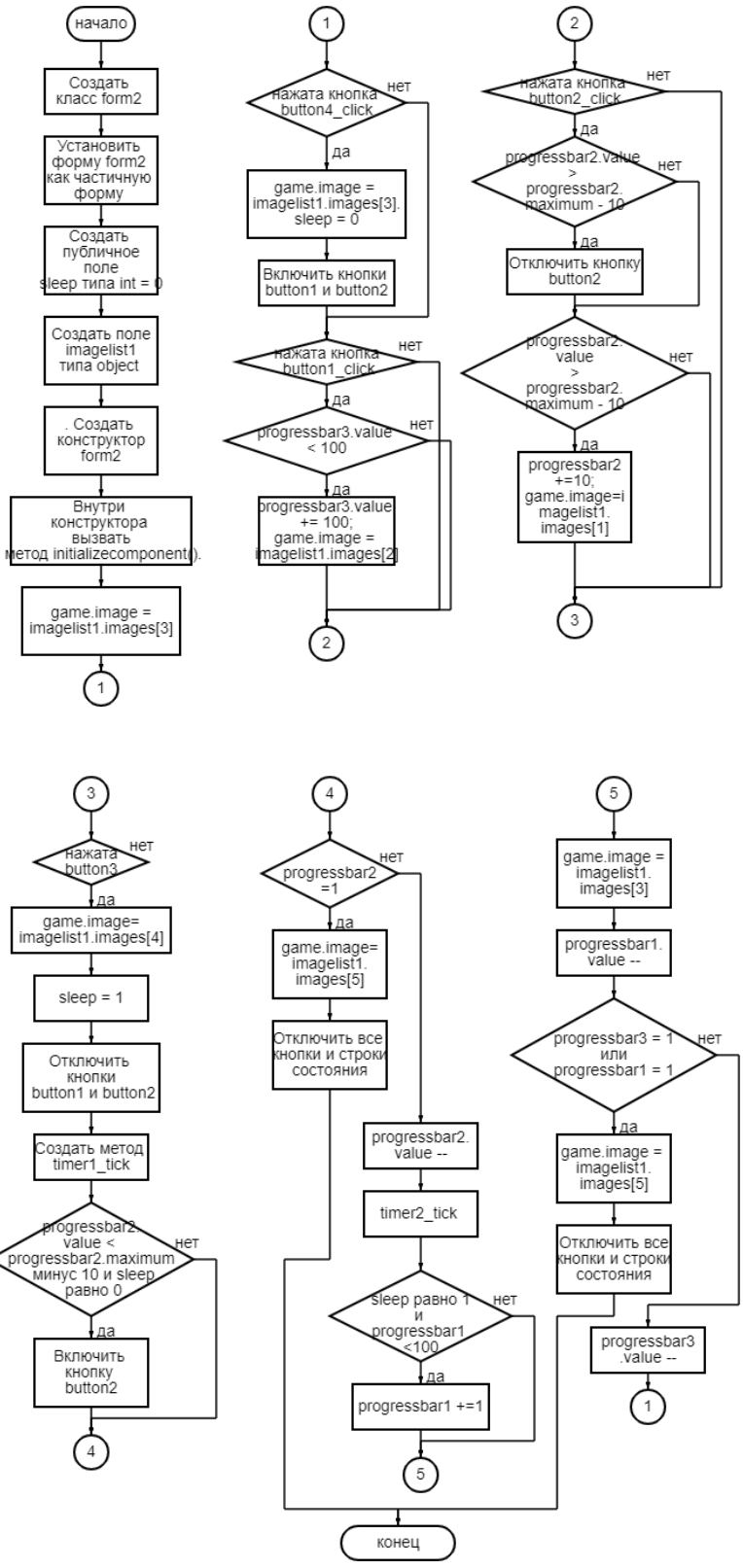
9. https://www.yuripetrov.ru/edu/python/ch_06_01.html;

10. https://practicum.yandex.ru/blog/chto-takoe-test-keys-i-kak-ego-sostavit/;

Приложение А

(Обязательное)

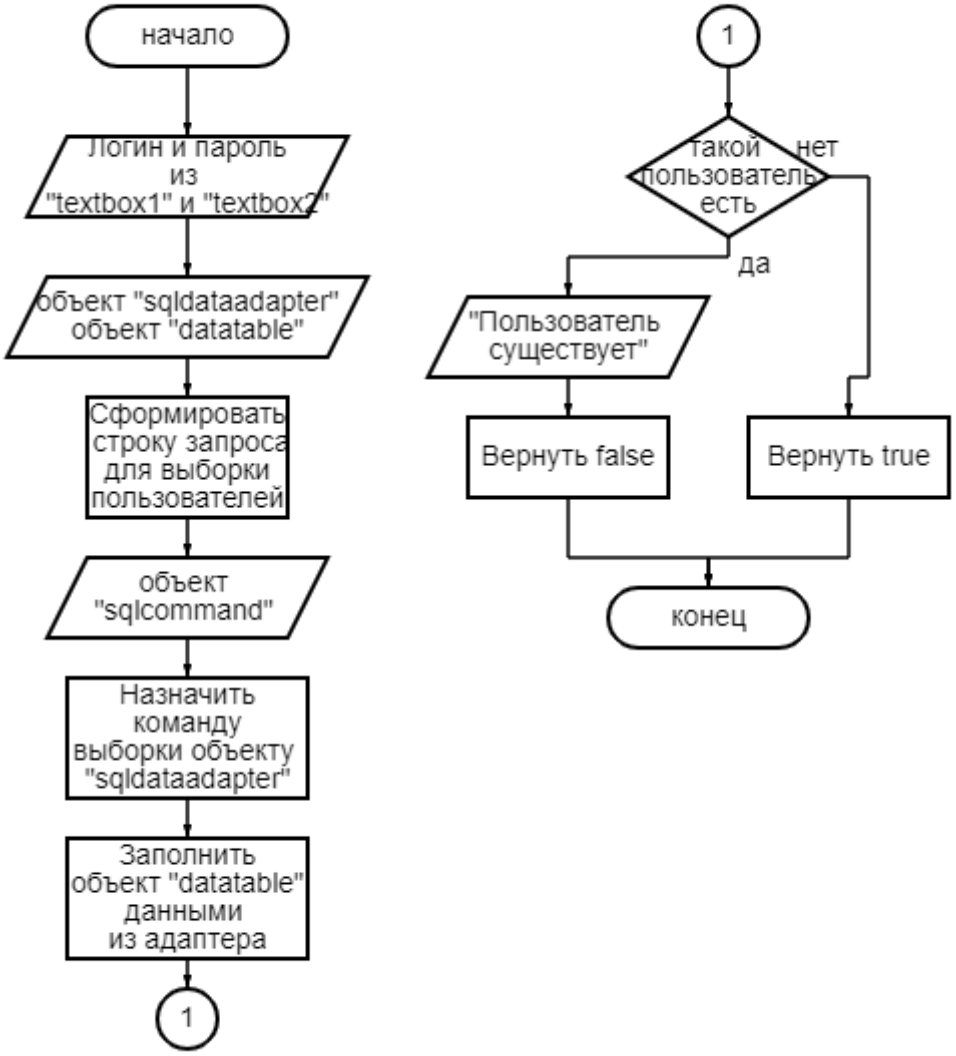
Блок-схема к алгоритму А



Приложение Б

(Обязательное)

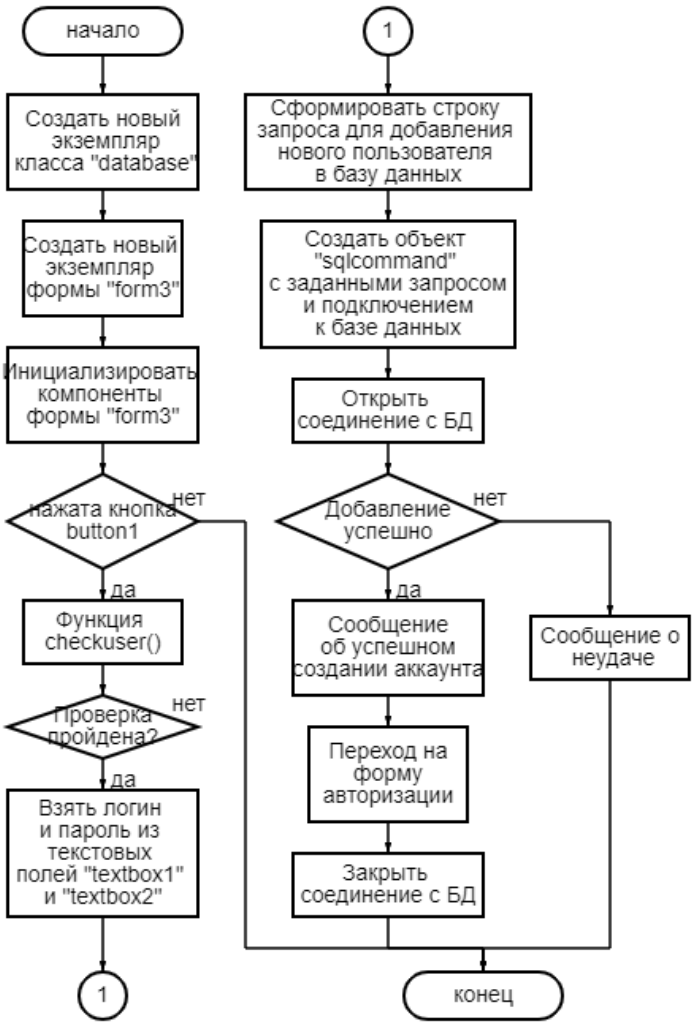
Блок-схема к алгоритму В



Приложение В

(Обязательное)

Блок-схема к алгоритму С



Приложение Г

(Обязательное)

Блок-схема к алгоритму D