**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ**

**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ**

**«ПЕРВЫЙ МОСКОВСКИЙ ОБРАЗОВАТЕЛЬНЫЙ КОМПЛЕКС»**

(ГБПОУ «1-й МОК»)

Допустить к защите

Заведующая отделением среднего профессионального образования факультета «Информационные технологии и управление»

\_\_\_\_\_\_\_\_\_\_\_\_/Пашохонова Е.Е.

 «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г

**ДИПЛОМНЫЙ ПРОЕКТ**по специальности

09.02.04 «Информационные системы (по отраслям)»

(наименование специальности)

Создание информационной подсистемы поддержки самостоятельного обучения профессиональному английскому языку.

**Тема: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Седов Артем Вадимович

группы \_\_\_\_\_

41ИС

(фамилия, имя, отчество)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тузовский Анатолий Федорович

(фамилия, имя, отчество)

**Работа защищена «\_\_»** июня 2021 года **с оценкой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Секретарь Государственной

экзаменационной комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Москва

2021

Оглавление

[ВВЕДЕНИЕ 3](#_Toc70360504)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОЙ ИНФОРМАЦИОННОЙ ПОДСИСТЕМЫ 5](#_Toc70360505)

[1.1 Описание предметной области 5](#_Toc70360506)

[1.2 Функциональное моделирование 5](#_Toc70360507)

[2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 12](#_Toc70360508)

[2.1 Моделирование системы с помощью языка UML 12](#_Toc70360509)

[2.2 Проектирование информационной базы данных 18](#_Toc70360510)

[2.3 Моделирование пользовательского интерфейса 20](#_Toc70360511)

[3. ОПИСАНИЕ РЕАЛИЗАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ 24](#_Toc70360512)

[3.1 Описание архитектуры программного обеспечения 24](#_Toc70360513)

[3.2 Создание базы данных 24](#_Toc70360514)

[3.3 Разработка программного кода 28](#_Toc70360515)

[3.4 Инструкции пользователей информационной системы 43](#_Toc70360516)

[4 ТЕСТИРОВАНИЕ 47](#_Toc70360517)

[4.1 Теория тестирования 47](#_Toc70360518)

[4.2 Баг репорт 48](#_Toc70360519)

[ЗАКЛЮЧЕНИЕ 58](#_Toc70360520)

[ССЫЛКИ 60](#_Toc70360521)

# Введение

В последние десятилетия изучение иностранных языков вызывает повышенный интерес. При этом отмечается возрастающая роль, которую играют иностранные языки в осуществлении влияния на сознание и деятельность людей. Также необходимо учитывать, что знание языков может играть важную роль и давать некотор ые преимущества в личной и профессиональной коммуникации.

Самый распространённый иностранный язык для обучения является английский язык. Люди, которые идут в институты или техникумы на специальность в сфере IT обязаны знать английский, так как все понятия написаны на английском. В связи с этим, многие люди вынуждены самостоятельно заниматься вопросом изучения. Конечно же, это нелегкий путь, но наличие современных технологий способно во много раз упростить данный процесс.

Данная работа направлена на то, чтобы пользователь при работе с информационной подсистемой мог легко обучатся профессиональному английскому языку. Интерфейс очень простой и пользователя-дружелюбный, который приведет к быстрому и легкому обучению.

Целью дипломного проекта является проектирование и разработка подсистемы обучения английскому языку, которое позволит облегчить изучение и практики английского языка в IT сфере.

Для осуществления цели нужно выполнить следующие задачи:

* провести анализ использования информационных технологий в процессе обучения английскому языку;
* провести анализ предметной области;
* выполнить описание предметной области с использованием моделей структурного и объектно-ориентированного подхода;
* определить требования к функциональным характеристикам будущей подсистемы в UML-моделях;
* выполнить проектирование подсистемы обучения английскому языку:
  + описание решаемых задач
  + логическая модель данных
  + макет графического интерфейса
* разработать базу данных с использованием СУБД Microsoft SQL Server и SQL Server Management Studio (SSMS);
* разработать программный код подсистемы с помощью среды разработки Visual Studio.
* провести тестирование разработанной подсистемы с целью определения и повышения ее качества.

# 1. анализ предметной области и требования к разрабатываемой Информационной подсистемы

## 1.1 Описание предметной области

Информационная система «Rattlesnake» специализируется в профессиональном самообучении английскому языку в IT-сфере пользователя путем предоставления много изучаемого материала, тестов по темам и словаря.

Процесс обучения систематизирован. Пользователю дается материал для изучения, после чего он проходит тестовую часть для отслеживания прогресса и перехода на следующую тему, которая будут уже посложнее.

Сначала пользователь выбирает первую тему и изучает учебный материал. Учебный материал предоставляется пользователю по темам и по уровню сложности. После того как он изучил материал он может перейти на тестирование.

Для каждой учебной темы есть несколько тестов, который пользователь должен проходить чтобы перейти на следующую тему. Он также может проходить эти тесты несколько раз и результат их будут отображаться в профиле пользователя. Результат тестов отображается в процентах.

После прохождении всех тестов по теме, пользователь может переходить на следующий тему, изучая учебный материал и также выполняя тестовые задания по ним.

В прогрессе обучения, сложные профессиональные слова из учебных материалов будут добавляться в словарь. Пользователь может просматривать новые слова в словаре, посмотреть перевод на русский или просто повторить.

## 1.2 Функциональное моделирование

Функциональное моделирование отражает последовательность бизнес-процессов, протекающих на предприятии, а также то, как эти процессы обрабатывают и преобразуют физические и информационные ресурсы. Функциональное моделирование производиться с помощью диаграмм серии IDEF (IDEF0 - IDEF14) и DFD относящихся к методологии SADT.

SADT (Structured Analysis and Design Technique) — методика структурного анализа и проектирования, интегрирующая процесс моделирования, управление конфигурацией проекта, использование дополнительных языковых средств и руководство проектом со своим графическим языком.

**IDEF0**

Исторически IDEF0 как стандарт был разработан в 1981 году в рамках обширной программы автоматизации промышленных предприятий, которая носила обозначение ICAM (Integrated Computer Aided Manufacturing). Семейство стандартов IDEF унаследовало свое обозначение от названия этой программы (IDEF=Icam DEFinition), и последняя его редакция была выпущена в декабре 1993 года Национальным Институтом по Стандартам и Технологиям США (NIST).

Целью методики является построение функциональной схемы исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы.

Каждая IDEF0-диаграмм а содержит блоки и дуги. Блоки изображают функции моделируемой системы. Дуги связывают блоки вместе и отобра­жают взаимодействия и взаимосвязи между ними.

Функциональные блоки (работы) на диаграммах изображаются прямоугольниками, означающими поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты.

IDEF0 требует, чтобы в диаграмме было не менее трех и не более шести блоков. Эти ограничения поддерживают сложность диаграмм и модели на уровне, доступном для чтения, понимания и использования.

Каждая сторона блока имеет особое, вполне определенное назначение. Левая сторона блока предназначена для входов, верхняя - для управления, правая – для выходов, нижняя - для механизмов. Такое обозначение отражает определенные системные принципы: входы преобразуются в выходы управление ограничивает или предписывает условия выполнения преобразований, механизмы показывают, что и как выполняет функция.

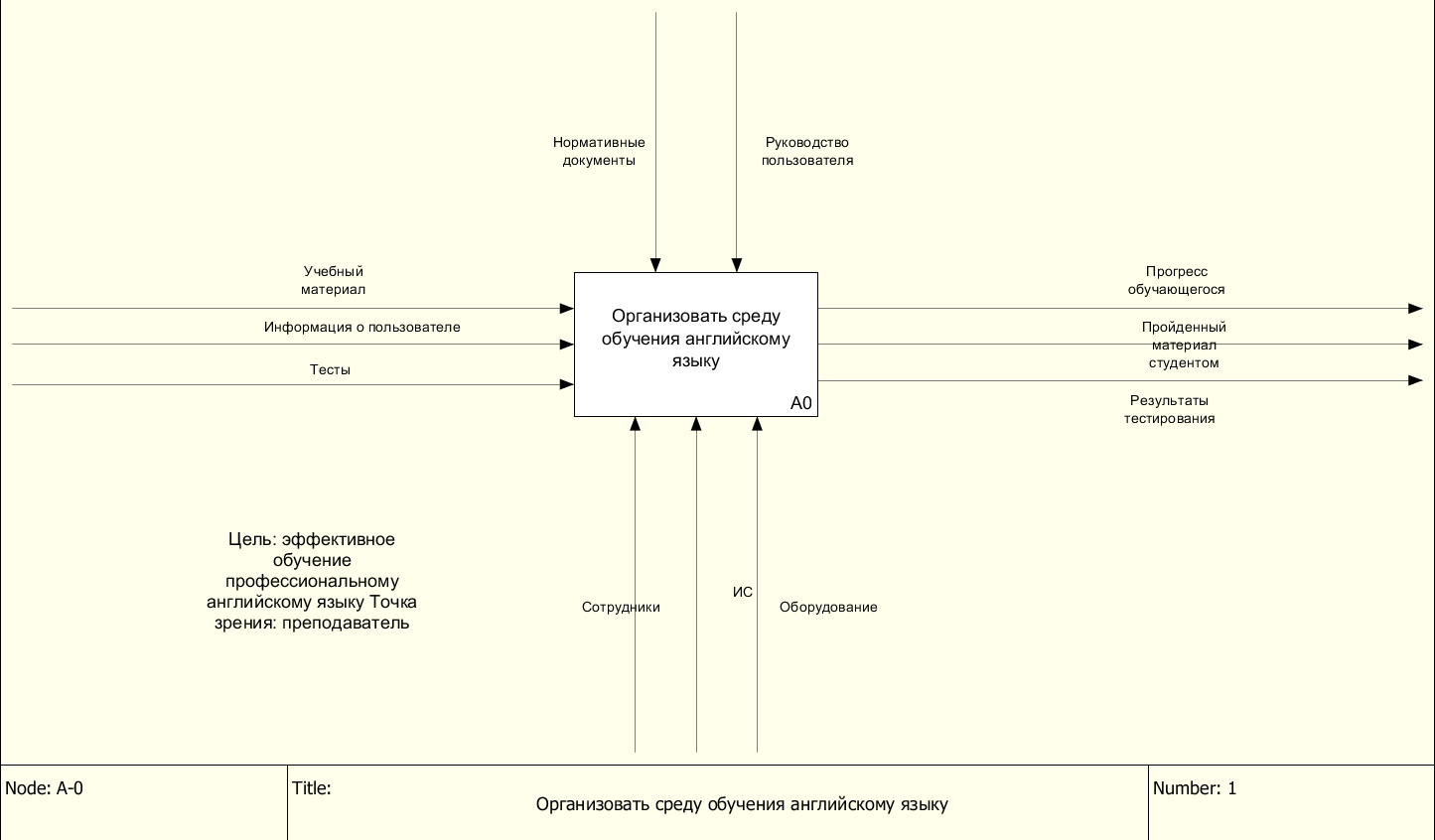
Блоки в IDEF0 размещаются по степени важности, как ее понимает автор диаграммы. Этот относительный порядок называется доминированием. Доминирование понимается как влияние, которое один блок оказывает на другие блоки диаграммы. Например, самым доминирующим блоком диаграммы может быть либо первый из требуемой последовательности функций, либо планирующая или контролирующая функция, влияющая на все другие.

Взаимодействие работ с внешним миром и между собой описывается в виде стрелок, изображаемых одинарными линиями со стрелками на концах. Стрелки представляют собой некую информацию и именуются существительными.

**Контекстная диаграмма**

В диаграмме бизнес-процессом является «Организовать среду обучения английскому языку». Представлен в функциональном блоке, для выполнения которого на входе используется информация о пользователе, учебные материалы и тесты.

В качестве управления выступают нормативные документы и руководство пользователя. Инструментами выполнения главной бизнес-функции служат сотрудники, сама информационная система и оборудование (компьютеры). На выходе мы получаем прогресс пользователя, пройденный материал и результат тестирования. (Рисунок 1).



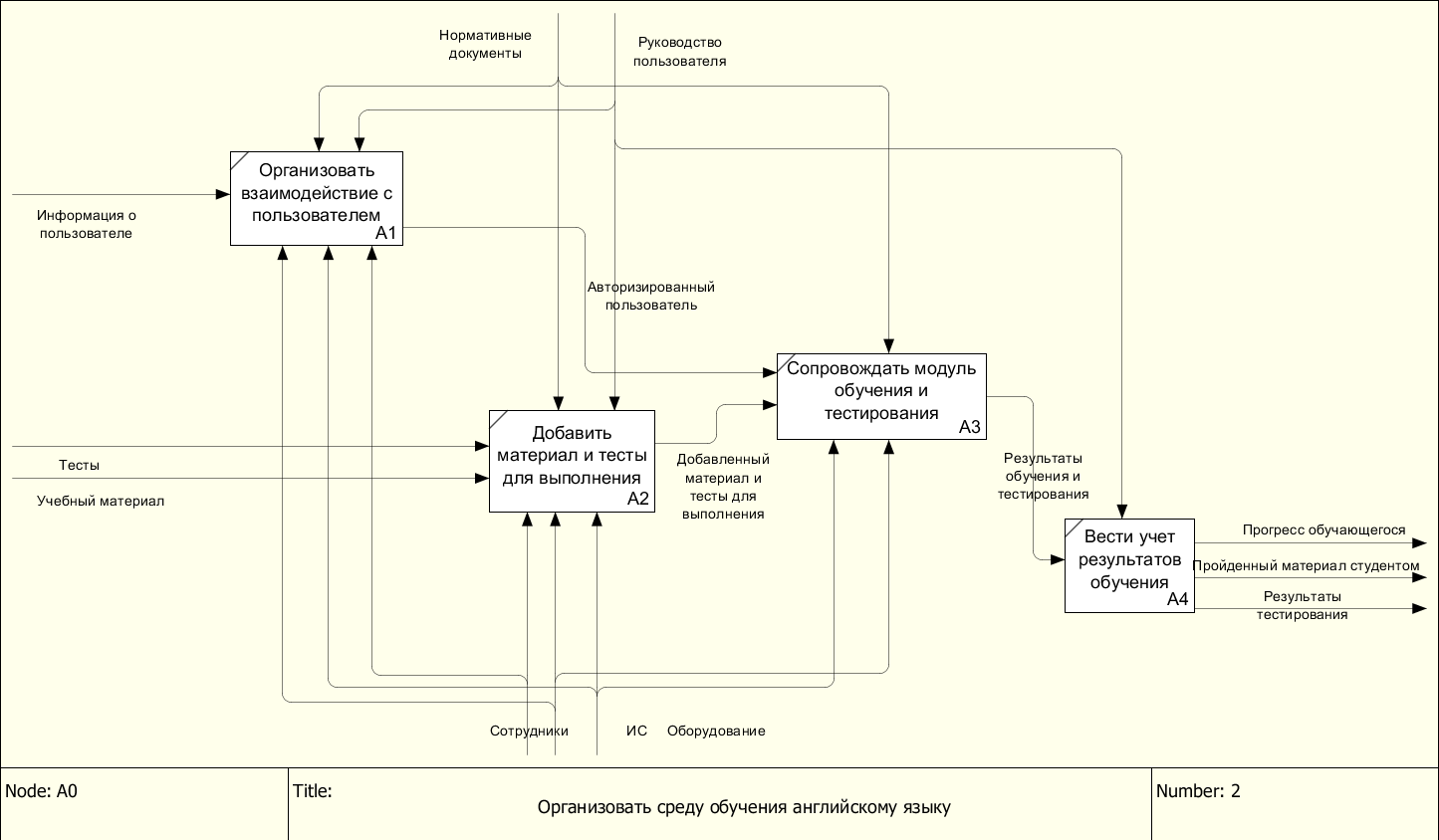
**Рисунок 1 «Контекстная диаграмма»**

**Диаграмма декомпозиция**

Проанализировав деятельность рекламного агентства, и проведя предпроектное исследование, можно выделить три основных бизнес-подпроцесса:

1. Организовать взаимодействие с пользователем;
2. Добавить материал и тесты для выполнения;
3. Сопровождать модуль обучения и тестирования;
4. Вести учет результатов обучения.

В подпроцесс «Добавить материал и тесты для выполнения» на входе идут учебные материалы и тесты. Это осуществляется при управлении руководством пользователя и нормативных документов и механизмов: сотрудники, ИС и оборудования. На выходе получаем «Добавленный материал и тесты для выполнения», которые входит в третий подпроцесс, «Сопровождать модуль обучения и тестирования». Первый подпроцесс «Организовать взаимодействие с пользователем» также входит в третий подпроцесс. На выходе третьего подпроцесса является «Результаты обучения и тестирования», которые входят в четвертый подпроцесс – «Вести учет результатов обучения». Из четвертого подпроцесса выходит пройденный материал, прогресс и результаты тестирования (Рисунок 2).



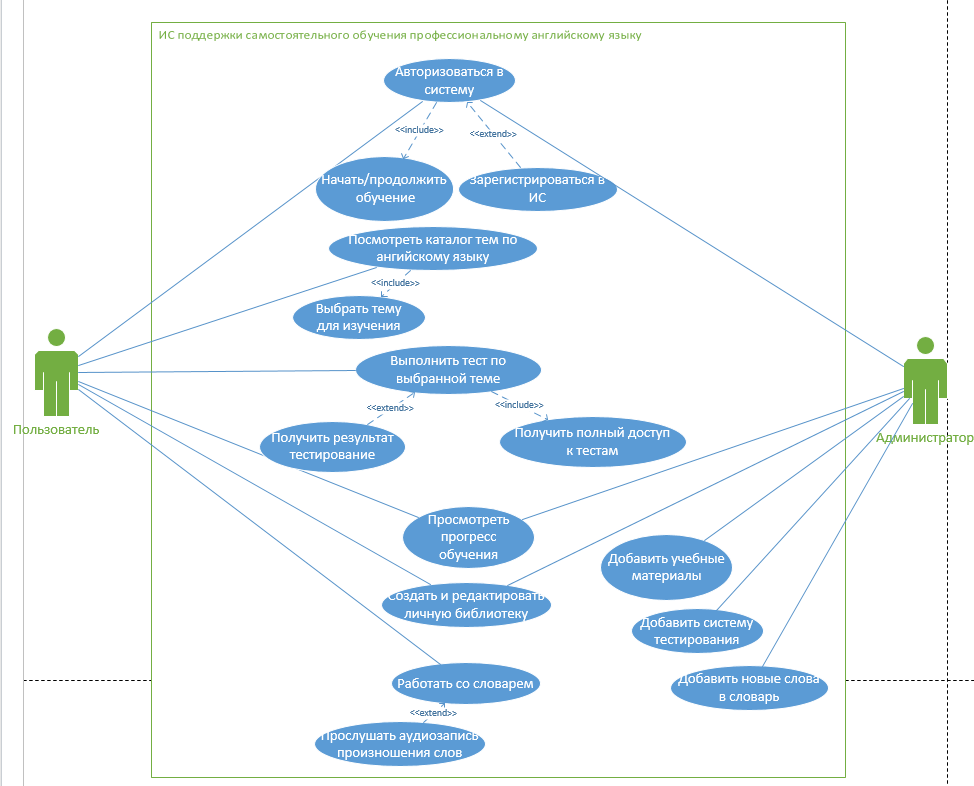
**Рисунок 2 «Диаграмма декомпозиции»**

**Диаграмма прецедентов**

Диаграмма вариантов прецедентов – это тип поведенческой диаграммы UML, который часто используется для анализа различных систем. Они позволяют визуализировать различные типы ролей в системе и то, как эти роли взаимодействуют с системой. Это руководство по диаграмме вариантов использования охватывает следующие темы и поможет вам лучше создавать сценарии использования.

Как видно на (Рисунок 3), происходит взаимодействие системы с внешним миром, имеются два основных актера, такие как администратор и пользователь. Пользователь и администратор могут авторизоваться в подсистему. Прецедент «Начать/продолжить обучение» является включением в авторизацию, пользователь сможет продолжить с того место, где он закончил в прошлый раз.

**Рисунок 3 – Диаграмма прецедентов**

 Из авторизации расширяется прецедент «Зарегистрироваться в ИС». Далее пользователь может посмотреть каталог тем в это включает выбрать тему для изучения и получение полный доступ к тестам по теме. После этого пользователь может выполнить тест. Так же пользователь может посмотреть прогресс своего обучения в его профиле и работать со словарём.

Администратор может создавать новые учебные материалы и добавлять новые тесты по учебному материалу.

# 2 проектирование системы

## 2.1 Моделирование системы с помощью языка UML

**Что такое UML?**

Унифицированный язык моделирования (UML) был разработан с целью обеспечить единый визуальный язык с богатой семантикой и развернутым синтаксисом для проектирования и внедрения программных систем со сложной структурой и комплексным поведением.

UML напоминает стандарты, используемые в других отраслях, и поддерживает диаграммы нескольких типов. В целом, диаграммы UML описывают границы, структуру и поведение как всей системы, так и отдельных объектов в ее составе.

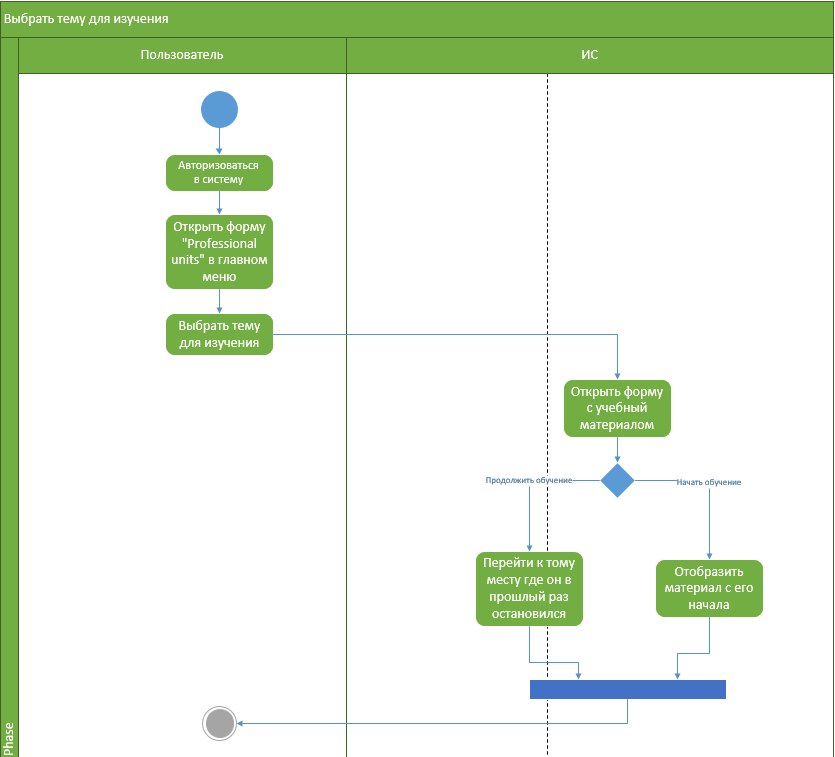
Моделирование необходимо для понимания системы. При этом единственной модели никогда не бывает достаточно. Напротив, для понимания любой нетривиальной системы приходится разрабатывать большое количество взаимосвязанных моделей. В применении к программным системам это означает, что необходим язык, с помощью которого можно с различных точек зрения описать представления архитектуры системы на протяжении цикла ее разработки

Словарь и правила такого языка, как UML, объясняют, как создавать и читать хорошо определенные модели, но ничего не сообщают о том, какие модели и в каких случаях нужно создавать. Это задача всего процесса разработки программного обеспечения. Хорошо организованный процесс должен подсказать, какие требуются артефакты, какие ресурсы необходимы для их создания, как можно использовать эти артефакты, чтобы оценить выполненную работу и управлять проектом в целом.

Объектно-ориентированные языки преобладают в сфере программирования, так как моделируют объекты реального мира. UML сочетает в себе несколько разновидностей объектно-ориентированной нотации — объектно-ориентированный дизайн, технику объектного моделирования и разработку объектно-ориентированного программного обеспечения.

**Диаграммы деятельности**

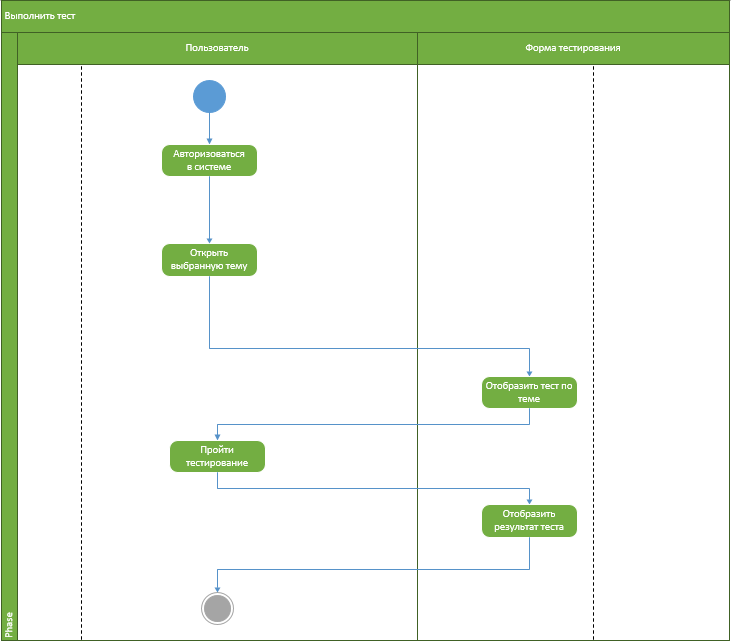
Данная диаграмма деятельности составлена по прецеденту «Выбрать тему для изучения». Существует две дорожки: пользователь и информационная система (ИС). Создается начальный узел на дорожке пользователя, он авторизуется в систему и открывает форму «Professional units» в главном меню. Далее пользователь выбирает тему для изучения, после этого отправляется сообщение дорожки ИС, где происходит действие, которое открывает форму с учебным материалом. Из этого действия идет решение, которое разветвляется на два пути: если пользователь уже начал изучать темы и, если пользователь впервые открыл выбранную тему. Если пользователь уже начал изучать темы, то происходит действие, где он открывает форму на том месте, где он последний раз остановился, а если нет, то происходит действе, где открывается форма с ее начала. Эти два действия входят в соединительный узел, который ведет к конечному узлу на дорожки пользователя (Рисунок 4).



**Рисунок 4 – Диаграмма деятельности по прецеденту**

**«Выбрать тему для изучения»**

Следующая диаграмма деятельности составлена по прецеденту «Выполнить тест». Существует две дорожки: пользователь и информационная система (ИС). Создается начальный узел на дорожке пользователя, он авторизуется в систему и открывает выбранную тему, по которой пользователь хочет пройти тест. Следующие действие — отобразить тест после чего переходит к действие выполнить тест на дорожки пользователь. После выполнение теста отправляется сообщение ИС, где происходит действие, которое отображает результат теста, которое соединяется с конечным узлом на дорожке пользователь (Рисунок 5).

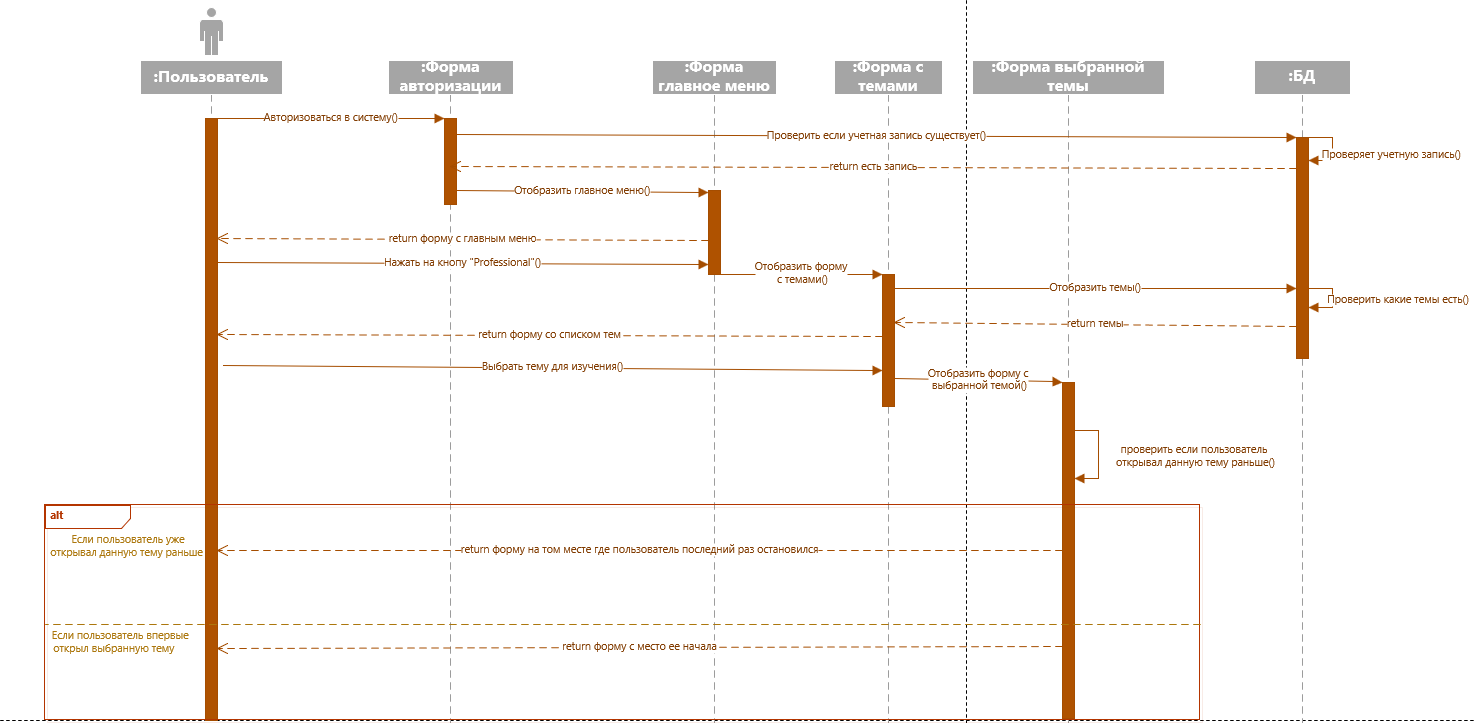


**Рисунок 5 – Диаграмма деятельности по прецеденту**

**«Выполнить тест по выбранной теме»**

**Диаграмма последовательности**

Диаграмму инициирует объект пользователь начинает действие и создается фокус управления, который указывает что происходит в данный момент, в этом случаи он отправляет сообщение форме авторизации: авторизоваться в систему, форма авторизации делает рефлексивное сообщение проверяет если учетная запись существует и создается фокус управления в объекте «:Форма главное меню», которому отправляется сообщение «Отобразить главное меню». Далее «:Форма главное меню» возвращает сообщение с «форма главное меню», после чего пользователь отправляет сообщение «Нажать на кнопку «Professional units»()». После чего объект «:Форма главное меню» отправляет сообщение объекту «:Форма с темами», где создается фокус управления, отобразить форму с темами, что «:Форма с темами» и возвращает пользователю. Далее объект «Пользователь» отправляет сообщение объекту «:Форма с темами», «Выбрать тему для изучения()», после чего объект «:Форма с темами» отправляет сообщение объекту «:Форма выбранной темы», чтобы она отобразила форму с выбранной темой. Объект «:Форма выбранной темы» делает рефлексивное сообщение, проверяет если пользователь открывал данную тему раньше. После чего создается оператор взаимодействия альтернативных фрагментов: фрагмент если пользователь уже открывал данную тему раньше, то возвращает форму на том месте, где пользователь последний раз остановился, и фрагмент если пользователь впервые открыл выбранную тему, то возвращает форму с места ее начала (Рисунок 6).



**Рисунок 6 – Диаграмма последовательности по прецеденту**

**«Выбрать тему для изучения»**

**Диаграмма кооперации**

Понятие кооперации (collaboration) является одним из фундаментальных понятий в языке UML. Оно служит для обозначения множества взаимодействующих с определенной целью объектов в общем контексте моделируемой системы. Цель самой кооперации состоит в том, чтобы специфицировать особенности реализации отдельных наиболее значимых операций в системе. Кооперация определяет структуру поведения системы в терминах взаимодействия участников этой кооперации.

Кооперация может быть представлена на двух уровнях:

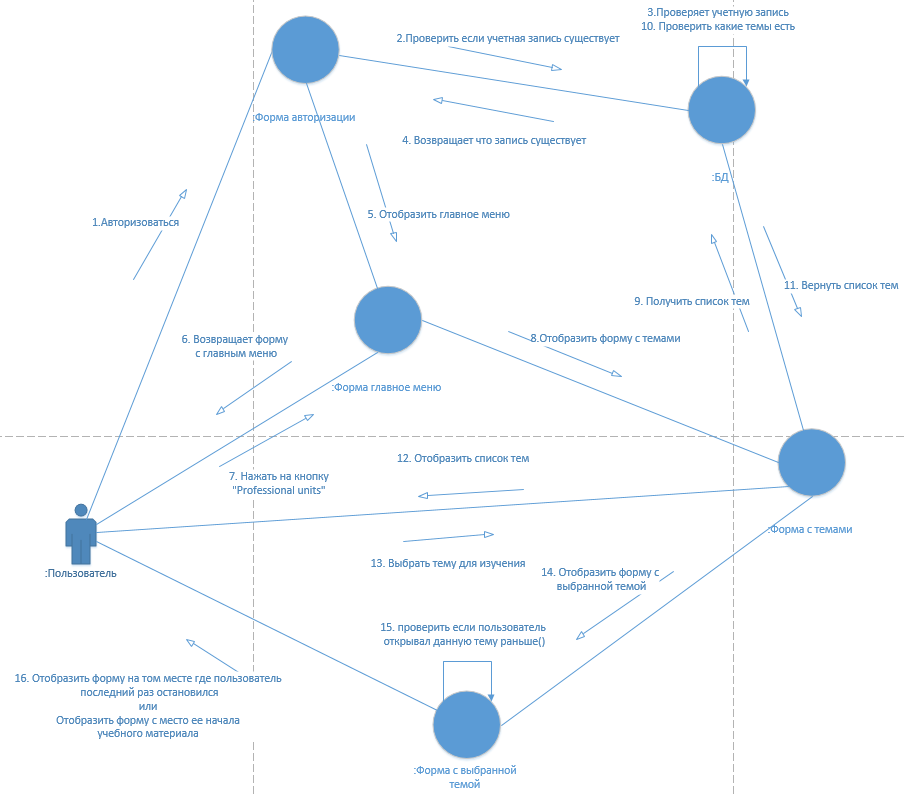
* На уровне спецификации - показывает роли классификаторов и роли ассоциаций в рассматриваемом взаимодействии.
* На уровне примеров - указывает экземпляры и связи, образующие отдельные роли в кооперации.

Диаграмма кооперации уровня спецификации показывает роли, которые играют участвующие во взаимодействии элементы. Элементами кооперации на этом уровне являются классы и ассоциации, которые обозначают отдельные роли классификаторов и ассоциации между участниками кооперации.

Диаграмма кооперации уровня примеров представляется совокупностью объектов (экземпляры классов) и связей (экземпляры ассоциаций). При этом связи дополняются стрелками сообщений. На данном уровне показываются только релевантные объекты, т. е. имеющие непосредственное отношение к реализации операции или классификатора.

В кооперации уровня примеров определяются свойства, которые должны иметь экземпляры для того, чтобы участвовать в кооперации. Кроме свойств объектов на диаграмме кооперации также указываются ассоциации, которые должны иметь место между объектами кооперации. При этом вовсе не обязательно изображать все свойства или все ассоциации, поскольку на диаграмме кооперации присутствуют только роли классификаторов, но не сами классификаторы. Таким образом, в то время как классификатор требует полного описания всех своих экземпляров, роль классификатора требует описания только тех свойств и ассоциаций, которые необходимы для участия в отдельной кооперации.

Отсюда вытекает важное следствие. Одна и та же совокупность объектов может участвовать в различных кооперациях. При этом, в зависимости от рассматриваемой кооперации, могут изменяться как свойства отдельных объектов, так и связи между ними. Именно это отличает диаграмму кооперации от диаграммы классов, на которой должны быть указаны все свойства и ассоциации между элементами диаграммы.



**Рисунок 7 – Диаграмма кооперации по прецеденту**

**«Выбрать тему для изучения»**

Объекты диаграммы: пользователь, форма авторизации, БД, форма главное меню, форма с темами и форма с выбранной темой. Объект «:Пользователь» инициирует диаграмму, он обращается к объекту «Форма авторизации» с сообщением «Авторизоваться». После чего объект «Форма авторизации» отправляет сообщение объекту «БД» чтобы база данных проверила если данные, введенные пользователем, существуют, база данных делает рефлексивное сообщение проверяет существование учетной записи и возвращает сообщение форме авторизации что запись существует. Объект «Форма авторизация» отправляет сообщение объекту «Форма главное меню» чтобы та отобразила форму главного меню, что оно и возвращает объекту «Пользователь». Далее пользователь отправляет сообщение форме с главным меню что он нажимает на кнопку «Professional units» (учебные материалы), после чего форма главного меню отправляет сообщение форме с темами чтобы он отобразил форму с темами. Форма с темами отправляет сообщение базе данных чтобы он отобразил список тем, он проверяет какие темы есть и отправляет их форме с темами после чего этот объект отображает пользователю. Пользователь выбирает тему для изучения на форме с темами после чего форма с темами открывает форму с выбранной темой, которая проверяет на каком месте пользователь остановился и отображает пользователю (Рисунок 7).

## 2.2 Проектирование информационной базы данных

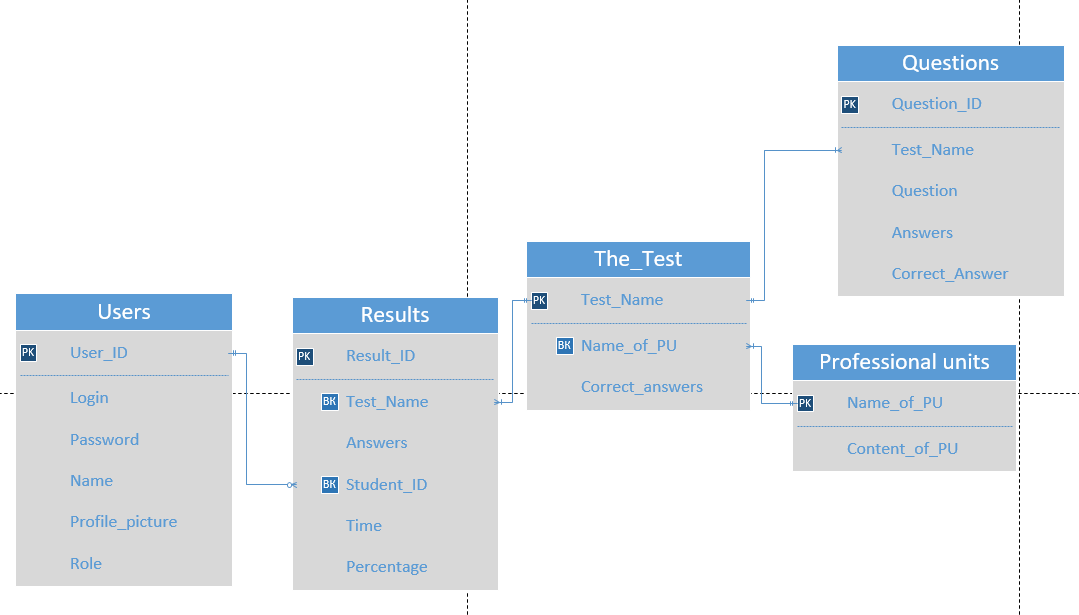
Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса. ER-диаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи — глаголов.

ER-диаграммы — «родственники» схем структуры данных (DSD), где вместо связей между самими сущностями отображаются отношения между элементами внутри них. ER-диаграммы часто используются в сочетании с диаграммами DFD, которые схематично показывают движение потоков информации в рамках процесса или системы.

**ER диаграмма системы**

В данной ER диаграмме системы имеется 6 сущностей:

1. Users (Пользователи)
2. Results (Результаты)
3. The\_Test (Тесты)
4. Questions (Вопросы)
5. Professional units (Учебный материал)



**Рисунок 8 – ER-диаграмма**

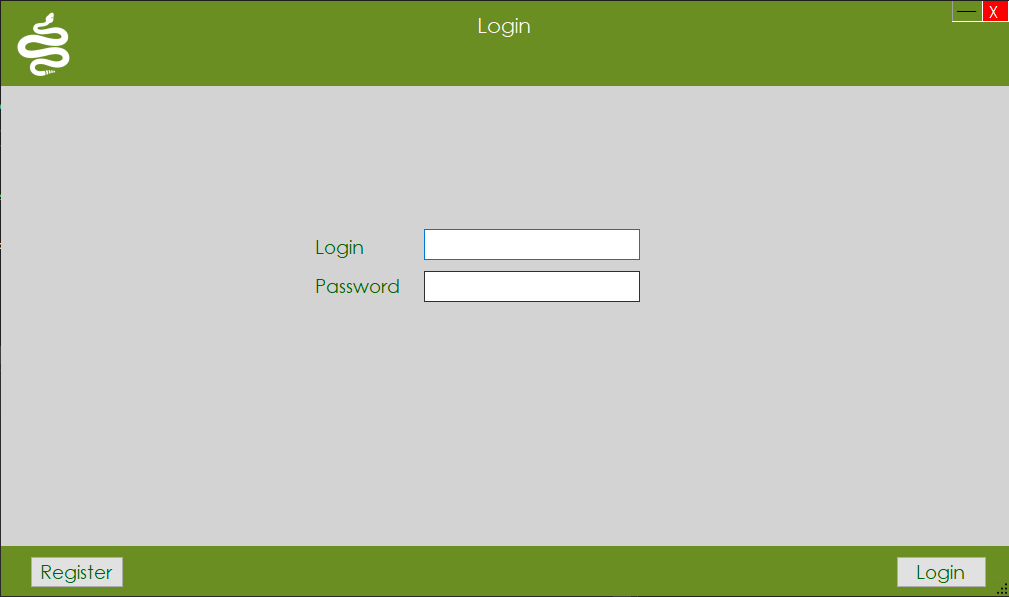
У каждой сущности существует первичный ключ с помощью, которого он связывается с другими сущностями, у которых создается внешний ключ. Связь, которая используется для связывания это – один ко многим. Символ начала у первичного ключа является 1 и только один, а конечный символ у внешнего ключа – 1 или более (Рисунок 8).

## 2.3 Моделирование пользовательского интерфейса

**Макет формы авторизации**

Верхняя область состоит из кнопок управления формы и надписи с названием формы. В центральном блоке расположены текстовые поля для ввода логина и пароля и рядом расположены подписи полей, определяющие что нужно вводить. В нижней области размещена кнопка “Login” (при нажатии которой пользователь попадает на следующую форму если входные данные совпадают с данными из базы данных системы) и кнопка “Register” (регистрации), которая перенесет пользователя на форму регистрации (Рисунок 9).

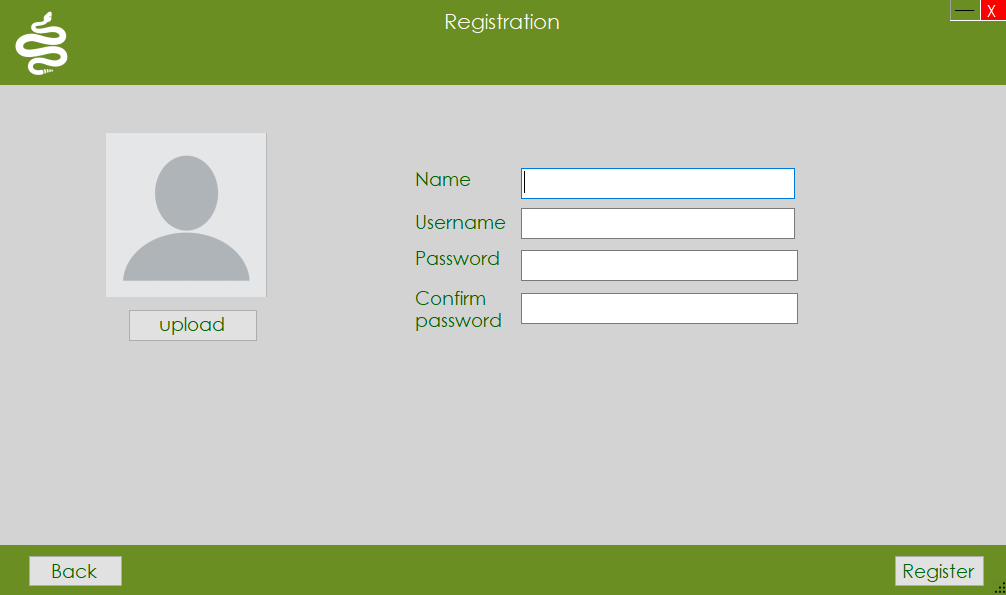
**Рисунок 9 – Макет формы «Login»**



**Макет формы регистрации**

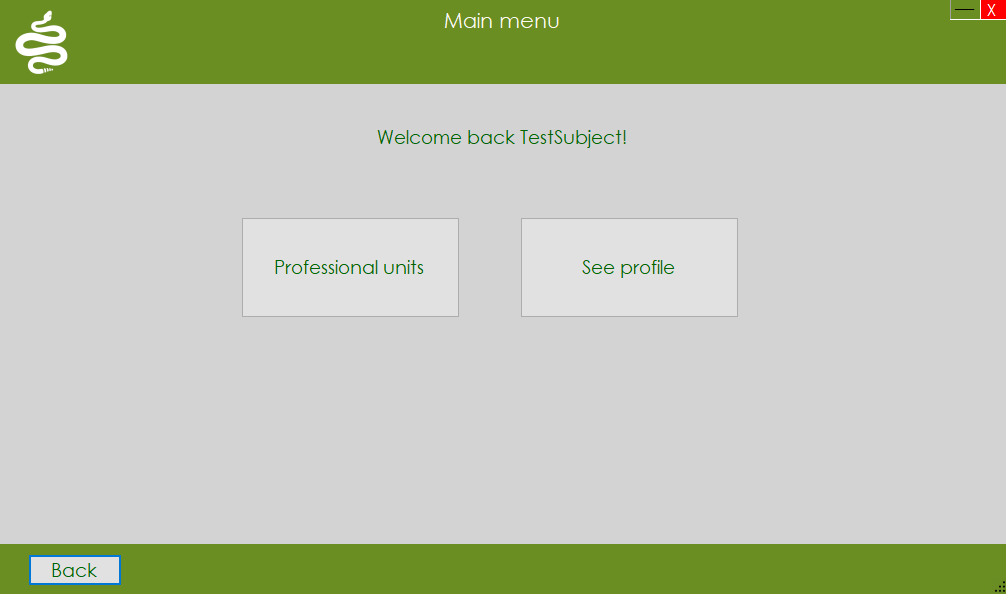
Верхняя область состоит из кнопок управления формы и надписи с названием формы. В центральном блоке расположены текстовые поля для ввода данных пользователя необходимые для регистрации и подписи полей, определяющие, что нужно вводить. Так же на левой стороне размещена фотография, которую он может загрузить с компьютера при нажатии на кнопку “Upload”. На нижнем блоке расположена кнопка назад (закрывает данную форму, и пользователь переходит на форму авторизации) и кнопка регистрация (заносит данные в базу данных и переносит пользователя на форму главное меню) (Рисунок 10).

**Рисунок 10 – Макет формы «Registration»**



**Макет формы главное меню**

Верхняя область состоит из кнопок управления формы и надписи с названием формы. В центральном блоке расположены две кнопки: левая кнопка (professional units) переносит пользователя на форму с темами, правая кнопка (typing exercise) переносит пользователя на форму с упражнениями связанные с печатаньем на клавиатуре. В нижнем блоке расположена кнопка “Back”, которая переносит пользователя на предыдущую форму (Рисунок 11).

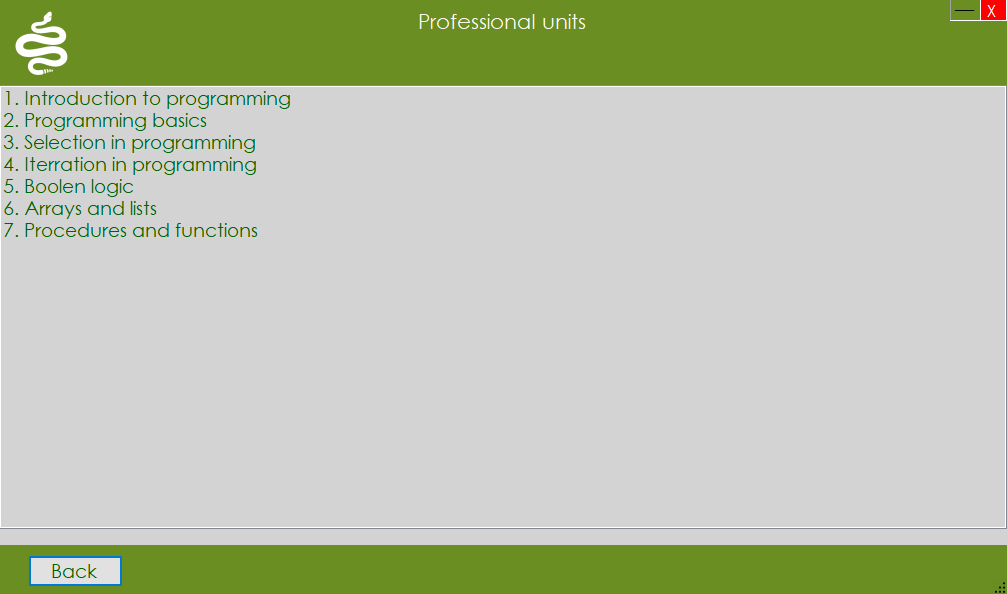


**Рисунок 11 – Макет формы «Main menu»**

**Макет формы список учебного материал (Professional units)**

Верхняя область состоит из кнопок управления формы и надписи с названием формы. В центральном блоке расположен список всех “Professional units” (учебных материалов) при нажатии на учебный материал появляются две кнопки над списком на правой стороне это кнопка “Write a test”, а на левой стороне кнопка “Open PU”. Кнопка “Open PU” открывает форму с учебным материалом, а кнопка “Write a test” открывает форму со списком тестов. В нижнем блоке расположена кнопка “Back” и “Menu” (Рисунк 12).

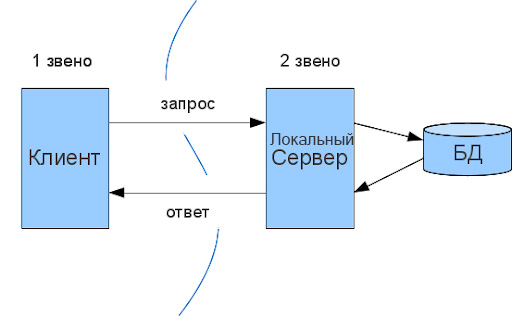
**Рисунок 12 – Макет формы «Professional units»**



# 3. описание реализации программного обеспечения информационной системы

## 3.1 Описание архитектуры программного обеспечения

Программное обеспечение разрабатываемой информационной системы было реализовано в виде клиент-серверного приложения, схема которого показана на рисунке 13.



**Рисунок13– Клиент-серверная архитектура**

На компьютере каждого пользователя должна быть установлена, разработанная в данной дипломной работе, клиентская программа «Rattlesnake». Данная программа взаимодействует с сервером, установленным в локальной сети, который работает с базой данных «RattleSnakeDB».

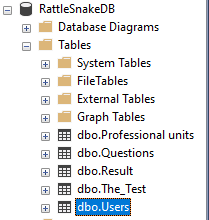
## 3.2 Создание базы данных

В качестве системы управления базами данных используется Microsoft SQL Server. MS SQL – это система управления реляционными базами данных (РСУБД). Данная система управления базами данных позволяет хранить большие объемы данных и выполнять различные манипуляции с ними (сохранение, выборка, изменение и удаление и т.п.). Она может использоваться для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

Для выполнения работы с MS SQL используется специальная программа SQL Server Management Studio (SSMS), которая позволяет с помощью графического интерфейса удобно выполнять конфигурирование, управление и администрирование всеми компонентами MS SQL Server. SSMS включает редактор SQL скриптов и графическую программу, которая работает с объектами и настройками сервера. Главным инструментом SSMS является компонент Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью управлять ими.

На основе диаграммы Entity Relationship с помощью программы SSMS была создана база данных «RattleSnakeDB». В состав этой базы данных были включены 6 взаимосвязанных таблиц:

**Рисунок 14 – Скриншот из MS SQL базы данных.**



В таблицах базы данных были заданы названия столбцов, тип данных, который может храниться в них и возможность не задавать им значения (тип NULL):

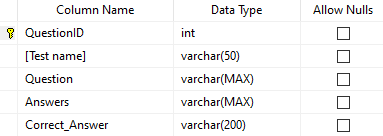
1. Таблица по учетными данными «Professional units»(Рисунок 15).

**Рисунок 15 – Таблица «Professional units»**



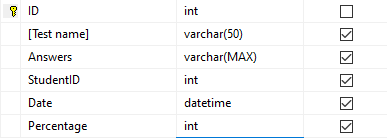
1. Таблица по учетными данными «Questions»(Рисунок 16).

**Рисунок 16 – Таблица «Questions»**

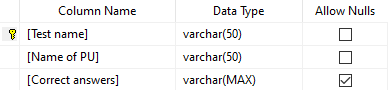


1. Таблица по учетными данными «Result»(Рисунок 17).

**Рисунок 17 – Таблица «Result»**

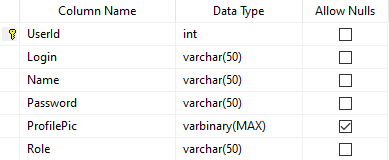


1. Таблица по учетными данными «The\_Test»(Рисунок 18).



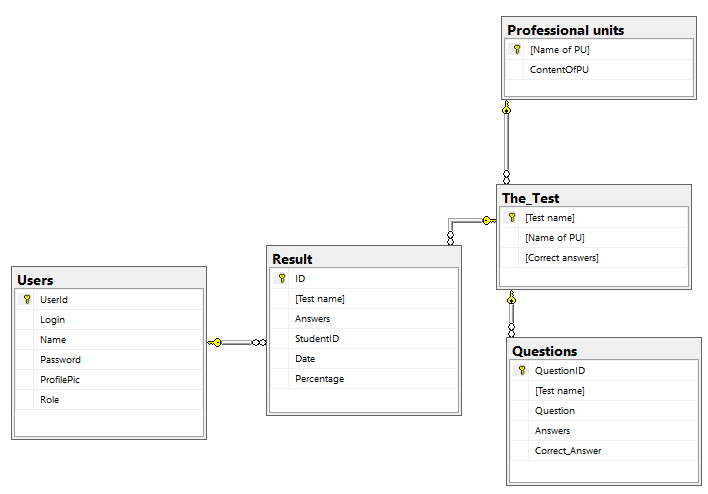
**Рисунок 18– Таблица «The\_Test»**

1. Таблица по учетными данными «Users»(Рисунок 19).



**Рисунок 19 – Таблица «Users»**

C помощью диаграммы базы данных между таблицами были заданы взаимосвязи, которые показаны на рисунке 20.



**Рисунок 20 – Диаграмма связей базы данных**

## 3.3 Разработка программного кода

Клиентская часть программного обеспечения была разработана на языке C# с использованием среды разработки Microsoft Visual Studio 2019. Программа имеет графический интерфейс, разработанный с использованием технологии Windows Forms, и взаимодействует с MS SQL сервером с использованием технологии Entity Frameworks ADO.NET Code first from database.

В программе разработано Номер форм.

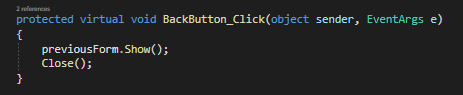
**Форма «basic Form»**

Это форма является базовой формой, которая наследуется во всех остальных формах. У нее стоят все свойства, которые наследуется в других формах, это цвет текста, цвет фона, цвет фона панели и шрифт текста.

**Рисунок 21 – Базовая форма «basicForm»**

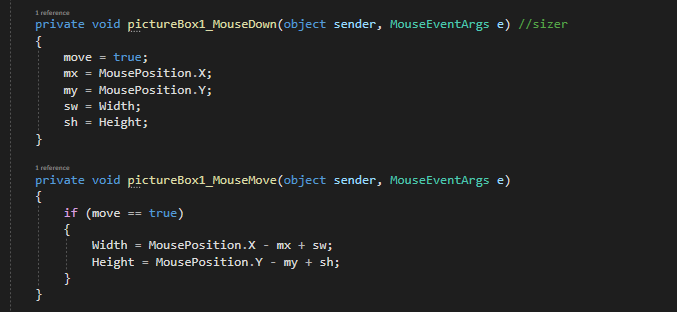


PictureBox в левом верхнем углу это иконка приложения, при нажатии на которую переходит в меню в зависимости от роли пользователя. Так же есть lable1 текст, которого вручную заполняется на других формах, но расположение и цвет текста, и шрифт переноситься в другие формы. В нижнем левом углу есть кнопка назад, которая закрывает данную форму и открывает предыдущую. Событие представлено на рисунке 22.



**Рисунок 22 – События кнопки назад «BackButton»**

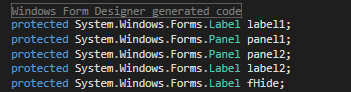
Так же в правом нижнем углу есть PictureBox изменение размера формы (sizer), при нажатии на него задается переменным sw (Ширина формы после нажатие на sizer), sh (Высота формы после нажатие на sizer), mx (X координат формы после нажатие на sizer), my (Y координат формы после нажатие на sizer) и move (Переменная проверяющая если движется мышка). Потом при движении мышки срабатывает событие pictureBox1\_MouseMove, где проверяется если переменная move равняется true то задается ширине формы (Width) позиция мышки в данном моменте из которого вычитается значение переменной mx и прибавляется значение sw. Это заставляет форму расширятся. Код событий представлен на рисунке 23.



**Рисунок 23 – События pictureBox1\_MouseDown и pictureBox1\_MouseMove**

Для того, чтобы производные формы, созданные на основе данной базовой формы, могли изменять свойства ЭУ (например, задавать свойство Text для текстовых меток или делать кнопки управлением формы) у этих элементов нужно изменить режим доступа с private (закрытый – доступный только в базовом классе) на protected (защищенный – доступный и в базовом и в производных классах).

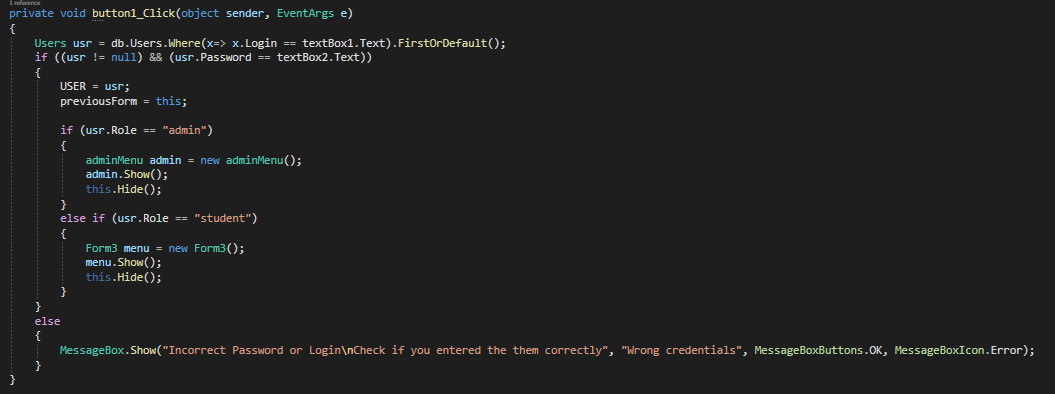
И в нижней части файла у всех ЭУ заменим режимы доступа с private на режимы доступа protected (Рисунок 24):



**Рисунок 24– Designer базовой формы**

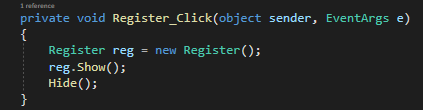
**Форма «Login»**

Данная форма представляет из себя форму авторизации в систему. На ней расположены текстовые поля для ввода логина и пароля и рядом расположены подписи полей, определяющие что нужно вводить. Так же в нижнем правом углу существует кнопка «Login», при нажатии которой совершается событие button1\_Click(), в котором создается локальный класс User и ему присваивается пользователя которой потом ищется в базе данных с помощью модели Entity framework используя событие Where в скобках, которых я задаю значение textBox1.Text(текст поля заполнения логина), потому что это ключевое поле таблицы. Дальше программа проверяет если этот пользователь существует в базе данных и, если текст в textBox2 (поля для ввода пароля) совпадает с паролем из базы данных.Если проверка проходит, то переменной previousForm, которая является типом Form присваивается значение формы Login и переменной USER (тип данных User). Дальше идет проверка на если роль найденного пользователя равняется роль админа, то открывается форма меню админа и скрывается данная форма, а если роль равняется роли студента, то открывается форма меню студента и скрывается данная форма. Код формы можно увидеть на приведенном ниже рисунке (Рисунок 25).



**Рисунок 25– Designer базовой формы**

В форме логина еще есть кнопка регистрации, которая открывает форму регистрации (Рисунок 26).



**Рисунок 26– Событие кнопки «Register»**

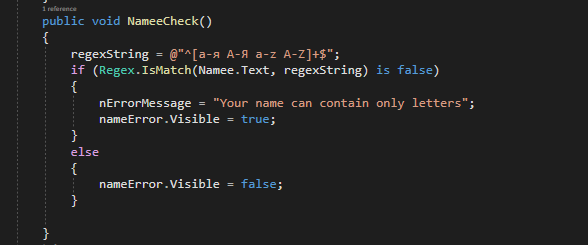
**Форма «Register»**

В форме регистрации есть поля для заполнения имени, логина, пароля и проверка на совпадение паролей.

Для каждого из полей существуют проверки на правильность заполнения события проверки правильность ввода данных с помощью Regex. Используя namespace System.Text.RegularExpressions.

Создается строковая переменная regexstring, в которой записываются какие символы проверять надо. Далее проверяется условие, в котором используется библиотека Regex и событие IsMatch в ней, в скобках которой прописывается какой текст нужно проверить и переменная regexString (символы которых мы проверяем наличие). Если мы не находим проверяемые символы, то в nameError (label, в котором есть индикация на то, что ошибка в заполнении текстового поля есть) меняется свойство Visible на false, значит, что этого label не будет на экране, означающие то, что ошибок при заполнении полей нет. А если обнаруживается символы, то в свойстве Visible nameError задается значение true это значит, что он появляется на форме и при наведении на него мышки будет выдаваться в чем ошибка заполнения, которая присваивается toolTip1 (Рисунок 27).

**Форма «Menu»**

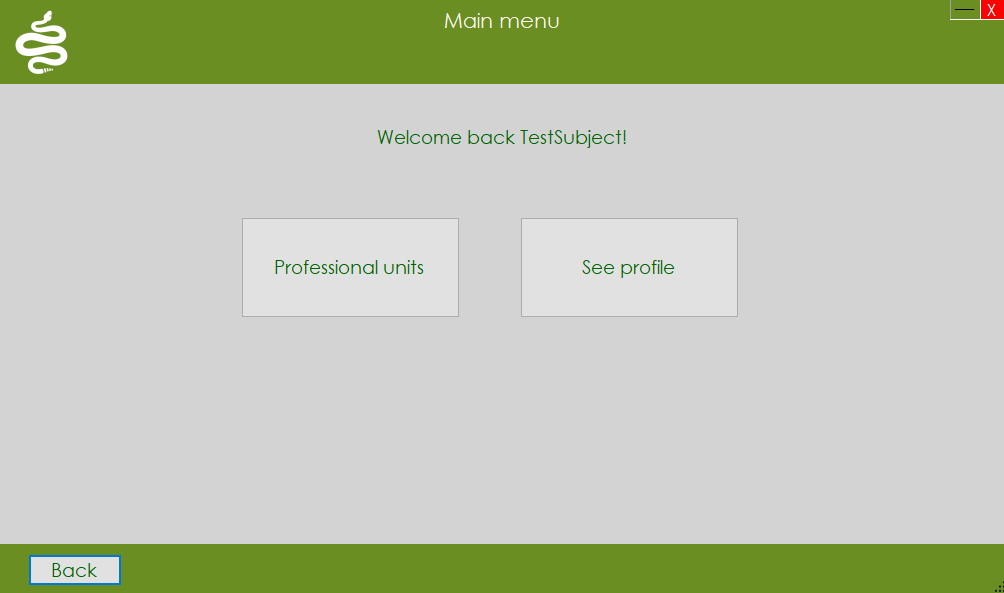


**Рисунок 27 – Одна из проверок: событие проверки имени**

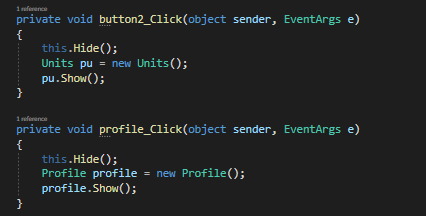
Данная форма — это главное меню для пользователей с ролью «student».

На ней есть кнопки: «Professional units» и «See profile». Кнопка «Professional units» скрывает форму «Menu» и открывает форму «Units» там, где находится учебный материал для изучения, а кнопка «See profile» открывает форму «Profile» где можно посмотреть результаты пройденных тестов и можно посмотреть информацию пользователя и если нужно, то поменять её.

Так же если пользователь с ролью «student» только что авторизовался в систему, это первая форма, которую он видит и появляется надпись, которая приветствует его (Рисунок 28).



**Рисунок 28 – Форма «Menu»(меню студента)**

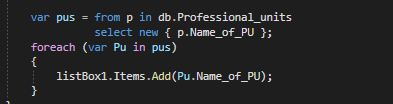


**Рисунок 29 – Код событий кнопок «Professional units» и «See profile»**

События кнопок «Professional units» и «See profile» представлены на рисунке 29.

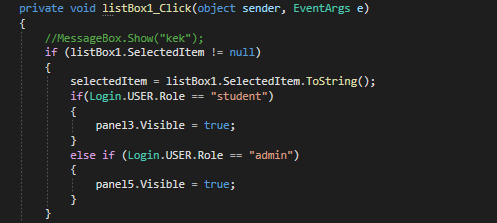
**Форма «Units»**

Форма «Units» предоставляет пользователю список всех существующих учебных материалов, которые берутся из базы данных и добавляется в lisBox, добавленный в дизайнере. Для нахождения всех названий учебных материалов используется LinQ запрос (Рисунок 30).



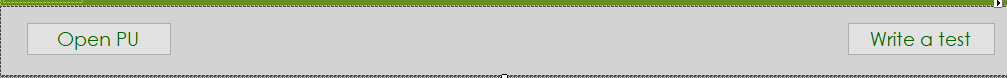
**Рисунок 30 – Запрос на выгрузку из базы данных названий учебных материалов**

Эта форма используется для пользователей с ролью «student» и «admin», только отображаются разные вещи при нажатии один раз на учебный материал. Используется оператор условий if, который проверяет роль пользователя и отображает ему панель с доступными ему функциями (Рисунок 31).



**Рисунок 31 – Код событий при выборе учебного материала**

Панель panel3 для студентов содержит в себе кнопки «Open PU», который открывает выбранный учебный материал и кнопка «Write a test», которая открывает форму с тестом по выбранному материалу (Рисунок 32).



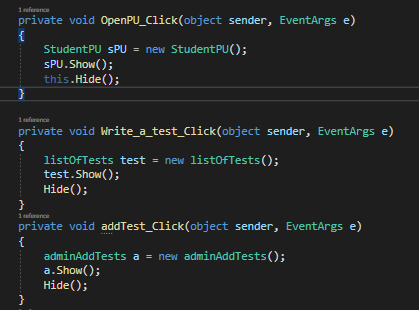
**Рисунок 32 – Панель panel3(для студентов)**

Панель panel5 для администратора содержит в себе кнопку «Add a Test», которая открывает форму «adminAddTests» (Рисунок 33).



**Рисунок 33 – Панель panel5(для администратора)**

События кнопок «Open PU», «Write a test» и «Add a Test» представлены на рисунке 34.



**Рисунок 34 – События кнопок «Open PU», «Write a test» и «Add a Test»**

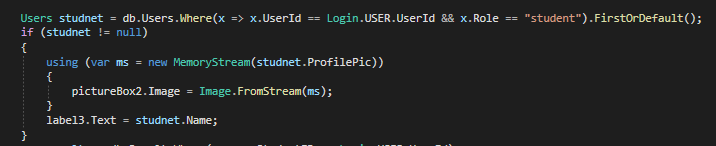
**Форма «Profile»**

Форма «Profile» отображает профиль студента. Эта форма есть только у пользователей с ролью «student». Там отображается фотография, username и кнопка редактирования информации о пользователе. Также отображается результаты всех тестов, которые он когда-либо проходил и результат прохождения их (Рисунок 34).

**Рисунок 34 – Форма профиля**



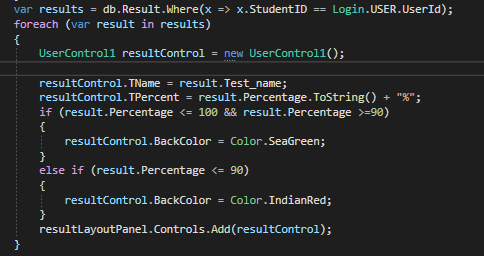
Результаты тестов загружаются в событие Profile\_Load(). Сначала как и на каждой форме задается переменной типа Form previousForm предыдущую форму. Далее ищется студент который авторизованный в данный момент с помощью функции Where(), которое возвращает первое или единственное значение. После этого происходит проверка если такой пользователь существует, используется простой логический оператор if (Рисунок 35).



**Рисунок 35 – Скиншот кода с проверкой на пользователя**

Далее чтобы конвертировать фотографию, которая сохранена в БД как байты, используется MemeryStream(), который создается в конструкторе using. После чего задается свойству Image объекта pictureBox, строка байтов, конвертированная под тип Image, с помощью функции Image.FromStream().

Чтобы отобразить все результаты нужно было сначала сделать поиск с помощью функции Where() в таблице«Results», которая находится в БД. Далее для каждого возвращенного результата создается UserControl1, в котором заполняются нужные поля уже созданные в UserControl1 (Рисунок 36).



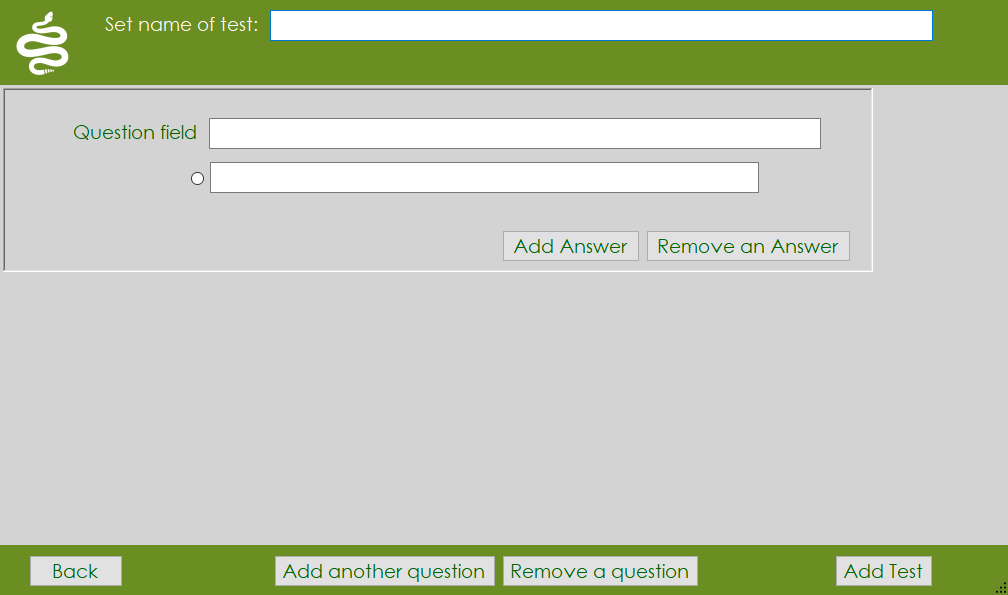
**Рисунок 36 – Скиншот кода с созданием UserControl1**

Так же проверяется поле resultControl.TPercent на значение. Если значение больше или равно 90 и меньше или равно 100, то цвет заднего фона UserControl1 для данного результата задается зеленый цвет, что означает что студент сдать тест. А если нет, то цвет заднего фона становиться красным.

После этого всего созданный результат добавляется в объект LayoutPanel, который находится на форме с помощью функции resultLayoutPanel.Control.Add(result Control).

**Форма «adminAddTests»**

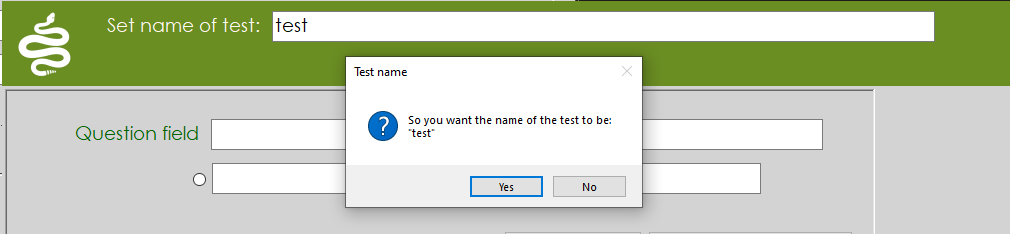
В данной форме пользователь с ролью «admin» может создавать и добавлять тесты для темы, которую он выбирает на предыдущей форме (Рисунок 37). Пользователь может добавлять вопросы, варианты ответов и указывать какой правильный при нажатии на radioButton. Также он может удалять вопросы и варианты ответов. После создания теста он может нажать на кнопку «Add Test», чтобы добавить тест в БД.



**Рисунок 37 – Форма «adminAddTests»**

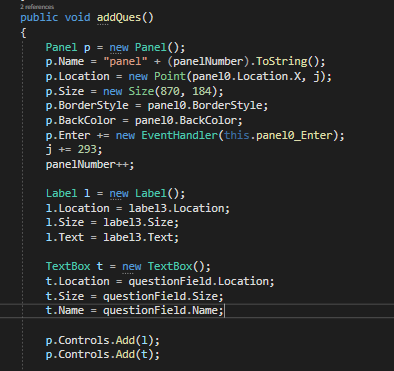
При входе на форму пользователь должен заполнить поле с названием теста, после чего он может приступить к созданию теста. При заполнении названия формы появляется сообщение с подтверждением что вы хотите оставить такое название для теста (Рисунок 37).

Для добавления нового вопроса используется функция addQues(). В этой функции создается панель со своими свойствами как Name, которому задается слово panel + panelNumber, значение которого изначально равняется 1. После создания панели происходит инкремент.



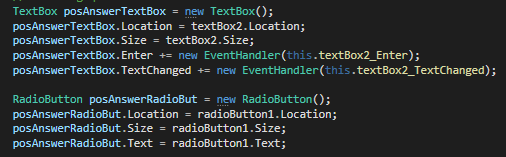
**Рисунок 37 – Скриншот сообщения подтверждения названия теста**

Далее создается поле для ввода данных (TextBox) и текст (Label) обозначающий что нужно заполнить в этом поле, в этом случаи заполнение вопроса. Эти элементы добавляются в Conrol панели, с помощью функции Conrol.Add() (Рисунок 38.1).



**Рисунок 38.1– Скиншот кода функции addQues()**

Далее создается текстовое поле для вопроса (TextBox) и RadioButton перед ним (Рисунок 38.2).



**Рисунок 38.2– Скиншот кода функции addQues()**

После этого создается нижняя панель с помощью которой можно добавлять дополнительные варианты ответов или удалять предыдущие варианты. Для этого создается панель в которую добавляются две кнопки: «Add answer» и «Remove an answer» (Рисунок 38.3).



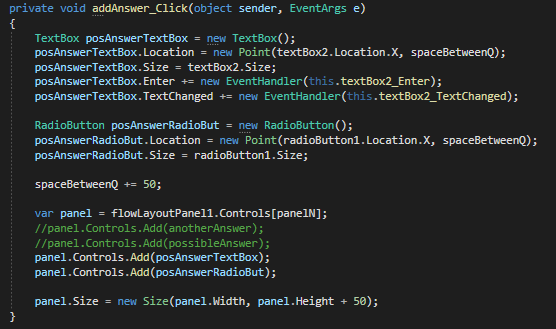
**Рисунок 38.3 – Скиншот кода функции addQues()**

Для них создается ссылка на функции: addAnswer\_Click() (Рисунок 39.1) и deletePossibleAnswer\_Click() (Рисунок 39.2).



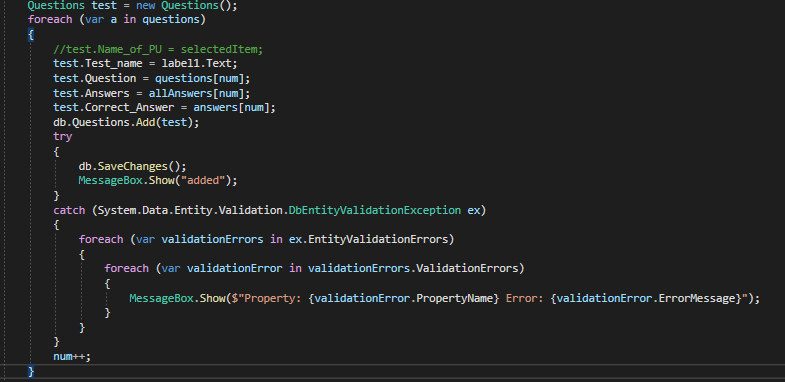
**Рисунок 39.2 – Скиншот кода функции deletePossibleAnswer\_Click()**

**Рисунок 39.1 – Скиншот кода функции addAnswer\_Click()**

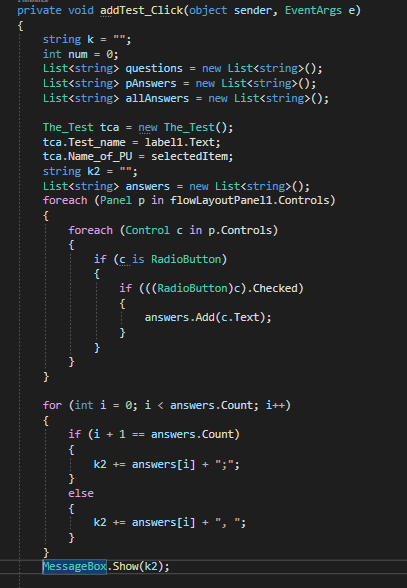


После создания теста пользователь может нажать на кнопку «Add test», функция которой является addTest\_Click, которые отображается на рисунке 40.

Добавление в базу данных происходит в конструкторе try catch с помощью функции db.SaveChanges(). Представлено на рисунке 41. До этого происходит присваивание данных, которые потом будут добавлены в БД. Создается новый класс из класса Questions, к полям которого присваиваются название тестов, все вопросы, ответы и правильный вариант ответа. После чего добавляется в базу данных пользуясь функции db.Questions.Add().



**Рисунок 41 – Скиншот кода функции добавление тетса в БД**



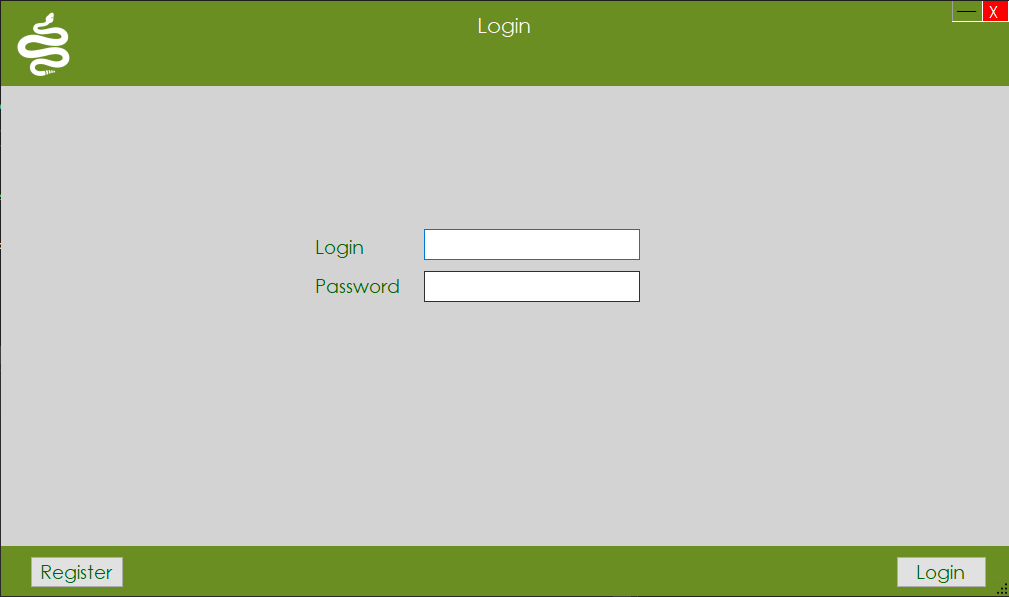
**Рисунок 40 – Скиншот кода функции addTest\_Click()**

## 3.4 Инструкции пользователей информационной системы

Пользователи системы могут быть:

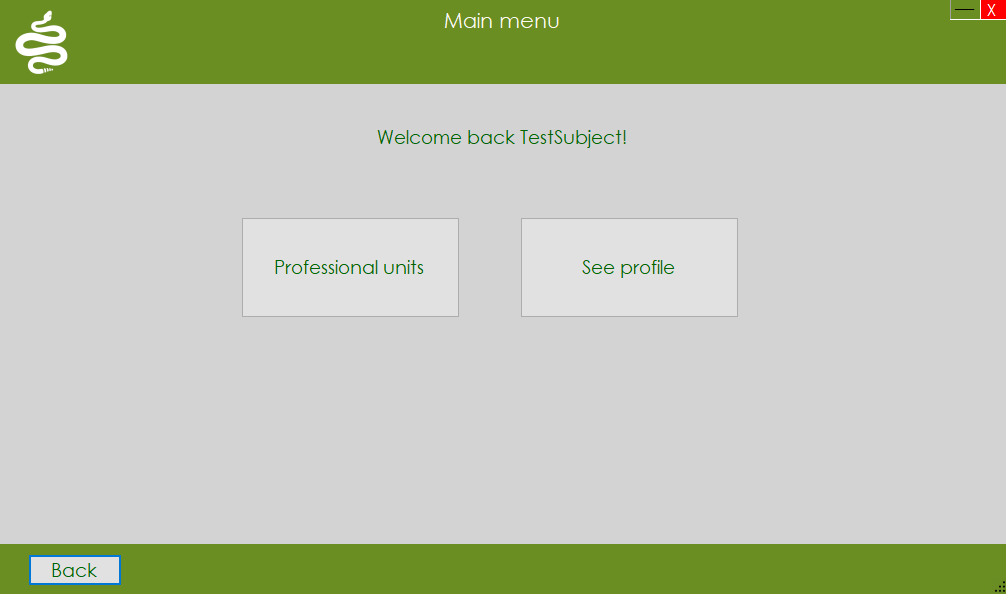
* Студент (student)
* Администор (admin)

Для работы с информационной системой пользователь должен быть в ней зарегистрирован (т.е. для него должна быть создана учетная запись в таблице Users). Пользователь (студент) может зарегистрироваться при нажатии на кнопку Register на форме логина, которая перенесет его на форму регистрации. При регистрации пользователю задается имя в системе (логин) и пароль (секретная строка, которую нужно будет вводить для начала работы в системе). Начальный пароль в дальнейшем пользователь может изменять. При наличии логина и пароля пользователь может выполнять работу с информационной системой «Rattlesnake». Для этого нужно запустить на выполнение программу Rattlesnake.exe на своем компьютере. При запуске программы появится начальное окно входа в систему:



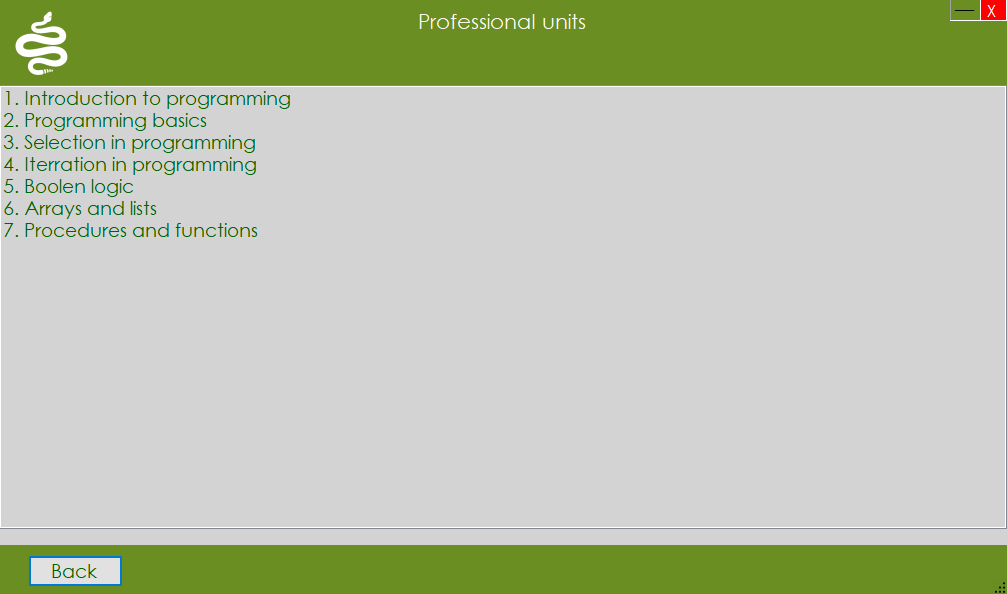
**Рисунок 42 – «Форма Login»**

Пользователь должен ввести свой логин и пароль, которые он заполнил в регистрации. При успешно входе в систему пользователю будет показана форма, которая соответствует его роли в системе. Например, если подключился студент, то будет показана следующая форма:



**Рисунок 43 – Макет формы «Main menu»**

При нажатии на кнопку «Professional units» система покажет все учебные материалы, которые доступны студентам. Например:



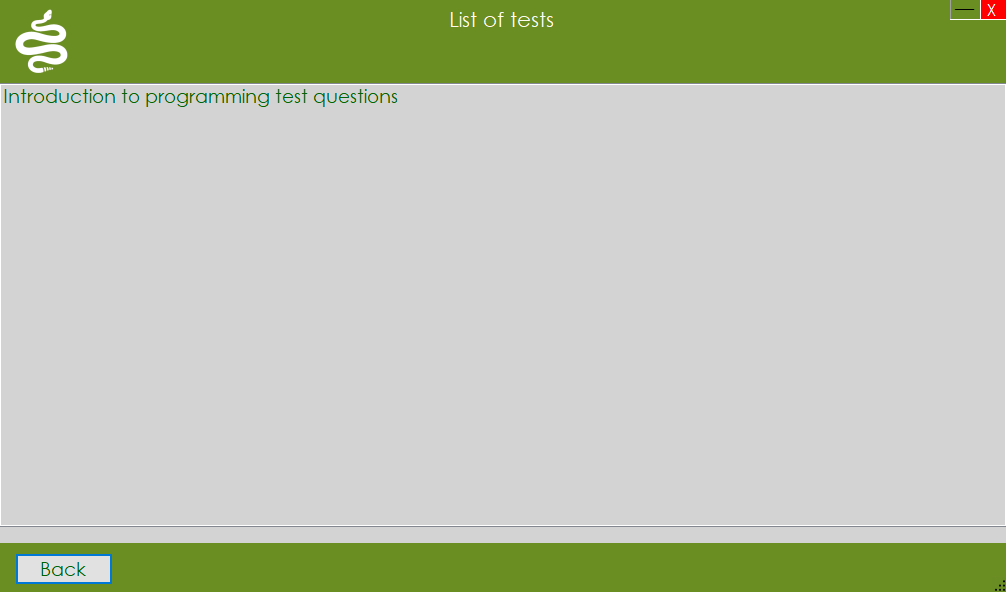
**Рисунок 44 – Форма «Professional units»**

После нажатие на любой учебный материал появляется панель где можно нажать на кнопку «Open professional unit» (открыть материал) или на кнопку «Write a test» (выполнить тест). Кнопка «Open professional unit» откроет форму с содержанием учебного материала. Например:



**Рисунок 45 – Форма с учебным материалом**

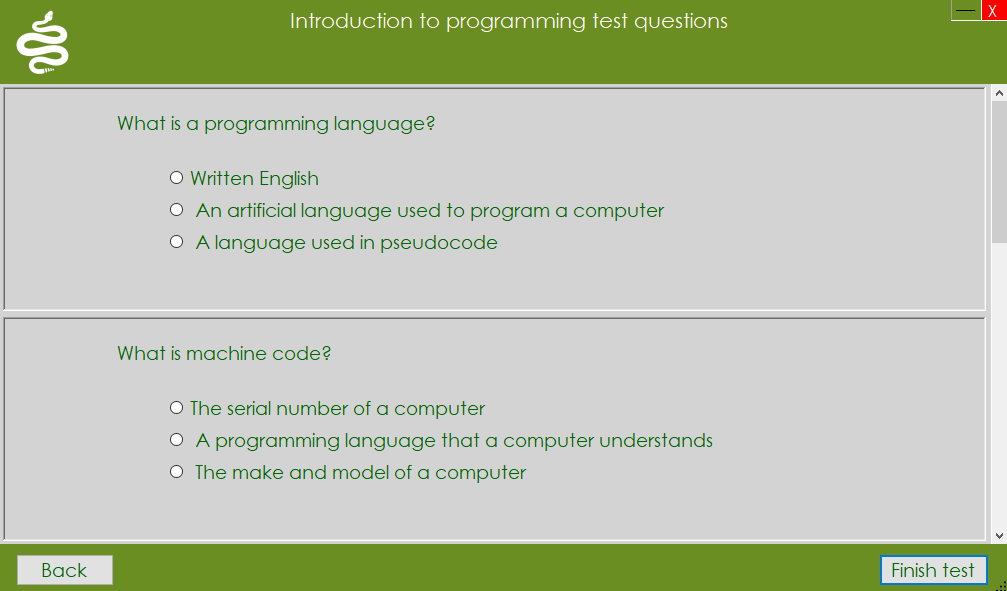
А если нажать на кнопку «Write a test», то открывается форма со списком тестов:



**Рисунок 46 – Форма «List of tests**

**s»**

На форме «List of tests» можно выбрать тест и два раза щеклнуть по нему, после чего откроется сам тест в новой форме:



**Рисунок 47– Форма теста**

**s»**

На этой форме можно выбирать правильный ответ при нажатии на radioButton, после того как студент закончил тест он нажимает копку «Finish test», которая заносит его ответы в базу данных и открывает главное меню пользователя («Main menu»).

# 4 Тестирование

## 4.1 Теория тестирования

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Качество программного продукта характеризуется набором свойств, определяющих, насколько продукт "хорош" с точки зрения заинтересованных сторон, таких как заказчик продукта, спонсор, конечный пользователь, разработчики и тестировщики продукта, инженеры поддержки, сотрудники отделов маркетинга, обучения и продаж. Каждый из участников может иметь различное представление о продукте и о том, насколько он хорош или плох, то есть о том, насколько высоко качество продукта. Таким образом, постановка задачи обеспечения качества продукта выливается в задачу определения заинтересованных лиц, их критериев качества и затем нахождения оптимального решения, удовлетворяющего этим критериям. Тестирование является одним из наиболее устоявшихся способов обеспечения качества разработки программного обеспечения и входит в набор эффективных средств современной системы обеспечения качества программного продукта.

С технической точки зрения тестирование заключается в выполнении приложения на некотором множестве исходных данных и сверке получаемых результатов с заранее известными (эталонными) с целью установить соответствие различных свойств и характеристик приложения заказанным свойствам. Как одна из основных фаз процесса разработки программного продукта (Дизайн приложения - Разработка кода - Тестирование), тестирование характеризуется достаточно большим вкладом в суммарную трудоемкость разработки продукта. Широко известна оценка распределения трудоемкости между фазами создания программного продукта: 40%-20%-40%, из чего следует, что наибольший эффект в снижении трудоемкости может быть получен прежде всего на фазах Design и Testing. Поэтому основные вложения в автоматизацию или генерацию кода следует осуществлять, прежде всего, на этих фазах. Хотя в современном индустриальном программировании автоматизация тестирования является широко распространенной практикой, в то же время технология верификации требований и спецификаций пока делает только свои первые шаги. Задачей ближайшего будущего является движение в сторону такого распределения трудоемкости (60%-20%-20%) чтобы суммарная цена обнаружения большинства дефектов стремилась к минимуму за счет обнаружения преимущественного числа на наиболее ранних фазах разработки программного продукта.

В своем проекте я использовал ручное тестирование. Ручное тестирование (manual testing) — часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно производится тестировщиком без использования программных средств, для проверки программы или сайта путём моделирования действий пользователя. В роли тестировщиков могут выступать и обычные пользователи, сообщая разработчикам о найденных ошибках.

## 4.2 Баг репорт

Таблица 1 – Тест 1. Форма «Регистрации»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 1** |
| Name | Андрей |
| Username | znatok |
| Password | pas123!L |
| Confirm password | pas123!L |
| pictureBox1 |  |
| Действие | Нажата кнопка "Register" |
| Ожидаемый результат | Данные занесены в поля таблицы "Users" |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 3 – Тест 3. Форма «Регистрации»

Таблица 2 – Тест 2. Форма «Регистрации»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 2** |
| Name |  |
| Username | znatok |
| Password | pas123!L |
| Confirm password | pas123!L |
| pictureBox1 |  |
| Действие | Нажата кнопка "Register" |
| Ожидаемый результат | Горит помогающие окно с надписью «The name field is unfilled». Появляется окно с надписью «Enter all the fields correctly!» |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 3** |
| Name | фва12 |
| Username | znatok |
| Password | pas123!L |
| Confirm password | pas123!L |
| pictureBox1 |  |
| Действие | Нажата кнопка "Register" |
| Ожидаемый результат | Горит помогающие окно с надписью «Your name can contain only letters». Появляется окно с надписью «Enter all the fields correctly!» |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 4 –Тест 4. Форма «Регистрации»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 4** |
| Name | фва12 |
| Username | znatok |
| Password | pas123!L |
| Confirm password | pas123!L |
| pictureBox1 |  |
| Действие | Нажата кнопка "Register" |
| Ожидаемый результат | Горит помогающие окно с надписью «Your name can contain only letters» |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 5 – Тест 5. Форма «Регистрации»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 5** |
| Name | Андрей |
| Username | андрей |
| Password | pas123!L |
| Confirm password | pas123!L |
| pictureBox1 |  |
| Действие | Нажата кнопка "Register" |
| Ожидаемый результат | Горит помогающие окно с надписью «Username cant contain Russian letters». Появляется окно с надписью «Enter all the fields correctly!» |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 6** |
| Name | Андрей |
| Username | Andrey123 |
| Password | pas123!L |
| Confirm password | pas123!L |
| pictureBox1 |  |
| Действие | Нажата кнопка "Register" |
| Ожидаемый результат | Данные занесены в поля таблицы "Users" |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 7 – Тест 7. Форма «Регистрации»

Таблица 6 – Тест 6. Форма «Регистрации»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 7** |
| Name | Андрей |
| Username | Andrey123 |
| Password |  |
| Confirm password |  |
| pictureBox1 |  |
| Действие | Нажата кнопка "Register" |
| Ожидаемый результат | Горит помогающие окно с надписью «The password field is unfilled». Появляется окно с надписью «Enter all the fields correctly!» |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 8** |
| Name | Андрей |
| Username | Andrey123 |
| Password | Pas123! |
| Confirm password | pas1123! |
| pictureBox1 |  |
| Действие | Нажата кнопка "Register" |
| Ожидаемый результат | Горит помогающие окно с надписью «Confirm password doesn’t match the Password field». Появляется окно с надписью «Enter all the fields correctly!» |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 8 – Тест 8. Форма «Регистрации»

Таблица 9 – Тест 9. Форма «Регистрации»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 9** |
| Name | Андрей |
| Username | Andrey123 |
| Password | Pas123! |
| Confirm password | Pas123! |
| pictureBox1 | Picture.png |
| Действие | Нажата кнопка «Upload», выбрана фотография с компьютера Нажата кнопка "Register". |
| Ожидаемый результат | Данные занесены в поля таблицы "Users" |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 10 – Тест 10. Форма «Регистрации»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 10** |
| Name | Андрей |
| Username | Andrey123 |
| Password | Pas123 |
| Confirm password | Pas123 |
| pictureBox1 | Picture.png |
| Действие | Нажата кнопка «Upload», выбрана фотография с компьютера Нажата кнопка "Register". |
| Ожидаемый результат | Горит помогающие окно с надписью «Password must contain at least one special character: !@#$%^&\*». Появляется окно с надписью «Enter all the fields correctly!» |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 11 – Тест 11. Форма «Добавление тестов»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 11** |
| Set test name | test |
| Question field | Where? |
| Possible answer | 1 |
| Possible answer | 2 |
| Действие | Нажата кнопка «Add test» |
| Ожидаемый результат | Появляется окно с надписью «This test name already exists» |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 13 – Тест 13. Форма «Добавление тестов»

Таблица 12 – Тест 12. Форма «Добавление тестов»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 12** |
| Set test name | Test2 |
| Question field | When? |
| Possible answer | 1 |
| Possible answer | 2 |
| Действие | Нажата кнопка «Add test» |
| Ожидаемый результат | Появляется помогающие окно с надписью «This question already exists» |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 13** |
| Действие | Нажата кнопка «Add answer» |
| Ожидаемый результат | Добавляется новый вариант ответа в данной панели вопроса. |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 14 – Тест 14. Форма «Добавление тестов»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 14** |
| Set test name | Test2 |
| Question field | Where? |
| Possible answer | 1 |
| Possible answer | 2 |
| Действие | Нажата кнопка «Remove an answer» |
| Ожидаемый результат | Удаляется последний созданный вариант ответа в данной панели вопроса. |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 15 – Тест 15. Форма «Добавление тестов»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 15** |
| Set test name | Test2 |
| Question field | Where? |
| Possible answer | 1 |
| Possible answer | 2 |
| Действие | Нажата кнопка «Add another question» |
| Ожидаемый результат | Добавляется новая панель макет заполнения вопроса. |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 16 – Тест 16. Форма «Добавление тестов»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 16** |
| Set test name | Test2 |
| Question field | Where? |
| Possible answer | 1 |
| Possible answer | 2 |
| Действие | Нажата кнопка «Add another question» |
| Ожидаемый результат | Удаляется последняя созданная панель макета заполнения вопроса. |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 17 – Тест 17. Форма «Добавления учебного материала»

Таблица 18 – Тест 18. Форма «Добавления учебного материала»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 17** |
| Действие | Нажата кнопка «Upload file» |
| Ожидаемый результат | Открывается окно проводника, где можно выбрать Word документ. |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 18** |
| Действие | Нажата кнопка «Upload file», выбран Word документ и нажата кнопка «Add PU». |
| Ожидаемый результат | В интерфейсе отображается учебный материал и данные добавляются в таблицу «Professional Units». |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 19 – Тест 19. Форма «Добавления учебного материала»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 19** |
| Действие | Нажата кнопка «Upload file», выбран Word документ, который уже добавлен был раньше и нажата кнопка «Add PU». |
| Ожидаемый результат | Появляется окно с надписью «This professional unit was already added». |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

Таблица 20 – Тест 20. Форма «Добавления учебного материала»

|  |  |
| --- | --- |
| **Значения полей формы:** | **Тест 20** |
| Действие | Нажата кнопка «Add PU». |
| Ожидаемый результат | Появляется окно с надписью «File not selected». |
|  |  |
| Результат тестирования | + |
| Описание результата |  |

# Заключение

В данной курсовой работе было проведено проектирование и разработка подсистемы обучения английскому языку, «Rattlesnake», целю которого является спроектировать и разработать подсистему обучения английскому языку, которое позволило бы облегчить изучение и практику английского языка в IT сфере.

При проведении анализа предметной области было проведено функциональное моделирование, включающее создание диаграмму IDEF0, с помощью которой были выявлены бизнес-процессов обучения английскому языку. Было проведено объектно-ориентированное моделирование, включающее построение основных диаграмм по нотации UML (диаграмма прецедентов, деятельности, последовательности и кооперации), а также построена ER-диаграмма по нотации Чена. Были сделаны выводы о необходимости создания подсистемы обучения английскому языку в IT сфере «Rattlesnake».

При создании макета интерфейса подсистемы учитывался проведенный анализ существующих подсистем в виде сайтов для определения необходимого функционала на формах и её дизайна.

Разработка подсистемы обучения английскому языку была проведена с помощью программного средства Visual Studio и SQL Server Management Studio (SSMS). А также сделан и представлен графический интерфейс по построенным макетам. Он был разработан с использованием технологии Windows Forms. Также разработка была проведена с взаимодействием с MS SQL сервером с использованием технологии Entity Framework ADO.Net «Code first from database». Руководство пользователя и описание разработки кода было расписано подробно, для понимания.

Тестирование разработанной подсистемы проводилось с целью установить соответствие различных свойств и характеристик приложения заданным свойствам. Оно было представлено таблицами баг-репортов без использования регрессионного и автоматического тестирования.

Все задачи курсового проекта выполнены. Цель достигнута.

# ссылки

1. ISO/IEC 12207:1995. (ГОСТ Р – 1999). ИТ. Процессы жизненного цикла программных средств.
2. ГОСТ Р 2.105-2019 Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам.
3. ИНТУИТ. Национальный открытый университет. Проектирование ИС. [Электронный ресурс] / http://www.intuit.ru/ - Электронные данные. – Режим доступа: http://www.intuit.ru/. Свободный. – Заглавие с экрана. – Яз. рус., анг.
4. ISO 9126:1991. (ГОСТ – 1993). ИТ. Оценка программного продукта. Характеристики качества и руководство по их применению.
5. ISO/IEC 14764: 1999. (ГОСТ Р – 2002). ИТ. Сопровождение программных средств.
6. ISO/IEC 15910:1999. (ГОСТ Р – 2002) ИТ. Пользовательская документация программных средств.
7. ГОСТ 34.602-89. ИТ. Техническое задание на создание автоматизированных систем
8. ГОСТ 34.201-89. ИТ. Виды, комплектность и обозначение документов при создании автоматизированных систем.
9. ISO 9001:2015. (ГОСТ Р – 2015). Система менеджмента качества. Требования.
10. РД 50-34.698-90. Методические указания. Информационная технология. Автоматизированные системы. Требования к содержанию документов
11. ИНТУИТ. Национальный открытый университет. Нотация и семантика языка UML. [Электронный ресурс] / http://www.intuit.ru/ - Электронные данные. – Режим доступа: http://www.intuit.ru/. Свободный. – Заглавие с экрана. – Яз. рус., англ.
12. ИНТУИТ. Национальный открытый университет. Программирование: Основы тестирования программного обеспечения [Электронный ресурс] / http://www.intuit.ru/ - Электронные данные. – Режим доступа: http://www.intuit.ru/. Свободный. – Заглавие с экрана. – Яз. рус., англ.
13. Албахари Джозеф, Албахари Бен C# 8.0 Карманный справочник: Скорая помощь для программистов на C# 8.0. – М.: Издательство Вильямс, 2020. – 240 с.;
14. Мюллер Джон Поль, Семпф Билл, Сфер Чак C# для чайников. – М.: Издательство Диалектика, 2019. – 608 с.;
15. Джеймс Р. Грофф Пол Н. Вайнберг Эндрю Дж. Оппель SQL Полное руководство 3-е издание. – М.: Издательство Вильямс, 2015. – 959 с.;
16. Хьюстон Келли А., Максимчук Роберт А. Объектно-ориентированный анализ и проектирование с примерами приложений – М.: Издательство Вильямс, 2017. – 720 с.;
17. Сэм Канер, Джек Фолк, Енг Кек Нгуен Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений – Издательство ДиаСофт, 2001. – 544 с.;
18. Кумсков М.И. Системный анализ. Предметная область. Модели на UML / М.И. Кумсков. – М.: Издательские решения, 2020. – 100 с.;
19. Фаулер Мартин UML. Основы. Краткое руководство по стандартному языку объектного моделирования – Издательство Символ-Плюс, 2018 – 192с.;
20. Исаев Георгий Николаевич Проектирование информационных систем. Учебное пособие – Издательство Омега-Л, 2015 – 424с.;
21. Елена Я. Аникина Entity Relationship Approach to Database Design – Издатель OmniScriptum Publishing KS, 2016 – 92 с.;
22. Фримен Адам Entity Framework Core 2 для ASP.NET Core MVC для профессионалов – Издательство Диалектика-Вильямс, 2019 – 624с.;