Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО КубГТУ)

Институт Компьютерных систем и информационной безопасности

Кафедра Информационных систем и программирования

Специальность 10.05.03 Информационная безопасность автоматизированных систем

Профиль Защищенные автоматизированные системы управления

**КУРСОВАЯ РАБОТА**

по дисциплине Технологии и методы программирования

(наименование дисциплины)

на тему: «WWW-конференция»

(тема курсовой работы)

Выполнил студент 2 курса группы 18-К-АС1

Пожигайло Роман Андреевич

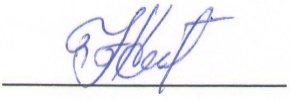
(Ф.И.О.)

Допущен к защите\_\_\_\_\_\_\_

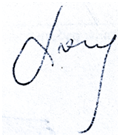
Руководитель (нормоконтролер) работы \_\_\_\_\_\_\_\_\_О.Б. Попова

Защищен \_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_

(дата)



Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_Н.В. Кушнир\_\_



К.Е. Тотухов

Краснодар

2020

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО КубГТУ)

Институт Компьютерных систем и информационной безопасности

Кафедра Информационных систем и программирования

Специальность 10.05.03 Информационная безопасность автоматизированных систем

Профиль Защищенные автоматизированные системы управления



УТВЕРЖДАЮ

Зав. кафедрой \_\_\_\_\_\_\_\_\_М.В. Янаева

«12» февраля 2020 г.

**ЗАДАНИЕ**

на курсовую работу

Студенту: Пожигайло Р. А. группы 18-К-АС1 курса 2

(Ф.И.О.) (№ группы и курса)

Тема проекта: «WWW-конференция»

План работы:

1. Изучение предметной области

2. Проектирование

3.  Описание реализованных диаграмм

Объем работы:

а) пояснительная записка 60 с.

Рекомендуемая литература

1.  Йордон. «Объектно-ориентированный анализ и проектирование систем»

2.  Роберт А. Максимчук. «UML для простых смертных»

3.  «Автоматизация проектирования вычислительных систем.»ред. М.Брейер

Срок выполнения: с «12» февраля по «28» марта 2020г.

Срок защиты: с «11» мая по «14» июня 2020 г.

Дата выдачи задания «12» февраля 2020г.

Дата сдачи работы на кафедру «01» июня 2020 г.

Руководитель работы \_\_\_\_\_\_\_\_\_ к.т.н., доцент Попова О.Б.

(должность, подпись,)

Задание принял студент Пожигайло Роман Андреевич Ф.И.О.

**Реферат**

Курсовая работа: 60 страниц, 25 рисунков, 10 используемых источников.

Ключевые слова : ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОЕКТИРОВАНИЕ, ПОСЛЕДОВАТЕЛЬНОСТИ, МОДЕЛЬ, КЛАСС, WWW-КОНФЕРЕНЦИЯ, UML ДИАГРАММЫ, IDEF0 ДИАГРАММА, BPMN ДИАГРАММА, DFD ДИАГРАММА, EPC ДИАГРАММА, ДИАГРАММА ГАНТА.

Объектом исследования является программное обеспечение WWW-конференции, которая представляет собой хранилище сообщений. Для каждого сообщения конференции хранятся значения следующих полей: номер сообщения, автор, тема, текст сообщения, дата добавления сообщения, ссылка на родительское сообщение. Пользователь может как добавить свою тему для обсуждения, так и присоединиться к обсуждению другой темы.

Цель работы состоит в разработке проекта программного обеспечения «WWW-конференция» с использованием UML диаграмм и диаграмм другого вида таких, как IDEF0, DFD, EPC, BPMN и других. С помощью них описано как внутреннее устройство исследуемой системы, так и всевозможные взаимодействия между ее компонентами.

В результате были получены UML диаграммы, обладающие исчерпывающей информацией о программном обеспечение WWW-конференции. К ним относятся: диаграмма вариантов использования, диаграмма классов, диаграмма последовательности. Также были получены такие диаграммы, как диаграмма Ганта, IDEF0, DFD, EPC, BPMN.

**Содержание**

[Введение 5](#_Toc40032528)

[1 Формулировка задачи 6](#_Toc40032529)

[2 Язык проектирования UML 6](#_Toc40032530)

[3 Проектирование диаграммы последовательности 8](#_Toc40032531)

[4 Проектирование диаграммы BPMN 11](#_Toc40032532)

[5 Проектирование диаграммы EPC 15](#_Toc40032533)

[6 Создание диаграммы IDEF0 16](#_Toc40032534)

[7 Диаграмма потоков данных (DFD) 20](#_Toc40032535)

[8 Проектирование диаграммы вариантов использования 23](#_Toc40032536)

[9 Проектирование диаграммы классов 24](#_Toc40032537)

[10 Проектирование диаграммы Ганта 26](#_Toc40032538)

[11 Требования к системе: классификация FURPS+ 27](#_Toc40032539)

[12 Тестирование программы 29](#_Toc40032540)

[13 Сопровождение ПО 34](#_Toc40032541)

[14 Система контроля версий ПО 35](#_Toc40032542)

[Заключение 36](#_Toc40032543)

[Список использованных источников 37](#_Toc40032544)

[Приложение А. Диаграмма IDEF0 38](#_Toc40032545)

[Приложение Б. Проверка на антиплагиат 41](#_Toc40032546)

[Приложение В. Код программы 42](#_Toc40032547)

# Введение

В настоящее время WWW-конференции нужны для обмена опытом, для знакомства с новыми прогрессивными технологиями и проектами, для общения с коллегами, для продвижения своих идей и просто для того, чтобы быть в курсе последних изменений из мира науки.

Практика проведения конференций показывает, что это довольно затратное мероприятие. Для того, чтобы стать участником какой-либо всероссийской или международной научной конференции, необходимо внести оргвзнос и потратить большое количесво средств на перелеты и переезды. Для какой-нибудь организации, особенно для бюджетных, такие расходы могут быть очень существенны.

С развитием интернета появились возможности не только для развлечений, но и для общения. Такой инструмент как WWW-конференции дают возможность собирать участников конференции из разных точек земного шара. Также нельзя не принять во внимание экономию времени и средств для организаторов и участников.

Несмотря на повсеместное использование данной технологии она не смогла бы обеспечить надлежащий уровень работы без тщательного исследования предметной области и проведения различных тестов, учитывающих всевозможные взаимодействия с компонентами системы. Именно для этих целей можно использовать проектирование на основе различного рода диаграмм таких, как UML, IDEF0, DFD, EPC, BPMN и других. Например, UML диаграммы позволяют описать любую систему в максимально удобном и понятном виде, даже для незнакомого с этой областью человека.

WWW-конференция созданная на основании всех проведенных исследований, будет соответствовать всем основным требованиям для обеспечения связи между участниками.

# Формулировка задачи

Задачей данного курсового проекта является разработка программного обеспечения WWW-конференции, которая представляет собой хранилище сообщений. Для каждого сообщения конференции хранятся значения следующих полей: номер сообщения, автор, тема, текст сообщения, дата добавления сообщения, ссылка на родительское сообщение. Начальной страницей конференции является иерархический список сообщений. Верхний уровень иерархии составляют сообщения, открывающие новые темы, а подуровни составляют сообщения, полученные в ответ на сообщения верхнего уровня. Сообщение-ответ всегда имеет ссылку на исходное сообщение. В списке отображаются только темы сообщений, их авторы и даты добавления. Просматривая список, пользователь выбирает сообщение и открывает страницу с текстом сообщения.

Сообщения добавляются в конференцию зарегистрированными пользователями, которые при отправке сообщения должны указать своѐ имя и пароль. Регистрирует новых пользователей модератор конференции - еѐ ведущий. При регистрации пользователь заполняет специальную форму, содержимое которой затем пересылается модератору и запоминается в базе пользователей.

При добавлении сообщений пользователь имеет возможность начать новую тему или ответить на ранее добавленные сообщения. После добавления сообщения оно доступно для чтения всем пользователям, и список сообщений обновляется. Модератор имеет право по тем или иным причинам удалять сообщения любых авторов. Он также может наказывать пользователей, нарушающих правила поведения в конференции, лишая на некоторое время пользователя возможности добавлять и редактировать сообщения.

# Язык проектирования UML

Многие программисты, которые сталкиваются со сложной задачей, пренебрегают этапом проектирования и говорят о том, что это потеря времени. Зачастую это утверждение оказывается верным, если задача небольшая. Однако при решении более сложных задач планирование и моделирование упрощает программирование.

UML — это унифицированный графический язык моделирования для описания, визуализации, проектирования и документирования объектно-ориентированных систем. Он состоит из словаря и правил, которые позволяют получать осмысленные конструкции. Этот язык является стандартным средством для составления схем программного обеспечения.

Такое моделирование необходимо для понимания системы. Однако единственной модели не всегда бывает достаточно. Для описания какой-либо сложной системы необходимо разрабатывать большое количество взаимосвязанных моделей. UML позволяет с разных точек зрения описать какую-либо программную систему на протяжении цикла ее разработки.

В список основных UMLдиаграмм входят:

* диаграмма вариантов использования;
* диаграмма классов;
* диаграмма последовательности;
* диаграмма физического аспекта системы;
* диаграмма коммуникации;
* диаграмма состояний;
* диаграмма компонентов.

Предпосылки для создания языка UML появились в связи в активным развитием во второй половине XX века объектно-ориентированных языков программирования таких, как [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk), [Objective C](https://ru.wikipedia.org/wiki/Objective_C" \o "Objective C), [C++](https://ru.wikipedia.org/wiki/C%2B%2B) и другие. В связи с усложнением создаваемых программ возникла потребность в новых возможностях языков и средств разработки при их проектировании.

Отдельные языки объектно-ориентированного моделирования начали появляться в середине 1970-х годов, когда различные исследователи и программисты предлагали свои подходы к ООАП. Многие пользователи испытывали серьезные затруднения при выборе языка объектно-ориентированного анализа и проектирования, поскольку ни один из них не удовлетворял всем требованиям, предъявляемым к построению моделей сложных систем

К середине 1990-х некоторые методы были существенно улучшены и приобрели самостоятельное значение при решении различных задач ООАП. Наиболее известными в этот период становятся:

* Метод Гради Буча (Grady Booch), получивший условное название Booch или Booch'91, Booch Lite (позже - Booch'93)
* Метод Джеймса Румбаха (James Rumbaugh), наименованный Object Modeling Technique - OMT (позже - OMT-2)
* Метод Айвара Джекобсона (Ivar Jacobson), под названием Object-Oriented Software Engineering - OOSE

UML появился в результате слияния нескольких объектное ориентированных языков моделирования, зародившихся в начале 90-х.

Создатели, каждого из вышеперечисленных языков моделирования, решили объединиться и разработать новый унифицированный язык, который избавится от всех ошибок своих предшественников и произведёт настоящую эволюцию в мире объектно-ориентированных языков проектирования.

В результате появился самый мощный унифицированный язык моделирования на сегодняшний день – UML. При проектировании различных систем на данном языке.

# Проектирование диаграммы последовательности

Диаграмма последовательности отражает взаимодействие определенного набора объектов на некоторой временной оси.

Основными ее элементами являются обозначения объектов, линии жизни объектов и стрелки, показывающие обмен сигналами или сообщениями между ними.

На этой диаграмме изображаются только те объекты, которые непосредственно участвуют во взаимодействии. Для диаграммы последовательности ключевым моментом можно назвать именно динамику взаимодействия объектов во времени. Графически каждый объект изображается прямоугольником и располагается в верхней части своей линии жизни.

Линия жизни объекта изображается пунктирной вертикальной линией. Она служит для обозначения периода времени, в течение которого объект существует в системе. Отдельные объекты, выполнив свою роль в системе, могут быть уничтожены путем обрыва линии жизни.

Для программы «WWW-конференция», моделируемой в данном курсовом проекте можно выделить несколько вариантов развития событий. Первый – человек хочет авторизоваться в системе. Этот процесс показан на рисунке 1:

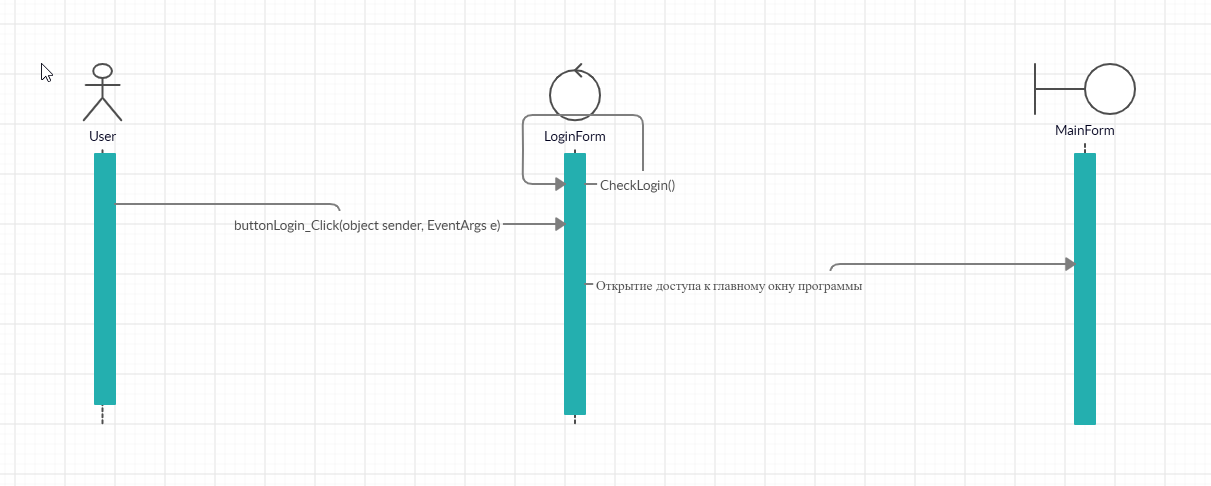


Рисунок 1 – Диаграмма последовательности (авторизация)

Как видно из рисунка, приведённого выше для данной последовательности существует хороший и плохой вариант развития событий. Функция «CheckLogin()» проверяет имеется ли такой пользователь в базе данных. Если такого пользователя нет, то его просят ввести данные повторно или зарегистрироваться. Процесс регистрации показан ниже.

Следующий вариант развития событий показывает, цепочку действий для регистрации. Этот процесс показан на рисунке 2:

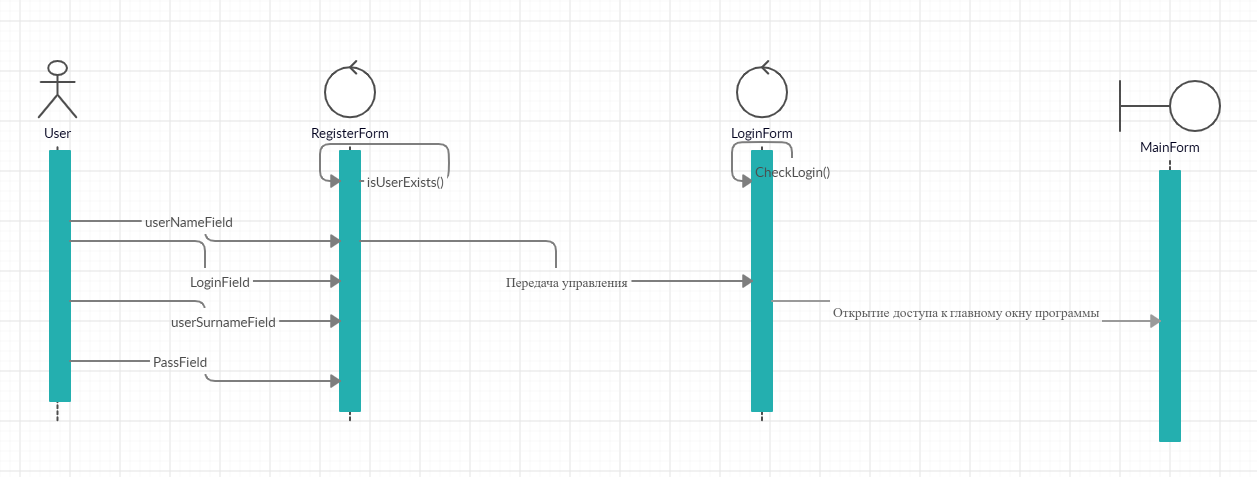


Рисунок 2 – Диаграмма последовательности (регистрация)

Третий вариант развития событий демонстрирует последовательность действий, необходимую для добавления темы на конференцию. Этот процесс показан на рисунке 3:

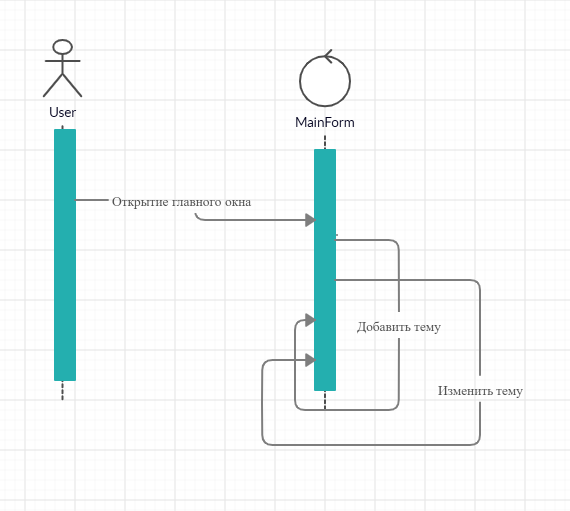


Рисунок 3 – Диаграмма последовательности (добавление темы)

Четвертый вариант развития событий демонстрирует последовательность действий, необходимую для обсуждения темы. Этот процесс показан на рисунке 4:

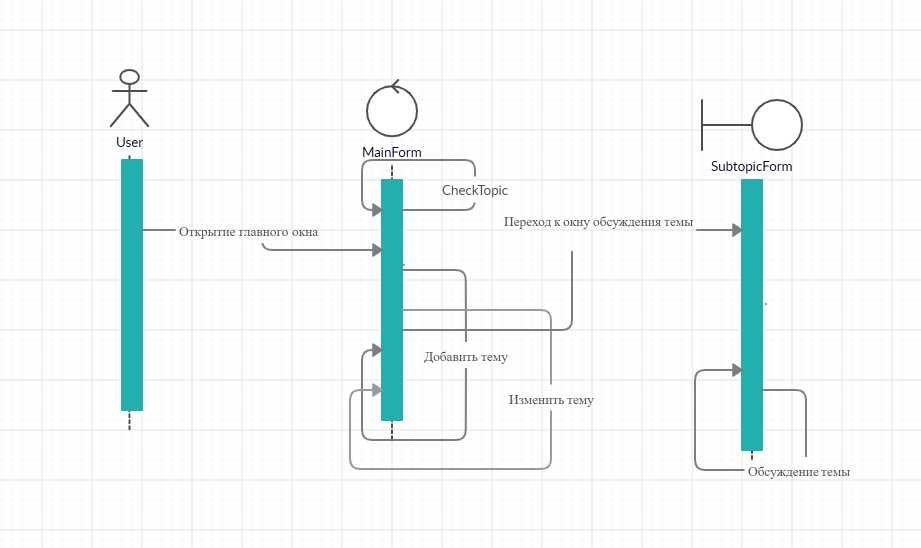


Рисунок 4 – Диаграмма последовательности (обсуждение темы)

Таким образом при помощи диаграммы последовательности были описаны цепочки действий, происходящие между объектами системы при всевозможных развитиях событий.

# Проектирование диаграммы BPMN

BPMN ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Business Process Model and Notation) — система условных обозначений и их описания в XML для [моделирования бизнес-процессов](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D0%B2&action=edit&redlink=1).

Спецификация BPMN описывает условные обозначения и их описание в XML для отображения бизнес-процессов в виде диаграмм. Этот язык использует набор интуитивно понятных элементов, которые позволяют описывать сложные процессы.

Язык описания бизнес-процессов опирается на следующие базовые объекты:

* Event – Событие;
* Activity – Действия;
* Gateway – Шлюзы или Развилки;
* Flow – Поток.
* Date – Данные;
* Artefact – Артефакты;
* Pool (Пул) - набор.

События изображаются как окружность. Они показывают действия или являются их результатами. Они могут быть начальными, конечными и промежуточными. В диаграмме на рисунке 11 имеется одно начальное событие – это «Человек открыл программу». Также есть конечное событие такое, как «Завершение работы».

Activity – это те задачи, которые должны быть выполнены на определенном этапе бизнес-процесса. Их обозначают в виде прямоугольников, в которых необходимо вписать суть действия. Действия могут быть элементарными и не элементарными. Например на рисунке 11 показано такое действие, как «Авторизация пользователя».

Gateway – это контрольный узел, который появляется в случае условного ветвления бизнес-процесса. Его изображают в виде ромба. Они также необходимы, когда порядок действий зависит от какого-либо условия, результат выполнения которого влияет на дальнейшее направление движения.

Поток – это последовательность действий. Его обозначают в виде стрелки. Он показывает какое действие необходимо совершить.

Пул – это объект описывающий какой-то один процесс на диаграмме. Их может быть несколько на диаграмме. Даже если ни одного пула не изображено, один всегда есть. Его можно развернуть для просмотра деталей. Так на рисунке 11 имеется пул, который называется «Авторизация». В нем показан процесс авторизации пользователя.

Date – это элемент, который показывает, какие данные и документы нужны для того, чтобы какое-то действие запустилось, либо которые являются результатом выполненного действия. Онипоказывают читателю, какие данные необходимы действиям для выполнения и какие данные действия производят. Так на диаграмме показана БД «Данные о пользователях», которая хранит данные о зарегистрированных пользователях.

Диаграмма BPMN показана на рисунке 5:

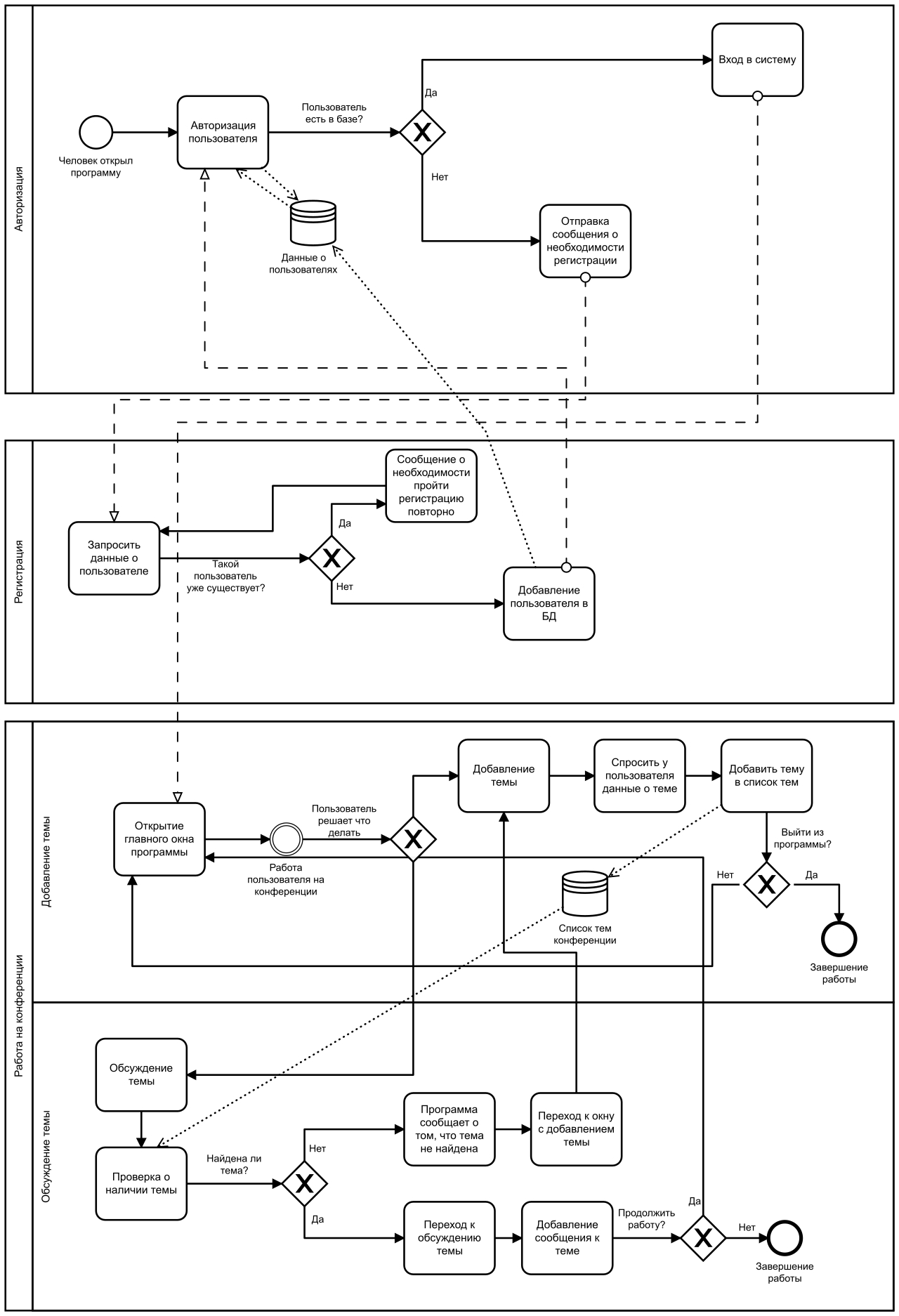


Рисунок 5 – Диаграмма BPMN

# Проектирование диаграммы EPC

EPC (событийная цепочка процессов) – это тип диаграмм, которые используются для моделирования, анализа и реорганизации бизнес-процессов. Они также могут использоваться для моделирования отдельных частей сложных систем и служить заменой традиционных блок-схем.

Этот тип диаграмм был разработан Августом-Вильгельмом Шеером в рамках работ над созданием методологии ARIS в начале 1990-х годов. Используется многими организациями для создания и анализа бизнес-процессов.

Этот тип диаграмм состоит из таких основных частей, как:

* События – состояние, которое встречается перед или после функции, то есть фиксирует состояние определённых параметров на определенный момент времени. Примеры событий: «договор подписан», «требование зафиксировано». Они изображаются в виде шестиугольника. Диаграммы как начинаются, так и заканчиваются событием.
* Функции – определенное действие, выполняемое в течение некоторого промежутка времени.
* Файл, база данных – информация, представляемая в компьютерном виде.
* Организационная единица – должность в организации или подразделение организации, элемент, которому может быть поручено выполнение функции.
* Поток управления – создает логическую последовательность между событиями и функциями. Обозначается в виде стрелок.
* Поток информации – Связывает действие и элемент, являющийся источником или приемником информации.
* Путь процесса – элемент, показывающий взаимосвязь с другими процессами.

На рисунке 6 показана EPC диаграмма:

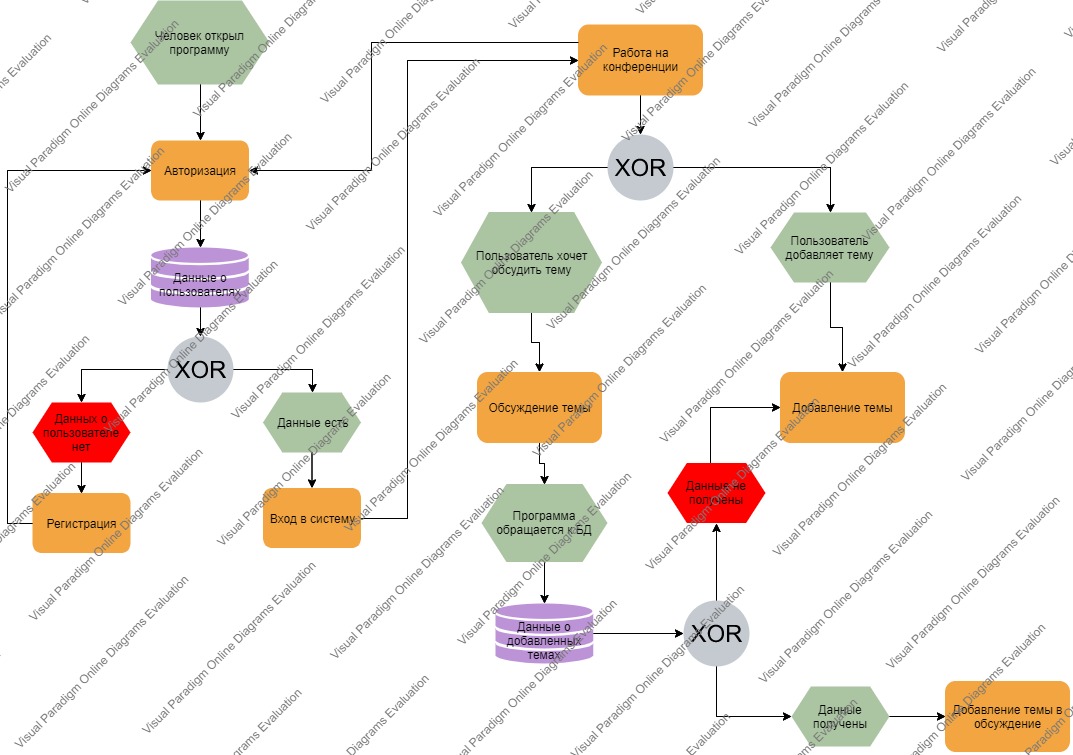


Рисунок 6 – Диаграмма EPC

# Создание диаграммы IDEF0

IDEF0 – это графическая нотация, предназначенная для описания бизнес-процессов. Описание системы с ее помощью называется функциональной моделью. Такая модель служит для описания существующих систем. Для этого могут использоваться как естественный, так графический языки. Для передачи информации о системе с помощью графического языка используется IDEF0 методология. Применение данной модели позволит чётко зафиксировать какие информационные объекты принимают участие в жизненном цикле системы, какая информация будет поступать на вход и что будет получаться на выходе.

Методология IDEF0 служит для построения иерархической системы диаграмм, которые представляют собой единичные описания фрагментов системы. В начале описывается контекстная диаграмма, т.е. проводится описание системы в целом и ее взаимодействие с окружающим миром. Затем необходимо провести функциональную декомпозицию, т.е. разбить систему на подсистемы, где каждая подсистема описывается отдельно.

IDEF0 диаграмма содержит блоки и стрелки. Блоки изображают функции исследуемой системы, а стрелки показывают взаимодействие между ними. IDEF0 требует, чтобы в диаграмме было не более шести блоков. Это необходимо для более доступного чтения.

Блоки в IDEF0 необходимо размещать по степени важности, как ее понимает автор. Этот порядок называется доминированием. Наиболее доминирующий блок обычно размещается в верхнем левом углу диаграммы.

Также каждая сторона блока имеет определенное значение. Левая предназначена для входов, верхняя нужна для управления, правая – для выходов, нижняя – для механизмов.

Для каждой функции существует правило сторон:

* стрелкой слева обозначаются входные данные, которые представляют собой объекты, используемые функцией для получения результата.
* стрелкой сверху – управление, которое представляет собой информацию, управляющую действиями системы.
* стрелкой справа – выходные данные, которые представляют собой результат работы функции.
* стрелкой снизу – механизм, которые представляют собой ресурсы, выполняющие работу.

Учитывая всё вышеперечисленное на рисунке 7 была составлена IDEF0 диаграмма системы «WWW-конференция».

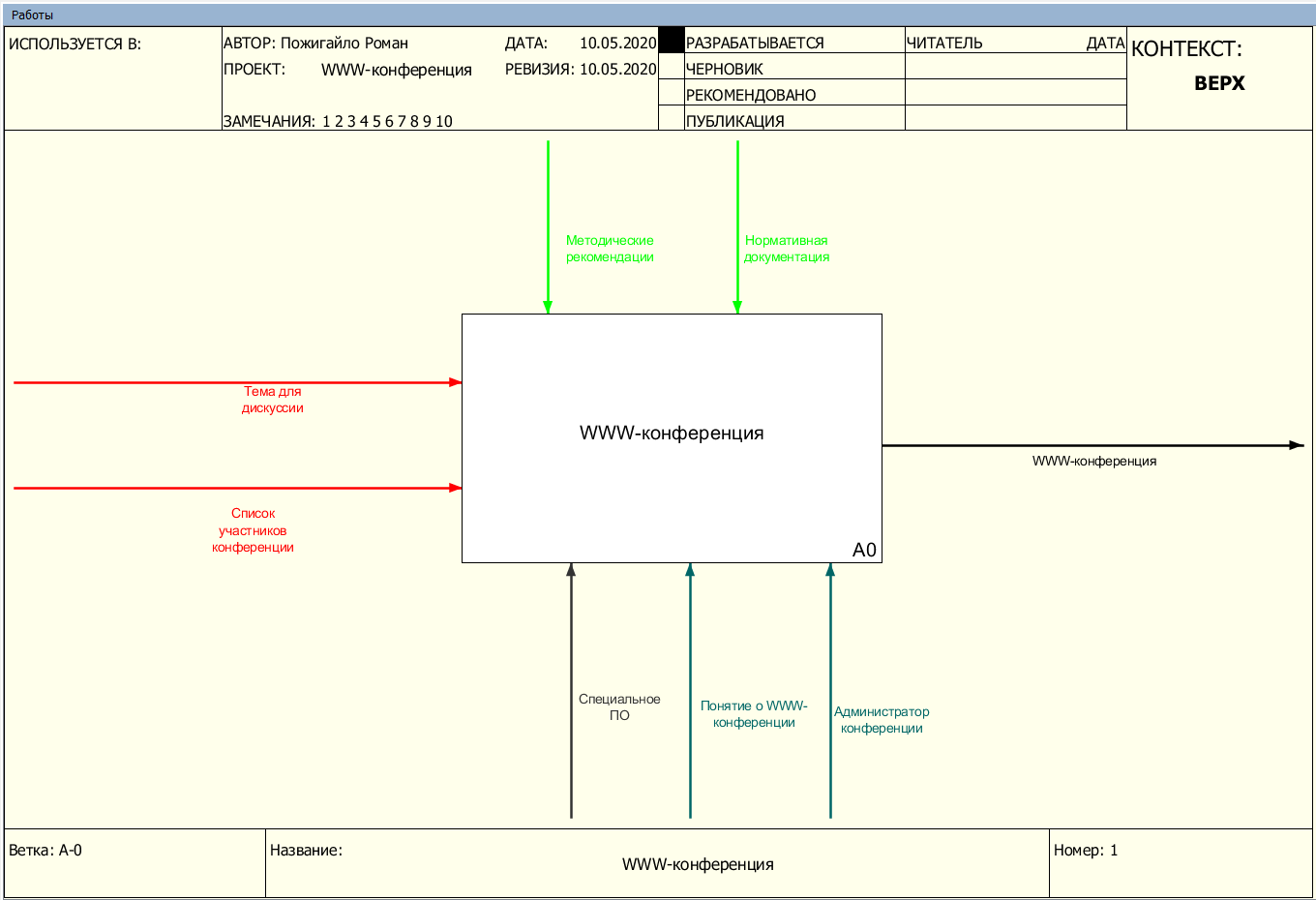


Рисунок 7 – IDEF0 диаграмма системы «WWW-конференция»

В качестве входных данных для данной системы выступают тема для дискуссии, список участников конференции. Это те вводные, которые необходимы для начала работы.

Управление происходит благодаря методическим рекомендациям по проекту и нормативной документации. Это та необходимая информация для реализации проекта.

Механизмом реализации работы системы являются администратор конференции, специальное ПО, само понятие о WWW-конференциях. Это те инструменты, которыми будут пользоваться разработчики в процессе создания проекта.

Результатом деятельности будет являться создание программного обеспечения для проведения WWW-конференции.

Полученная модель системы может быть представлена в более подробном виде путём разбиения на большее количество составных элементов.

На рисунке 8 можно видеть модель системы «WWW-конференция» после декомпозиции.

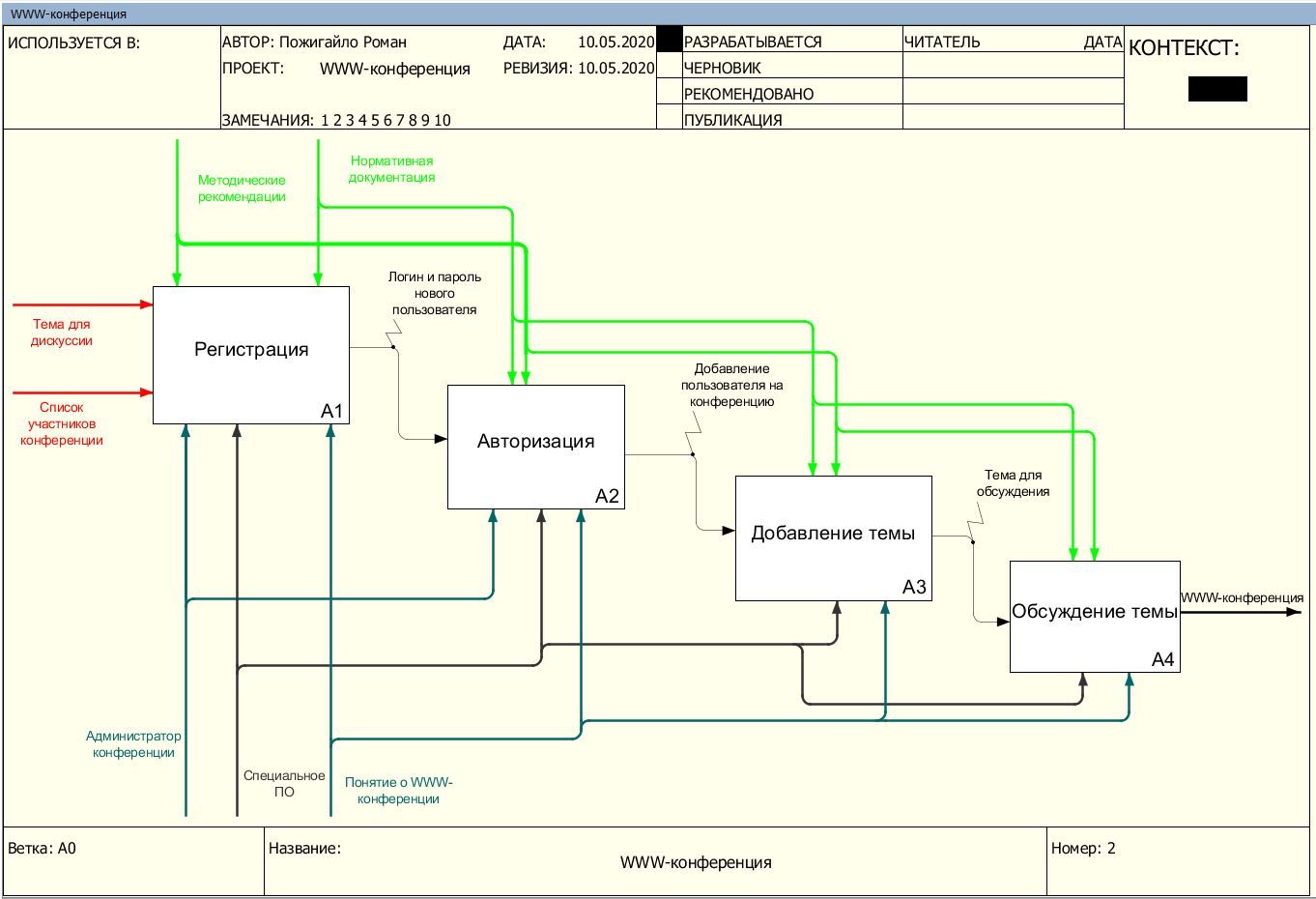


Рисунок 8 – Декомпозиция системы «WWW-конференция»

Таким образом процесс создания проекта разбивается на четыре основных функции:

* регистрация;
* авторизация;
* добавление темы;
* обсуждение темы.

Также данная диаграмма имеет 3 уровень вложенности. Например, можно произвести декомпозицию блока «Регистрация». Она показана на рисунке 9:

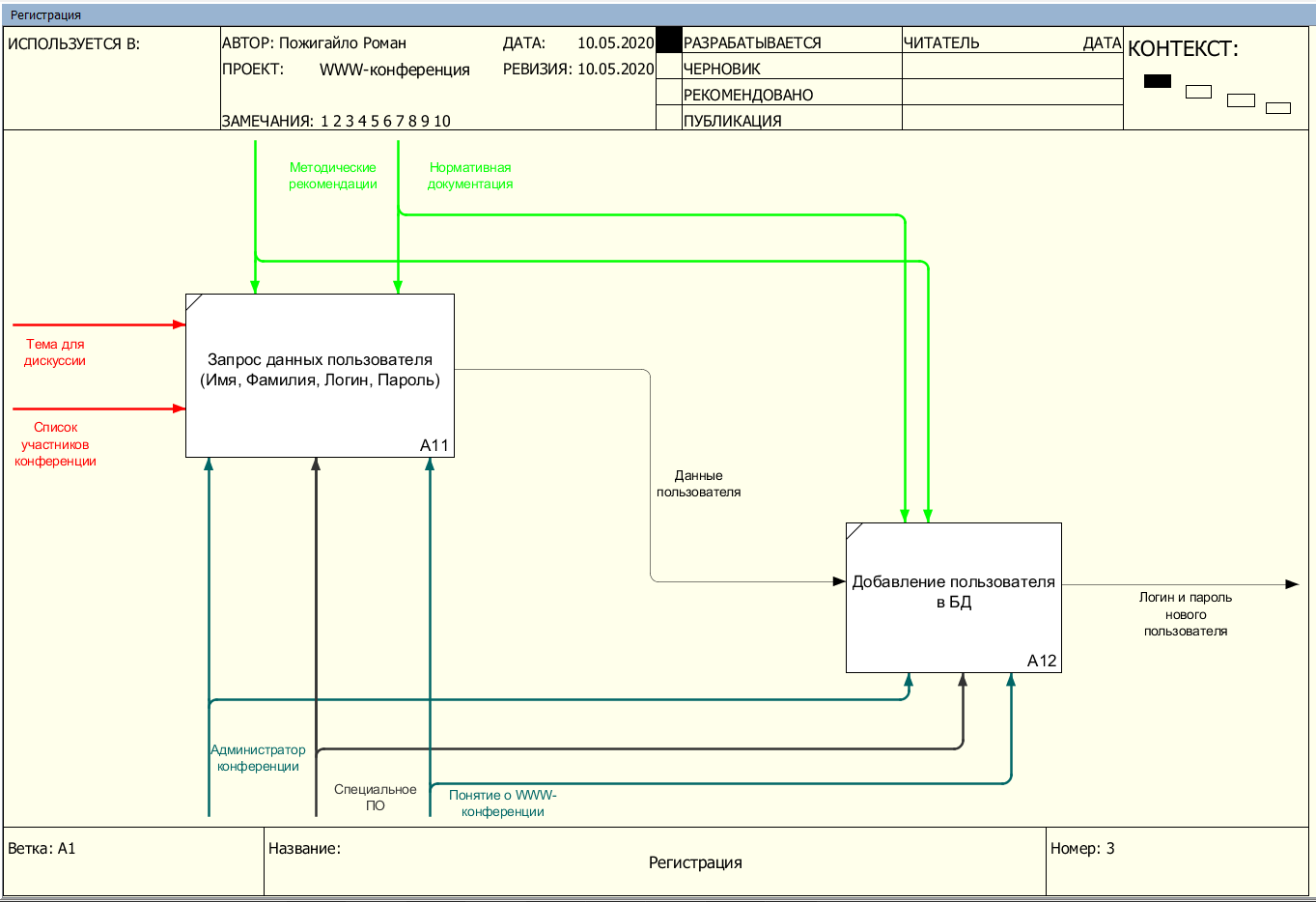


Рисунок 9 – Декомпозиция блока «Регистрация»

Здесь показан процесс регистрации пользователя. Он состоит из запроса данных и добавление их в БД. По итогу мы получаем готовый программный продукт, который можем в дальнейшем использовать. Остальные части диаграммы показаны в приложении А.

# Диаграмма потоков данных (DFD)

Диаграмма потоков данных DFD (DataFlowDiagrams) – это нотация, предназначенная для моделирования информационный систем с точки зрения хранения, обработки и передачи данных. Применяется этот вид нотации в случае, когда требуется описание системы как хранилища данных. ДиаграммаDFD – это один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML.

Непосредственно DFD нотация состоит из следующих элементов:

* **Процесс** – функция или последовательность действий, которые нужно предпринять, чтобы данные были обработаны. Это может быть создание заказа, регистрация клиента и т.д. В названиях процессов принято использовать глаголы, т.е., например, «создать клиента» или «обработать заказ».
* **Внешние сущности** – это любые объекты, которые не входят в саму систему, но являются для нее источником информации либо получателями какой-либо информации из системы после обработки данных. Это может быть человек, внешняя система, какие-либо носители информации и хранилища данных.
* **Хранилище данных** – внутреннее хранилище данных для процессов в системе. Поступившие данные перед обработкой и результат после обработки, а также промежуточные значения должны где-то храниться. Это и есть базы данных, таблицы или любой другой вариант организации и хранения данных.
* **Поток данных** – в нотации отображается в виде стрелок, которые показывают, какая информация входит, а какая исходит из того или иного блока на диаграмме.

В результате была получена следующая диаграмма DFD, изображенная на рисунке 10:

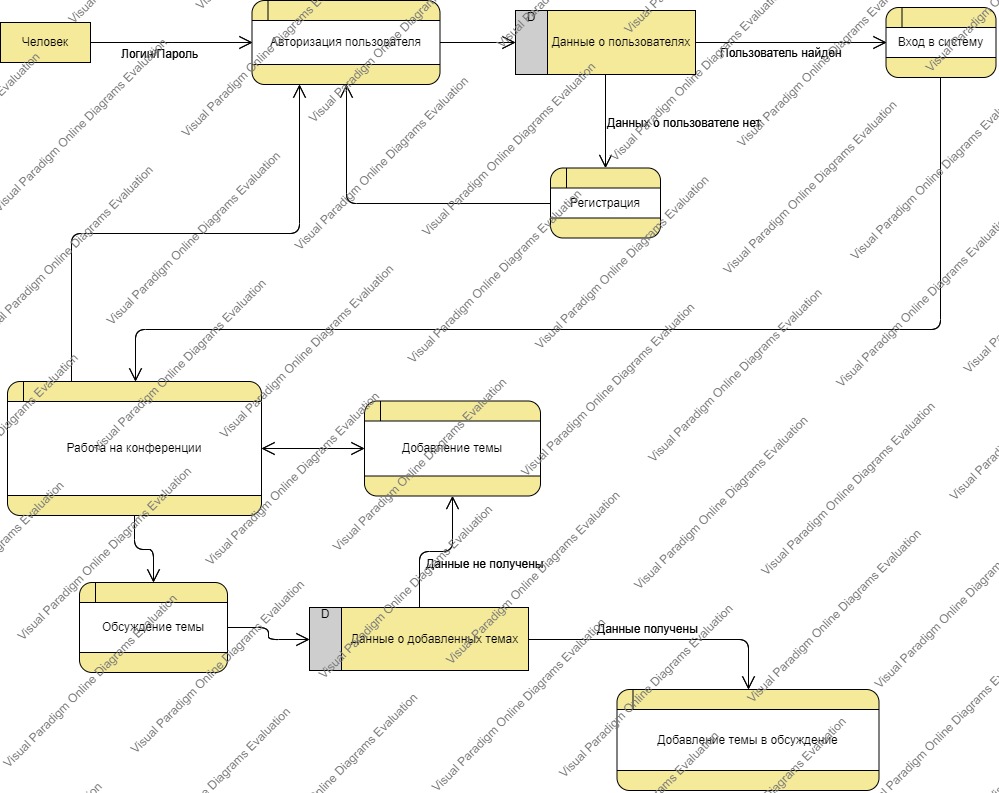


Рисунок 10 – Диаграмма DFD системы «WWW-конференция»

В результате получается следующая последовательность:

1. Пользователь производит авторизацию и предоставляет свои данные.
2. Программа проверяет есть ли пользователь в базе данных.
3. Если пользователь есть в базе данных, то программа пускает его на главный экран. Если же пользователя нет в базе, то он регистрируется.
4. Затем пользователь переходит к добавлению темы на конференцию или к обсуждению другой темы.

С точки зрения DFD у нас имеются:

1. Пользователь – это внешняя сущность.
2. Несколько процессов такие, как «Авторизация пользователя», «Регистрация», «Вход в систему», «Добавление темы» и другие.
3. «Данные о добавленных темах» и «Данные о пользователях» – это хранилища данных.

# Проектирование диаграммы вариантов использования

Диаграмма вариантов использования описывает функциональные требования системы с точки зрения способов её применения. Данная модель позволяет связать то, что нужно разработчикам от системы, с тем, как система удовлетворяет эти потребности. Данная диаграмма должна состоять из актеров, вариантов использования и связей между ними.

Ее суть состоит в том, что система представляется в виде актеров, которые взаимодействуют с системой. Они могут взаимодействовать исходя из различных вариантов использования. Актером может быть любой объект, который может взаимодействовать в исследуемой системой извне. Вариант использования – это набор функций, которая система может предоставить актеру.

Согласно UML актера графически можно изобразить в виде «человечка». Варианты использования обозначаются на диаграмме эллипсом, внутри которого содержится его описание. Вариант использования представляет собой конечную последовательность действий, которую может выполнить актер по отношению к системе.

В данной работе необходимо было реализовать программное обеспечение для WWW-конференции, которая представляет собой хранилище сообщений. Для каждого сообщения конференции хранятся значения следующих полей: номер сообщения, автор, тема, текст сообщения, дата добавления сообщения, ссылка на родительское сообщение. Пользователь может как добавить свою тему для обсуждения, так и присоединиться к обсуждению другой темы. Таким образом, можно сделать вывод, что WWW-конференция является одним из актеров. В качестве второго актера можно выделить человека, который будет пользоваться данным программным обеспечением.

Учитывая всё вышеперечисленное была построенная диаграмма, вариантов использования, на которой описаны всевозможные действия, производимые конференцией и человеком, работающим на ней. Данная диаграмма представлена на рисунке 11:



Рисунок 11 – Диаграмма вариантов использования

# Проектирование диаграммы классов

Диаграмма классов – это одна из основных методик моделирования, которая используется практически во всех объектно-ориентированных подходах. Она показывает классы, интерфейсы, объекты и их отношения.

Диаграмма классов содержит типы классов системы и связи, которые существуют между ними. Также отображаются атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Атрибуты представляют собой поля объектов класса. Объекты класса различаются друг от друга или получают свою индивидуальность благодаря различиям в их атрибутах. Однако существуют такие объекты, которые имеют идентичные значения атрибутов и взаимосвязей. Имя атрибута должно быть уникально в пределах класса. За его именем должен следовать его тип (integer, boolean, string и др.).

Операция представляет собой функцию класса. Она может иметь параметры и возвращать или не возвращать значение.

Данная диаграмма описывает типы объектов в системе и различные виды статических отношений, которые существуют между ними. Существуют три основных типа связей:

* + Ассоциация – представляет отношения между экземплярами классов.
  + Наследование – позволяет определить такое отношение между классами, когда один класс обладает поведением и структурой другого класса.
  + Агрегация – форма композиции объектов в объектно-ориентированном дизайне.

Классы на данной диаграмме представляют собой сущности так или иначе фигурирующие в системе. У каждой из сущностей есть свои атрибуты и функции.

В системе, описывающей программное обеспечение WWW-конференции, можно выделить следующие классы:

* User (человек, использующий программу);
* RegisterForm (окно регистрации пользователя);
* LoginForm (окно авторизации пользователя);
* MainForm (главное окно где находятся темы конференции);
* SubtopicForm (окно обсуждения тем).

Также рассматриваемая диаграмма определяет связи между сформированными классами. В этих связях присутствуют цифровые обозначения, показывающие тип связи. Например, на сформированной диаграмме классов видно, что одновременно только один объект класса User может взаимодействовать с единственным объектом класса LoginForm.

На рисунке 12 показана диаграмма классов:

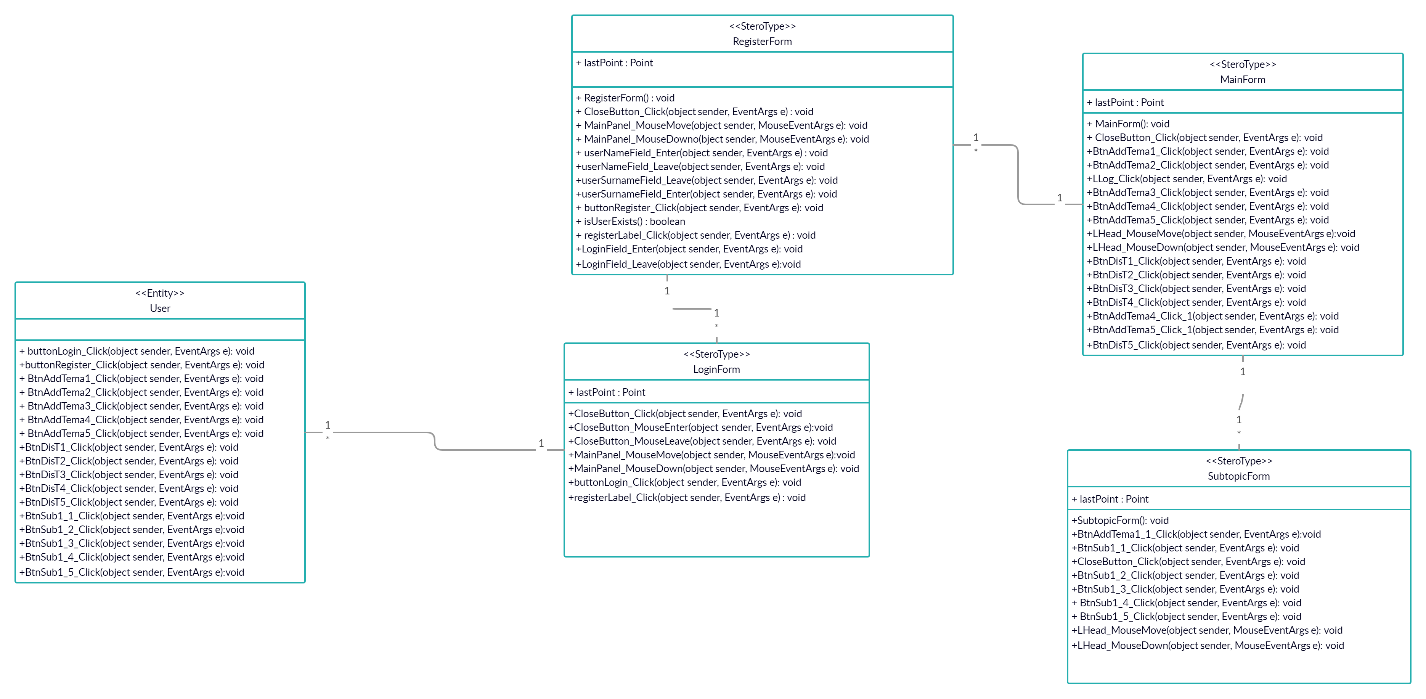


Рисунок 12 – Диаграмма классов

# Проектирование диаграммы Ганта

Диаграмма Ганта – это инструмент для планирования и управления задачами. Его придумал американский инженер Генри Гант. Эта диаграмма состоит из горизонтальных полос, которые располагаются между двумя осями: списком задач по вертикали и датами по горизонтали.

На диаграмме можно посмотреть не только задачи, которые следует выполнить, но и их последовательность. Это позволяет ни о чем не забыть.

Эта диаграмма состоит из полос, которые ориентированы вдоль оси времени. Каждая полоса на диаграмме представляет собой отдельную задачу в составе проекта. Концы этой полосы представляют собой начало и конец поставленной задачи, а ее протяженность – длительность работы.

Также на диаграмме могут быть отмечены процент завершения для каждой задачи, вехи, имена людей, которые выполняют ту или иную задачу.

Веха является ключевым понятием. Это метка значимого момента в ходе выполнения задачи, общая граница двух или более задач. Сдвиг вехи приводит к сдвигу всего проекта, так как все задачи связаны между собой.

На рисунке 13 показана диаграмма Ганта для проекта WWW-конференция:

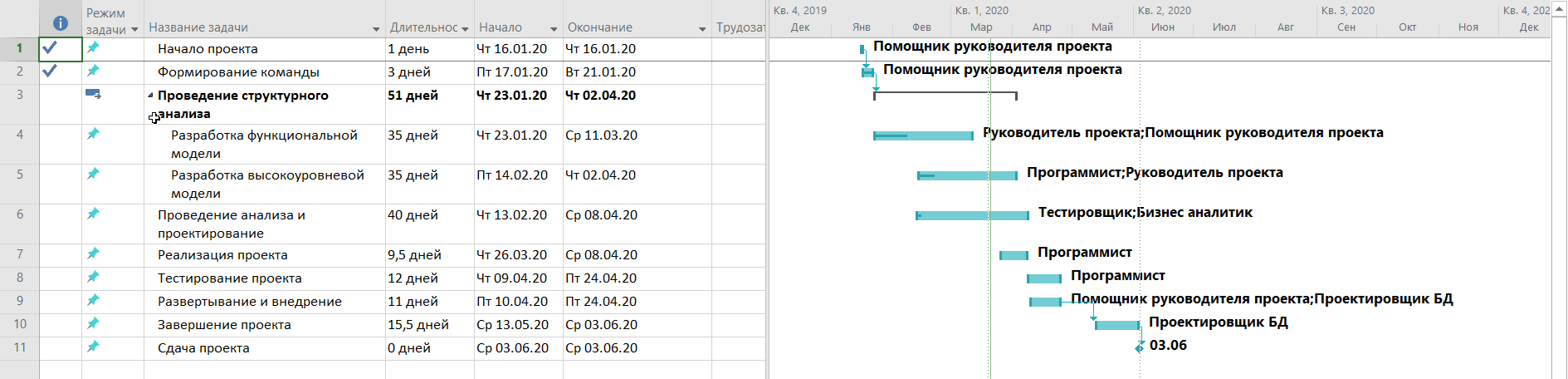


Рисунок 13 – Диаграмма Ганта

# Требования к системе: классификация FURPS+

Классификация требований к системе FURPS+ была разработана Робертом Грэйди и предложена в 1992 году. Сокращение FURPS расшифровывается так:

1. Functionality, функциональность
2. Usability, удобство использования
3. Reliability, надежность
4. Performance, производительность
5. Supportability, поддерживаемость
6. + необходимо помнить о таких возможных ограничениях, как:
   * ограничения проектирования, design
   * ограничения разработки, implementation
   * ограничения на интерфейсы, interface
   * физические ограничения, physical

Функциональность включает в себя основные свойства и функции системы. В качестве функциональных возможностей системы могут выступать возможность автоматизированного выполнения функций отдельных бизнес-процессов, автоматизация отдельных функций по информативной поддержке работы с партнерами и др.

Удобство использования включает в себя такие виды требований, как эстетика и логичность пользовательского интерфейса, защита от человеческого фактора, справочная информация о системе и др.

Надежность она включает в себя такие характеристики системы, как сбои, предсказуемость поведения, время готовности системы к работе, точность вычислений и др.

Производительность системы включает в себя такие характеристики, как скорость работы, эффективность, потребление ресурсов.

К поддержке относятся: тестирования, расширения системы, адаптации, совместимости, портативность и др.

Стоит отметить, что классификация FURPS+ содержит не только требования к ИС, но и ограничения, которые выделены в названии знаком «плюса». Выделяют четыре вида ограничений:

* ограничения проектирования (Design). В качестве примера может служить требование использовать реляционную базу данных в качестве основной;
* ограничения разработки (Implementation). В рамках данных ограничений может указываться конкретная методология и др.;
* ограничения интерфейсов (Interface), к которым относят ограничения на конкретные форматы данных;
* физические ограничения (Physical), включающие все возможные ограничения внешней среды. В качестве такого ограничения может служить уровень влажности, температурный режим и пр.

В качестве требований к системе рассматриваемого курсового проекта по классификации FURPS+ можно выделить следующие:

* Functionality (функциональность) – представляет собой хранилище сообщений. Для каждого сообщения конференции хранятся значения следующих полей: номер сообщения, автор, тема, текст сообщения, дата добавления сообщения, ссылка на родительское сообщение. Сообщения добавляются в конференцию зарегистрированными пользователями.
* Usability (удобство использования) – интуитивная понятность; красивый дизайн и приятный пользовательский интерфейс.
* Reliability (надежность) – в случае сбоя программы информация теряется; режим доступности системы 24/7; данные о пользователях хранятся в БД; 1 сбой/1 год.
* Performance (производительность) – время, необходимое для запуска и завершения работы не превышает 10 с; время отклика системы не превышает 0.5 с; 100% эффективность работы.
* Supportability (поддерживаемость) – совместимость с ОС Windows; простота установки ПО; работоспособность зависит от стороннего ПО.
* “+” – ограничений нет.

# Тестирование программы

При открытие программы пользователя встречает окно авторизации пользователя, представленное на рисунке 14. Если пользователь есть в базе, то он вводит свой логин и пароль и далее попадает на главное окно программы.

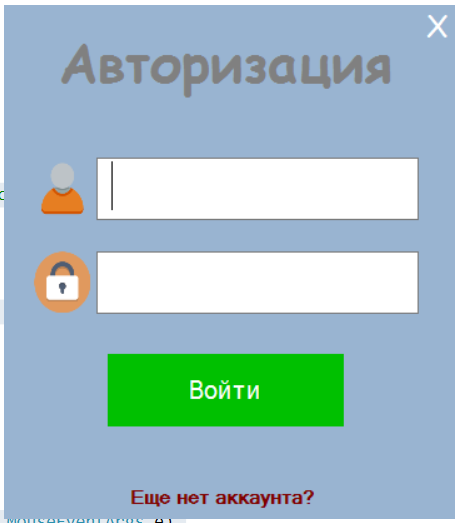


Рисунок 14 – окно авторизации

Однако рассмотрим случай, когда пользователя еще нет в базе данных. Тогда он переходит в окно регистрации, которое представлено на рисунке 15:

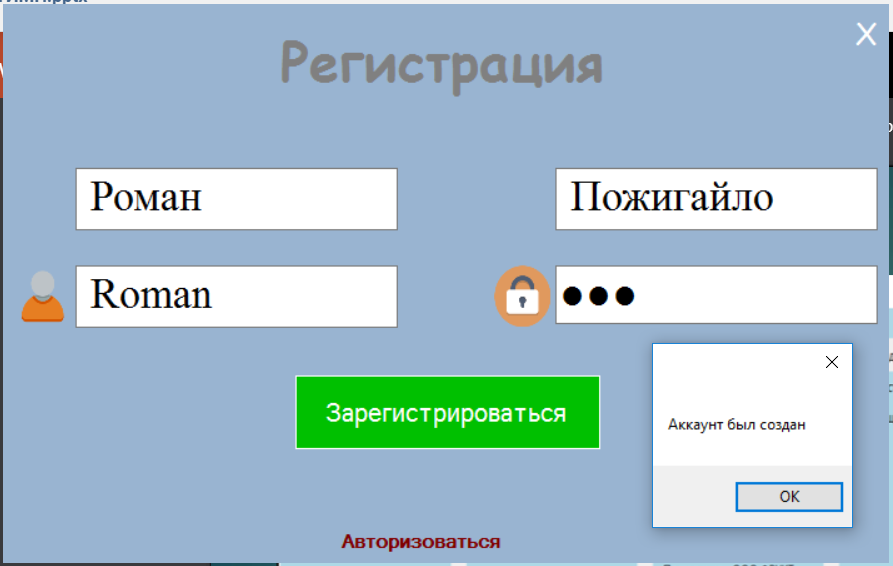


Рисунок 15 – окно регистрации

Далее пользователь снова вводит свои данные в окно авторизации, представленное на рисунке14, и попадает на главное окно программы. Оно показано на рисунке 16:



Рисунок 16 – главное окно программы

Далее пользователь может добавить новую тему в конференцию. Для этого необходимо открыть окно добавления тем, представленное на рисунке 17:

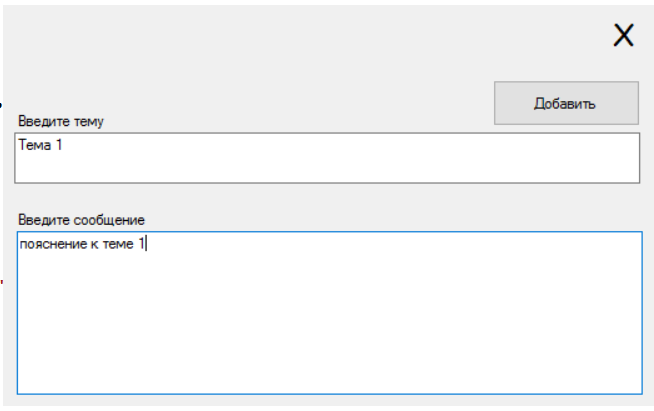


Рисунок 17 – окно добавления темы

Далее тема добавляется на конференцию (рисунок 18):



Рисунок 18 – тема добавлена на конференцию

Далее пользователь может приступить к обсуждению темы, для этого ему необходимо нажать на кнопку «Перейти к обсуждению». Откроется окно обсуждения темы, где можно добавить сообщение (рисунок 19):

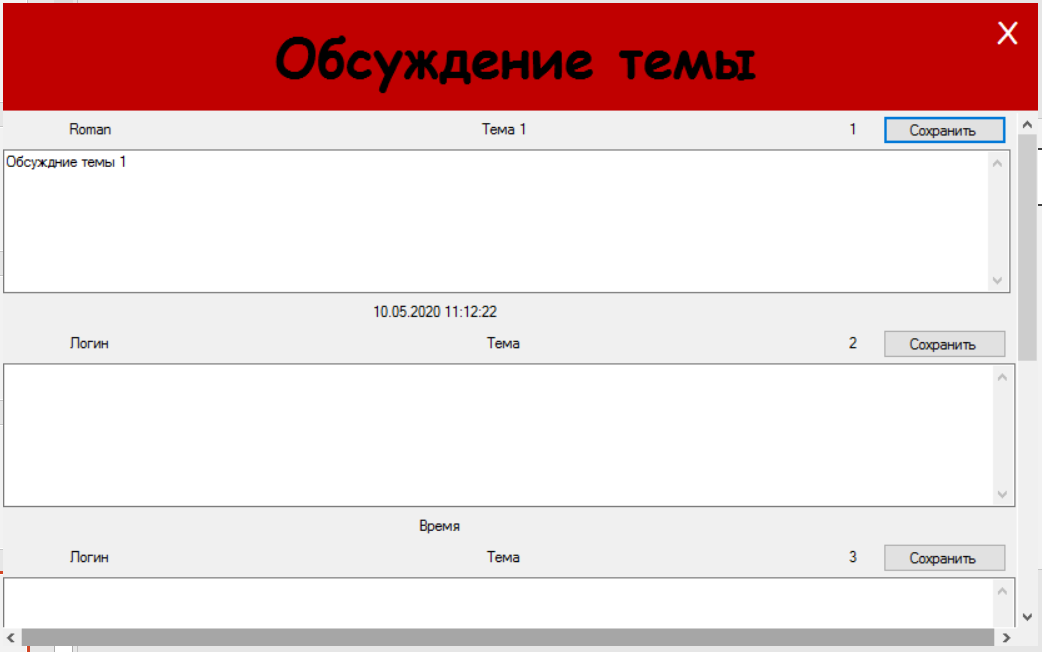


Рисунок 19 – окно обсуждения темы

# Сопровождение ПО

Представленное в курсовой работе программное обеспечение документируется в Wiki на гитхаб по данной ссылке: <https://github.com/RomanPozhig/WWW-conference/wiki>.

На рисунке 20 показано оглавление:

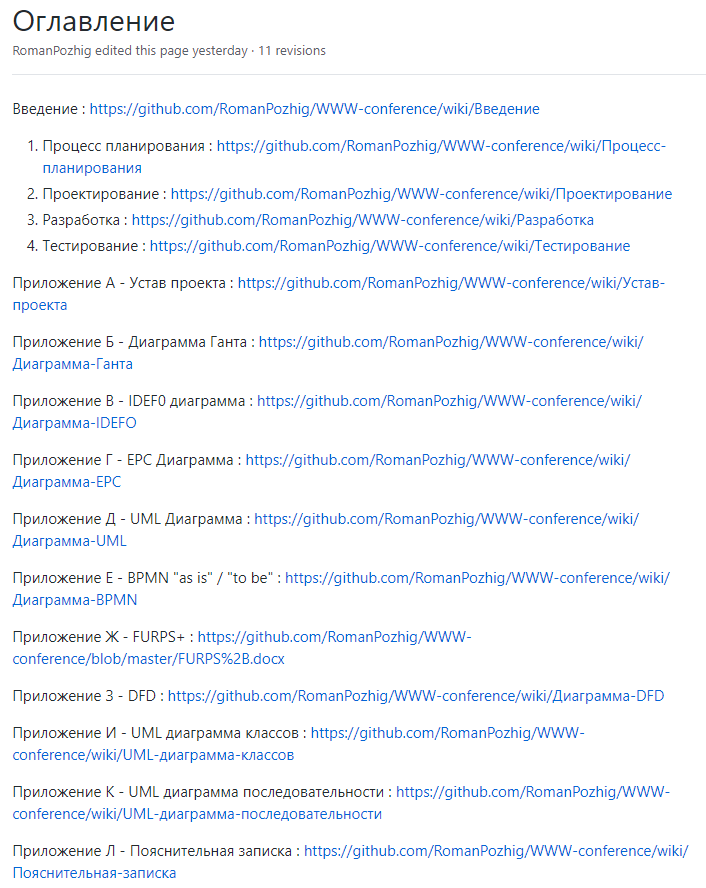


Рисунок 20 – оглавление Wiki

Можно переходить по данному оглавлению и просматривать различную документацию. Например, можно, кликнув по ссылке, перейти на «Приложение Б – Диаграмма Ганта». Данное окно представлено на рисунке 21:

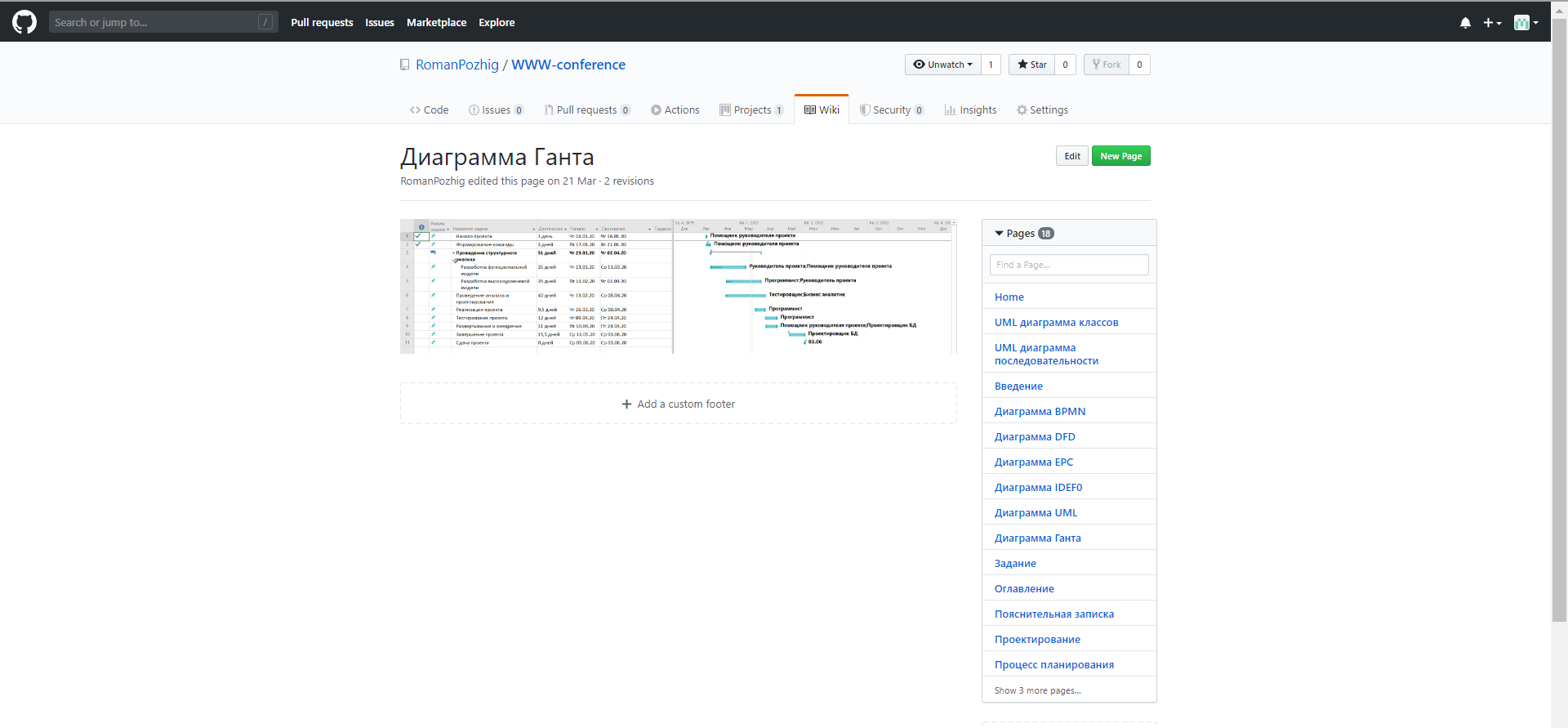


Рисунок 21 – окно «Приложения Б – Диаграмма Ганта»

И так далее переходя по интересующим ссылкам из Wiki можно просмотреть соответствующую документацию.

# Система контроля версий ПО

В рамках данного курсового проекта программа имеет две версии. В первой версии представлена только авторизация пользователя. В ней регистрация пользователя производилась путем ручного добавления его данных в БД администратором.

Во второй версии программы регистрация производится программным способом с помощью соответствующего окна программы. Так же были добавлены окна добавления темы и обсуждения темы.

# Заключение

В результате данного курсового проекта была спроектирована система «WWW-конференция» с использованием UML диаграмм и диаграмм другого вида таких, как IDEF0, DFD, EPC, BPMN и других. Они позволили наглядно показать иерархию и внутреннее строение всех ее компонентов. Полученные диаграммы имеют простое и понятное, даже для человека незнакомого с данной областью, строение. С помощью них удалось описать систему с разных сторон.

При построении диаграмм использовались основные правила и принципы моделирования, включающие графическое представление объектов и связей между ними, иерархическое построение, а также названия, отражающие назначение той или иной сущности, или взаимодействия.

При решении сложных задач заблаговременное планирование и моделирование значительно упрощает процесс. Можно провести аналогию с постройкой дома. Когда кто-то хочет построить дом, он не сразу приступает к работе. Ему необходим план постройки.

Главным преимуществом диаграмм является графическая подача материала. За счет этого его представление становится более понятным. С помощью этого возможно продемонстрировать ключевые приоритеты проекта.

Были получены важные знания и практические навыки как в области использования основных принципов языка UML и объектно-ориентированных языков моделирования в целом, так и в области построения диаграмм проектирования, отображающих поведение различных организационных структур.

# Список использованных источников

* 1. Автоматизация проектирования вычислительных систем. Языки, моделирование и базы данных / ред. М. Брейер. - М.: Мир, 2015. - 463 c.
  2. Ларман, Крэг. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / КрэгЛарман. - Москва: Гостехиздат, 2017. - 736 c.
  3. Слепцов, А.И. Автоматизация проектирования управляющих систем гибких автоматизированных производств / А.И. Слепцов, А.А. Юрасов. - М.: Техника, 2015. - 110 c.
  4. Роберт А. Максимчук. UML для простых смертных / Роберт А. Максимчук, Эрик Дж. Нейбург. - Москва: СИНТЕГ, 2014. - 272 c.
  5. Йордон, Эдвард. Объектно-ориентированный анализ и проектирование систем / Эдвард Йордон , Карл Аргила. - М.: ЛОРИ, 2014. - 264 c.
  6. SysAna– Требования к системе: классификация FURPS+ [Электронный ресурс]: - Режим доступа: <https://sysana.wordpress.com/2010/09/16/furps/> (Дата обращения 09.05.2020).
  7. Habr – Знакомство с нотацией IDEF0 и пример использования [Электронный ресурс]: - Режим доступа: <https://habr.com/ru/company/trinion/blog/322832/> (Дата обращения 09.05.2020).
  8. Habr – Зачем нам UML? Или как сохранить себе нервы и время [Электронный ресурс]: - Режим доступа: <https://habr.com/ru/post/458680/> (Дата обращения 09.05.2020).
  9. [www.trinion.org](http://www.trinion.org) – Краткое описание BPMN с примером [Электронный ресурс]: - Режим доступа: <https://www.trinion.org/blog/kratkoe-opisanie-bpmn-s-primerom> (Дата обращения 09.05.2020).
  10. GitHub – [RomanPozhig](https://github.com/RomanPozhig)/[WWW-conference](https://github.com/RomanPozhig/WWW-conference)[Электронный ресурс]: - Режим доступа: <https://github.com/RomanPozhig/WWW-conference>(Дата обращения 09.05.2020).

# Приложение А. Диаграмма IDEF0

Рисунок 22 третий уровень (Авторизация):

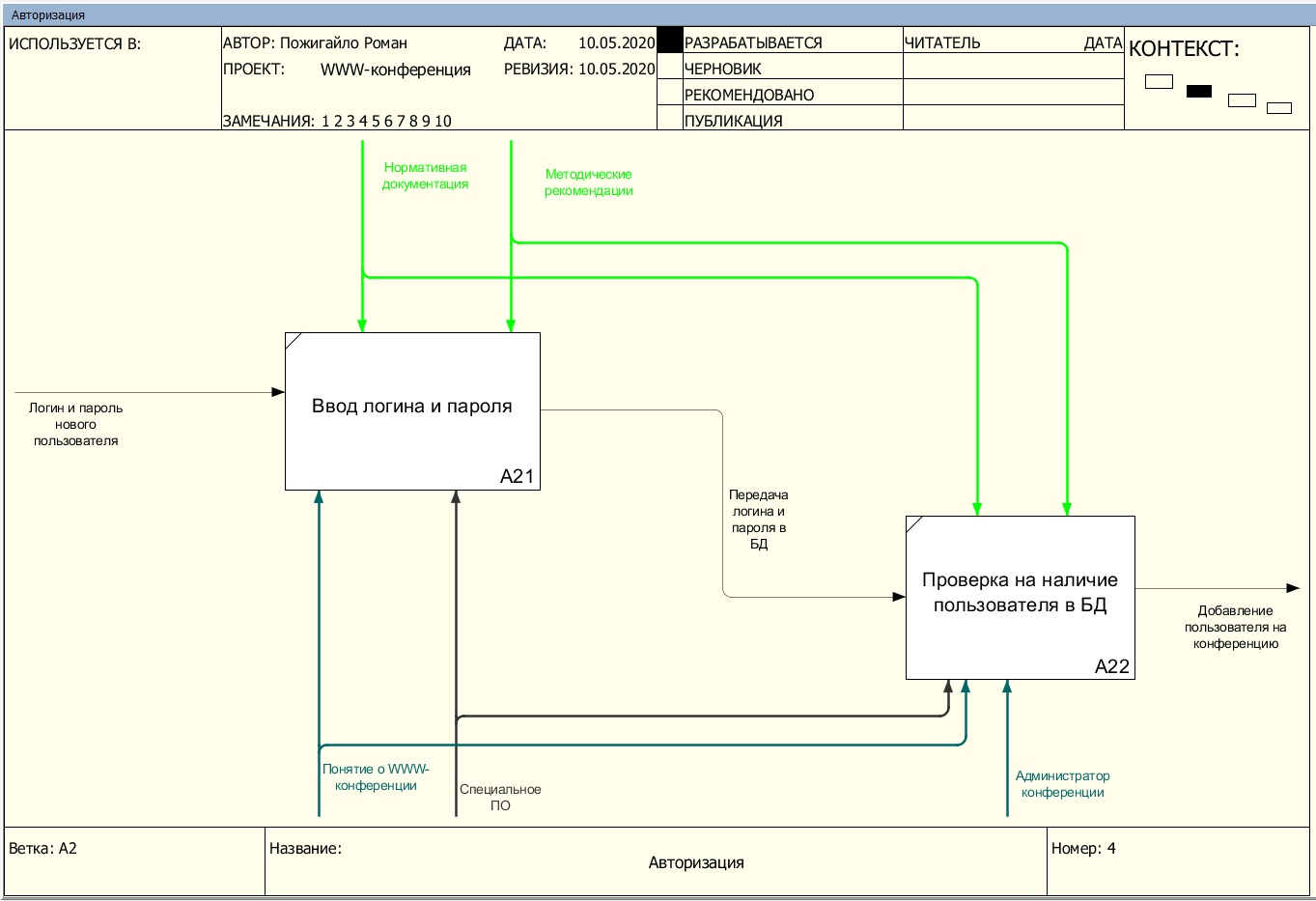


Рисунок 22 третий уровень (авторизация)

Рисунок 23 третий уровень (Добавление темы):

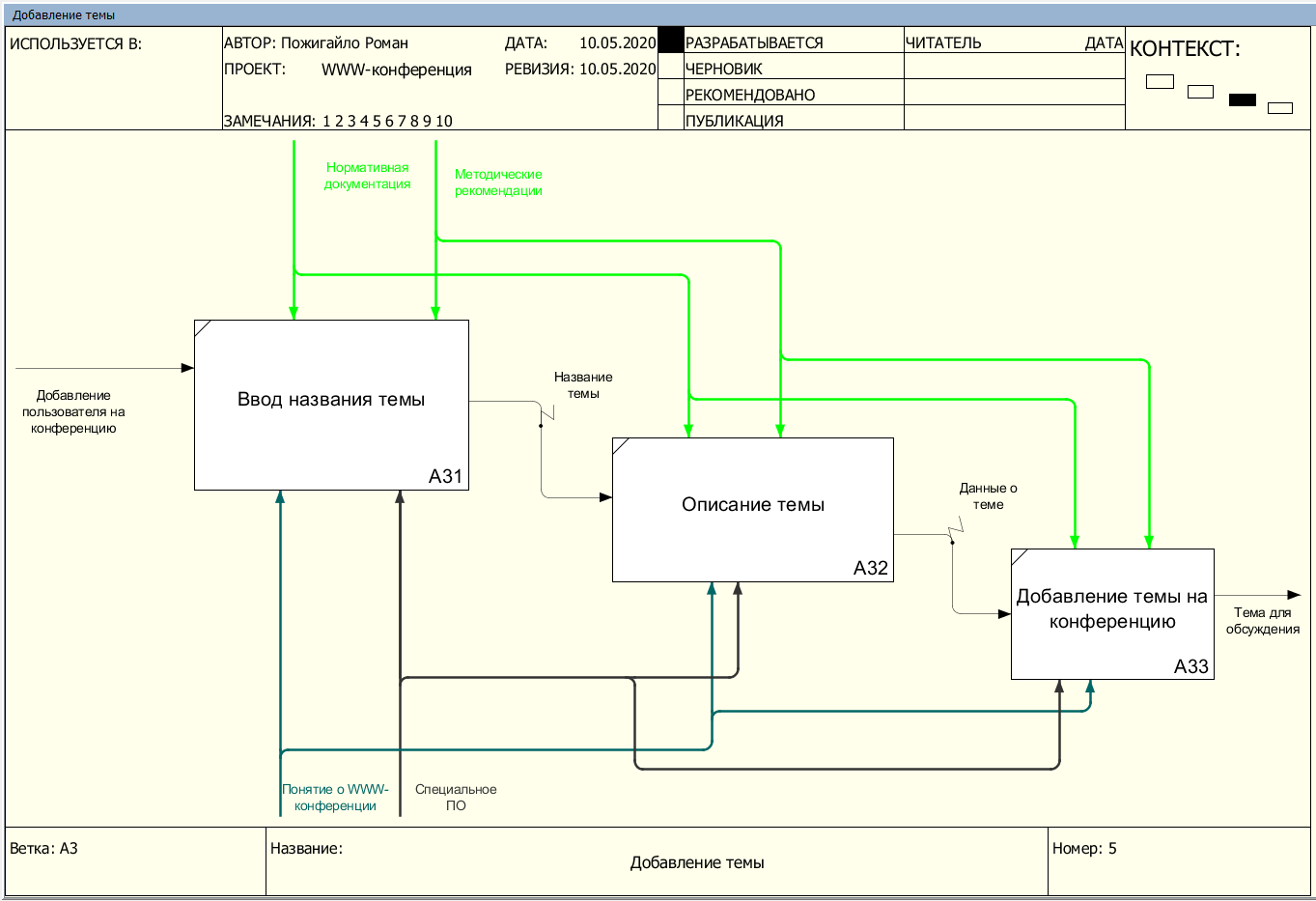


Рисунок 23 третий уровень (добавление темы)

Рисунок 24 третий уровень (Обсуждение темы):

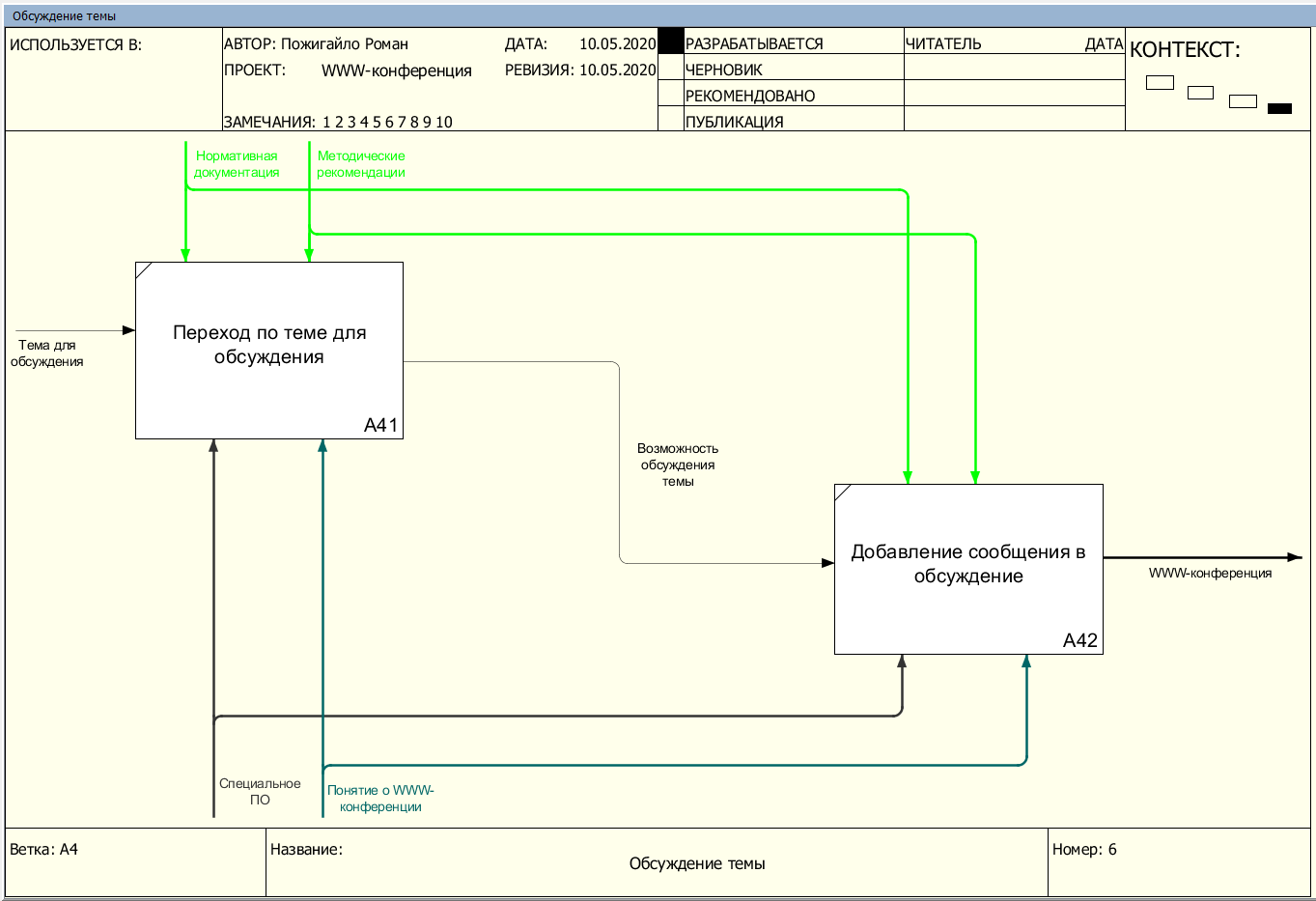


Рисунок 24 третий уровень (обсуждение темы)

# Приложение Б. Проверка на антиплагиат

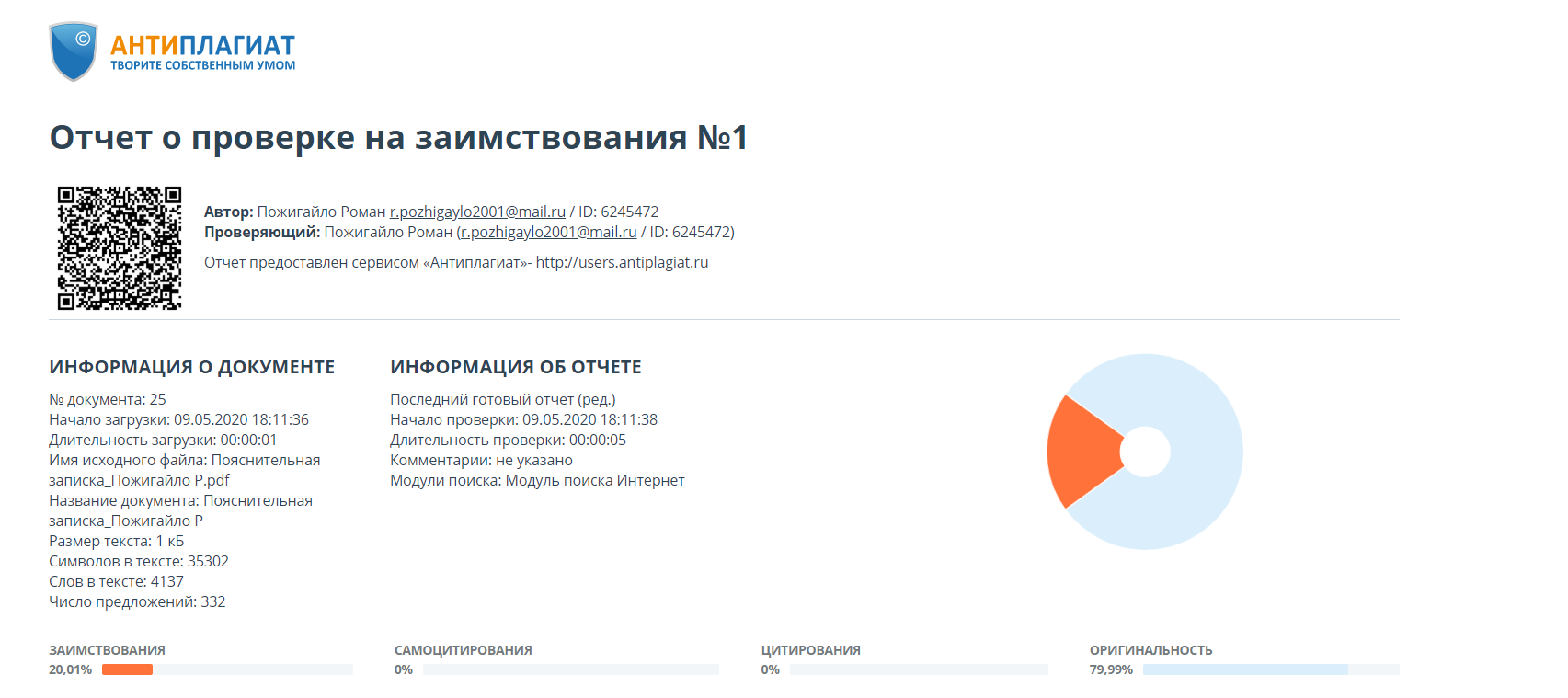


Рисунок 25 проверка на антиплагиат

# Приложение В. Код программы

Листинг 1 – код для окна авторизации

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TMP\_Kurs

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

this.PassField.AutoSize = false;

this.PassField.Size = new Size(this.PassField.Size.Width, this.LoginField.Size.Height);

}

private void CloseButton\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void CloseButton\_MouseEnter(object sender, EventArgs e)

{

CloseButton.ForeColor = Color.Green;

}

private void CloseButton\_MouseLeave(object sender, EventArgs e)

{

CloseButton.ForeColor = Color.White;

}

Point lastPoint;

private void MainPanel\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void MainPanel\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void buttonLogin\_Click(object sender, EventArgs e)

{

string loginUser = LoginField.Text;

string passUser = PassField.Text;

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `users` WHERE `login` = @uL AND `pass` = @uP", db.GetConnection());

command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = loginUser;

command.Parameters.Add("@uP", MySqlDbType.VarChar).Value = passUser;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

this.Hide();

MainForm mainForm = new MainForm();

mainForm.Show();

Topics.Login = LoginField.Text;

}

else

MessageBox.Show("No");

}

private void registerLabel\_Click(object sender, EventArgs e)

{

this.Hide();

RegisterForm registerForm = new RegisterForm();

registerForm.Show();

}

}

}

Листинг 2 – код для окна регистрации

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TMP\_Kurs

{

public partial class RegisterForm : Form

{

public RegisterForm()

{

InitializeComponent();

userNameField.Text = "Имя";

userNameField.ForeColor = Color.Gray;

userSurnameField.Text = "Фамилия";

userSurnameField.ForeColor = Color.Gray;

LoginField.Text = "Логин";

userSurnameField.ForeColor = Color.Gray;

}

private void CloseButton\_Click(object sender, EventArgs e)

{

Application.Exit();

}

Point lastPoint;

private void MainPanel\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void MainPanel\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void userNameField\_Enter(object sender, EventArgs e)

{

if (userNameField.Text == "Имя")

{

userNameField.Text = "";

userNameField.ForeColor = Color.Black;

}

}

private void userNameField\_Leave(object sender, EventArgs e)

{

if (userNameField.Text == "")

{

userNameField.Text = "Имя";

userNameField.ForeColor = Color.Gray;

}

}

private void userSurnameField\_Leave(object sender, EventArgs e)

{

if (userSurnameField.Text == "")

{

userSurnameField.Text = "Фамилия";

userSurnameField.ForeColor = Color.Gray;

}

}

private void userSurnameField\_Enter(object sender, EventArgs e)

{

if (userSurnameField.Text == "Фамилия")

{

userSurnameField.Text = "";

userSurnameField.ForeColor = Color.Black;

}

}

private void buttonRegister\_Click(object sender, EventArgs e)

{

if (userNameField.Text == "Имя")

{

MessageBox.Show("Введите имя");

return;

}

if (userSurnameField.Text == "Фамилия")

{

MessageBox.Show("Введите фамилию");

return;

}

if (LoginField.Text == "Логин")

{

MessageBox.Show("Введите логин");

return;

}

if (PassField.Text == "")

{

MessageBox.Show("Введите пароль");

return;

}

if (isUserExists())

return;

DB db = new DB();

MySqlCommand command = new MySqlCommand("INSERT INTO `users` (`login`, `pass`, `name`, `surname`) VALUES (@login, @pass, @name, @surname)", db.GetConnection());

command.Parameters.Add("@login", MySqlDbType.VarChar).Value = LoginField.Text;

command.Parameters.Add("@pass", MySqlDbType.VarChar).Value = PassField.Text;

command.Parameters.Add("@name", MySqlDbType.VarChar).Value = userNameField.Text;

command.Parameters.Add("@surname", MySqlDbType.VarChar).Value = userSurnameField.Text;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Аккаунт был создан");

}

else MessageBox.Show("Аккаунт не был создан");

db.closeConnection();

}

public Boolean isUserExists()

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `users` WHERE `login` = @uL", db.GetConnection());

command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = LoginField.Text;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MessageBox.Show("Такой логин уже есть, введите другой");

return true;

}

else

{

return false;

}

}

private void registerLabel\_Click(object sender, EventArgs e)

{

this.Hide();

LoginForm loginForm = new LoginForm();

loginForm.Show();

}

private void LoginField\_Enter(object sender, EventArgs e)

{

if (LoginField.Text == "Логин")

{

LoginField.Text = "";

LoginField.ForeColor = Color.Black;

}

}

private void LoginField\_Leave(object sender, EventArgs e)

{

if (LoginField.Text == "")

{

LoginField.Text = "Логин";

LoginField.ForeColor = Color.Gray;

}

}

}

}

Листинг 3 – код для главного окна

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TMP\_Kurs

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

LT1.Text = Topics.Tema1;

TxtM1.Text = Topics.Mess1;

LTime1.Text = Topics.Data1;

LLogin1.Text = Topics.Login1;

LT2.Text = Topics.Tema2;

TxtM2.Text = Topics.Mess2;

LTime2.Text = Topics.Data2;

LLogin2.Text = Topics.Login2;

LT3.Text = Topics.Tema3;

TxtM3.Text = Topics.Mess3;

LTime3.Text = Topics.Data3;

LLogin3.Text = Topics.Login3;

LT4.Text = Topics.Tema4;

TxtM4.Text = Topics.Mess4;

LTime4.Text = Topics.Data4;

LLogin4.Text = Topics.Login4;

LT5.Text = Topics.Tema5;

TxtM5.Text = Topics.Mess5;

LTime5.Text = Topics.Data5;

LLogin5.Text = Topics.Login5;

}

private void CloseButton\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void BtnAddTema1\_Click(object sender, EventArgs e)

{

this.Hide();

Topics.id1 = 1;

Topics.id2 = 0;

Topics.id3 = 0;

Topics.id4 = 0;

Topics.id5 = 0;

AddTemaForm addTemaForm = new AddTemaForm();

addTemaForm.Show();

}

private void BtnAddTema2\_Click(object sender, EventArgs e)

{

this.Hide();

Topics.id1 = 0;

Topics.id2 = 1;

Topics.id3 = 0;

Topics.id4 = 0;

Topics.id5 = 0;

AddTemaForm addTemaForm = new AddTemaForm();

addTemaForm.Show();

}

private void LLog\_Click(object sender, EventArgs e)

{

this.Hide();

LoginForm loginForm = new LoginForm();

loginForm.Show();

}

private void BtnAddTema3\_Click(object sender, EventArgs e)

{

this.Hide();

Topics.id1 = 0;

Topics.id2 = 0;

Topics.id3 = 1;

Topics.id4 = 0;

Topics.id5 = 0;

AddTemaForm addTemaForm = new AddTemaForm();

addTemaForm.Show();

}

private void BtnAddTema4\_Click(object sender, EventArgs e)

{

this.Hide();

Topics.id1 = 0;

Topics.id2 = 0;

Topics.id3 = 0;

Topics.id4 = 1;

Topics.id5 = 0;

AddTemaForm addTemaForm = new AddTemaForm();

addTemaForm.Show();

}

private void BtnAddTema5\_Click(object sender, EventArgs e)

{

this.Hide();

Topics.id1 = 0;

Topics.id2 = 0;

Topics.id3 = 0;

Topics.id4 = 0;

Topics.id5 = 1;

AddTemaForm addTemaForm = new AddTemaForm();

addTemaForm.Show();

}

Point lastPoint;

private void LHead\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void LHead\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void BtnDisT1\_Click(object sender, EventArgs e)

{

if (LT1.Text != "Тема" && LT1.Text != "")

{

this.Hide();

Topics.fl = true;

SubTopic.idS1 = 1;

SubTopic.idS2 = 0;

SubTopic.idS3 = 0;

SubTopic.idS4 = 0;

SubTopic.idS5 = 0;

SubtopicForm subtopicForm = new SubtopicForm();

subtopicForm.Show();

}

else MessageBox.Show("Тема не создана!");

}

private void BtnDisT2\_Click(object sender, EventArgs e)

{

if (LT2.Text != "Тема" && LT2.Text != "")

{

this.Hide();

Topics.fl = true;

SubTopic.idS1 = 0;

SubTopic.idS2 = 1;

SubTopic.idS3 = 0;

SubTopic.idS4 = 0;

SubTopic.idS5 = 0;

SubtopicForm subtopicForm = new SubtopicForm();

subtopicForm.Show();

}

else MessageBox.Show("Тема не создана!");

}

private void BtnDisT3\_Click(object sender, EventArgs e)

{

if (LT3.Text != "Тема" && LT3.Text != "")

{

this.Hide();

Topics.fl = true;

SubTopic.idS1 = 0;

SubTopic.idS2 = 0;

SubTopic.idS3 = 1;

SubTopic.idS4 = 0;

SubTopic.idS5 = 0;

SubtopicForm subtopicForm = new SubtopicForm();

subtopicForm.Show();

}

else MessageBox.Show("Тема не создана!");

}

private void BtnDisT4\_Click(object sender, EventArgs e)

{

if (LT4.Text != "Тема" && LT4.Text != "")

{

this.Hide();

Topics.fl = true;

SubTopic.idS1 = 0;

SubTopic.idS2 = 0;

SubTopic.idS3 = 0;

SubTopic.idS4 = 1;

SubTopic.idS5 = 0;

SubtopicForm subtopicForm = new SubtopicForm();

subtopicForm.Show();

}

else MessageBox.Show("Тема не создана!");

}

private void BtnAddTema4\_Click\_1(object sender, EventArgs e)

{

this.Hide();

Topics.id1 = 0;

Topics.id2 = 0;

Topics.id3 = 0;

Topics.id4 = 1;

Topics.id5 = 0;

AddTemaForm addTemaForm = new AddTemaForm();

addTemaForm.Show();

}

private void BtnAddTema5\_Click\_1(object sender, EventArgs e)

{

this.Hide();

Topics.id1 = 0;

Topics.id2 = 0;

Topics.id3 = 0;

Topics.id4 = 0;

Topics.id5 = 1;

AddTemaForm addTemaForm = new AddTemaForm();

addTemaForm.Show();

}

private void BtnDisT5\_Click(object sender, EventArgs e)

{

if (LT5.Text != "Тема" && LT5.Text != "")

{

this.Hide();

Topics.fl = true;

SubTopic.idS1 = 0;

SubTopic.idS2 = 0;

SubTopic.idS3 = 0;

SubTopic.idS4 = 0;

SubTopic.idS5 = 1;

SubtopicForm subtopicForm = new SubtopicForm();

subtopicForm.Show();

}

else MessageBox.Show("Тема не создана!");

}

}

}

Листинг 4 – код для окна обсуждения тем

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TMP\_Kurs

{

public partial class SubtopicForm : Form

{

public SubtopicForm()

{

InitializeComponent();

if (SubTopic.idS1 == 1)

{

LLogin1\_1.Text = SubTopic.Login1\_1;

LT1\_1.Text = SubTopic.Tema1\_1;

LTime1\_1.Text = SubTopic.Data1\_1;

TxtM1\_1.Text = SubTopic.Mess1\_1;

LLogin1\_2.Text = SubTopic.Login1\_2;

LT1\_2.Text = SubTopic.Tema1\_2;

LTime1\_2.Text = SubTopic.Data1\_2;

TxtM1\_2.Text = SubTopic.Mess1\_2;

LLogin1\_3.Text = SubTopic.Login1\_3;

LT1\_3.Text = SubTopic.Tema1\_3;

LTime1\_3.Text = SubTopic.Data1\_3;

TxtM1\_3.Text = SubTopic.Mess1\_3;

LLogin1\_4.Text = SubTopic.Login1\_4;

LT1\_4.Text = SubTopic.Tema1\_4;

LTime1\_4.Text = SubTopic.Data1\_4;

TxtM1\_4.Text = SubTopic.Mess1\_4;

LLogin1\_5.Text = SubTopic.Login1\_5;

LT1\_5.Text = SubTopic.Tema1\_5;

LTime1\_5.Text = SubTopic.Data1\_5;

TxtM1\_5.Text = SubTopic.Mess1\_5;

}

if (SubTopic.idS2 == 1)

{

LLogin1\_1.Text = SubTopic.Login2\_1;

LT1\_1.Text = SubTopic.Tema2\_1;

LTime1\_1.Text = SubTopic.Data2\_1;

TxtM1\_1.Text = SubTopic.Mess2\_1;

LLogin1\_2.Text = SubTopic.Login2\_2;

LT1\_2.Text = SubTopic.Tema2\_2;

LTime1\_2.Text = SubTopic.Data2\_2;

TxtM1\_2.Text = SubTopic.Mess2\_2;

LLogin1\_3.Text = SubTopic.Login2\_3;

LT1\_3.Text = SubTopic.Tema2\_3;

LTime1\_3.Text = SubTopic.Data2\_3;

TxtM1\_3.Text = SubTopic.Mess2\_3;

LLogin1\_4.Text = SubTopic.Login2\_4;

LT1\_4.Text = SubTopic.Tema2\_4;

LTime1\_4.Text = SubTopic.Data2\_4;

TxtM1\_4.Text = SubTopic.Mess2\_4;

LLogin1\_5.Text = SubTopic.Login2\_5;

LT1\_5.Text = SubTopic.Tema2\_5;

LTime1\_5.Text = SubTopic.Data2\_5;

TxtM1\_5.Text = SubTopic.Mess2\_5;

}

if (SubTopic.idS3 == 1)

{

LLogin1\_1.Text = SubTopic.Login3\_1;

LT1\_1.Text = SubTopic.Tema3\_1;

LTime1\_1.Text = SubTopic.Data3\_1;

TxtM1\_1.Text = SubTopic.Mess3\_1;

LLogin1\_2.Text = SubTopic.Login3\_2;

LT1\_2.Text = SubTopic.Tema3\_2;

LTime1\_2.Text = SubTopic.Data3\_2;

TxtM1\_2.Text = SubTopic.Mess3\_2;

LLogin1\_3.Text = SubTopic.Login3\_3;

LT1\_3.Text = SubTopic.Tema3\_3;

LTime1\_3.Text = SubTopic.Data3\_3;

TxtM1\_3.Text = SubTopic.Mess3\_3;

LLogin1\_4.Text = SubTopic.Login3\_4;

LT1\_4.Text = SubTopic.Tema3\_4;

LTime1\_4.Text = SubTopic.Data3\_4;

TxtM1\_4.Text = SubTopic.Mess3\_4;

LLogin1\_5.Text = SubTopic.Login3\_5;

LT1\_5.Text = SubTopic.Tema3\_5;

LTime1\_5.Text = SubTopic.Data3\_5;

TxtM1\_5.Text = SubTopic.Mess3\_5;

}

if (SubTopic.idS4 == 1)

{

LLogin1\_1.Text = SubTopic.Login4\_1;

LT1\_1.Text = SubTopic.Tema4\_1;

LTime1\_1.Text = SubTopic.Data4\_1;

TxtM1\_1.Text = SubTopic.Mess4\_1;

LLogin1\_2.Text = SubTopic.Login4\_2;

LT1\_2.Text = SubTopic.Tema4\_2;

LTime1\_2.Text = SubTopic.Data4\_2;

TxtM1\_2.Text = SubTopic.Mess4\_2;

LLogin1\_3.Text = SubTopic.Login4\_3;

LT1\_3.Text = SubTopic.Tema4\_3;

LTime1\_3.Text = SubTopic.Data4\_3;

TxtM1\_3.Text = SubTopic.Mess4\_3;

LLogin1\_4.Text = SubTopic.Login4\_4;

LT1\_4.Text = SubTopic.Tema4\_4;

LTime1\_4.Text = SubTopic.Data4\_4;

TxtM1\_4.Text = SubTopic.Mess4\_4;

LLogin1\_5.Text = SubTopic.Login4\_5;

LT1\_5.Text = SubTopic.Tema4\_5;

LTime1\_5.Text = SubTopic.Data4\_5;

TxtM1\_5.Text = SubTopic.Mess4\_5;

}

if (SubTopic.idS5 == 1)

{

LLogin1\_1.Text = SubTopic.Login5\_1;

LT1\_1.Text = SubTopic.Tema5\_1;

LTime1\_1.Text = SubTopic.Data5\_1;

TxtM1\_1.Text = SubTopic.Mess5\_1;

LLogin1\_2.Text = SubTopic.Login5\_2;

LT1\_2.Text = SubTopic.Tema5\_2;

LTime1\_2.Text = SubTopic.Data5\_2;

TxtM1\_2.Text = SubTopic.Mess5\_2;

LLogin1\_3.Text = SubTopic.Login5\_3;

LT1\_3.Text = SubTopic.Tema5\_3;

LTime1\_3.Text = SubTopic.Data5\_3;

TxtM1\_3.Text = SubTopic.Mess5\_3;

LLogin1\_4.Text = SubTopic.Login5\_4;

LT1\_4.Text = SubTopic.Tema5\_4;

LTime1\_4.Text = SubTopic.Data5\_4;

TxtM1\_4.Text = SubTopic.Mess5\_4;

LLogin1\_5.Text = SubTopic.Login5\_5;

LT1\_5.Text = SubTopic.Tema5\_5;

LTime1\_5.Text = SubTopic.Data5\_5;

TxtM1\_5.Text = SubTopic.Mess5\_5;

}

}

private void BtnAddTema1\_1\_Click(object sender, EventArgs e)

{

if (Topics.Tema1 != "Тема" || Topics.Tema1 != "")

{

this.Hide();

//Topics.fl = true;

AddTemaForm addTemaForm = new AddTemaForm();

addTemaForm.Show();

}

else MessageBox.Show("Тема не создана!");

}

private void BtnSub1\_1\_Click(object sender, EventArgs e)

{

if (SubTopic.idS1 == 1)

{

SubTopic.Tema1\_1 = Topics.Tema1;

SubTopic.Login1\_1 = Topics.Login;

SubTopic.Data1\_1 = Convert.ToString(DateTime.Now);

SubTopic.Mess1\_1 = TxtM1\_1.Text;

LLogin1\_1.Text = SubTopic.Login1\_1;

LT1\_1.Text = SubTopic.Tema1\_1;

LTime1\_1.Text = SubTopic.Data1\_1;

TxtM1\_1.Text = SubTopic.Mess1\_1;

}

if (SubTopic.idS2 == 1)

{

SubTopic.Tema2\_1 = Topics.Tema2;

SubTopic.Login2\_1 = Topics.Login;

SubTopic.Data2\_1 = Convert.ToString(DateTime.Now);

SubTopic.Mess2\_1 = TxtM1\_1.Text;

LLogin1\_1.Text = SubTopic.Login2\_1;

LT1\_1.Text = SubTopic.Tema2\_1;

LTime1\_1.Text = SubTopic.Data2\_1;

TxtM1\_1.Text = SubTopic.Mess2\_1;

}

if (SubTopic.idS3 == 1)

{

SubTopic.Tema3\_1 = Topics.Tema3;

SubTopic.Login3\_1 = Topics.Login;

SubTopic.Data3\_1 = Convert.ToString(DateTime.Now);

SubTopic.Mess3\_1 = TxtM1\_1.Text;

LLogin1\_1.Text = SubTopic.Login3\_1;

LT1\_1.Text = SubTopic.Tema3\_1;

LTime1\_1.Text = SubTopic.Data3\_1;

TxtM1\_1.Text = SubTopic.Mess3\_1;

}

if (SubTopic.idS4 == 1)

{

SubTopic.Tema4\_1 = Topics.Tema4;

SubTopic.Login4\_1 = Topics.Login;

SubTopic.Data4\_1 = Convert.ToString(DateTime.Now);

SubTopic.Mess4\_1 = TxtM1\_1.Text;

LLogin1\_1.Text = SubTopic.Login4\_1;

LT1\_1.Text = SubTopic.Tema4\_1;

LTime1\_1.Text = SubTopic.Data4\_1;

TxtM1\_1.Text = SubTopic.Mess4\_1;

}

if (SubTopic.idS5 == 1)

{

SubTopic.Tema5\_1 = Topics.Tema5;

SubTopic.Login5\_1 = Topics.Login;

SubTopic.Data5\_1 = Convert.ToString(DateTime.Now);

SubTopic.Mess5\_1 = TxtM1\_1.Text;

LLogin1\_1.Text = SubTopic.Login5\_1;

LT1\_1.Text = SubTopic.Tema5\_1;

LTime1\_1.Text = SubTopic.Data5\_1;

TxtM1\_1.Text = SubTopic.Mess5\_1;

}

}

private void CloseButton\_Click(object sender, EventArgs e)

{

this.Hide();

MainForm mainForm = new MainForm();

mainForm.Show();

}

private void BtnSub1\_2\_Click(object sender, EventArgs e)

{

if (SubTopic.idS1 == 1)

{

SubTopic.Tema1\_2 = Topics.Tema1;

SubTopic.Login1\_2 = Topics.Login;

SubTopic.Data1\_2 = Convert.ToString(DateTime.Now);

SubTopic.Mess1\_2 = TxtM1\_2.Text;

LLogin1\_2.Text = SubTopic.Login1\_2;

LT1\_2.Text = SubTopic.Tema1\_2;

LTime1\_2.Text = SubTopic.Data1\_2;

TxtM1\_2.Text = SubTopic.Mess1\_2;

}

if (SubTopic.idS2 == 1)

{

SubTopic.Tema2\_2 = Topics.Tema2;

SubTopic.Login2\_2 = Topics.Login;

SubTopic.Data2\_2 = Convert.ToString(DateTime.Now);

SubTopic.Mess2\_2 = TxtM1\_2.Text;

LLogin1\_2.Text = SubTopic.Login2\_2;

LT1\_2.Text = SubTopic.Tema2\_2;

LTime1\_2.Text = SubTopic.Data2\_2;

TxtM1\_2.Text = SubTopic.Mess2\_2;

}

if (SubTopic.idS3 == 1)

{

SubTopic.Tema3\_2 = Topics.Tema3;

SubTopic.Login3\_2 = Topics.Login;

SubTopic.Data3\_2 = Convert.ToString(DateTime.Now);

SubTopic.Mess3\_2 = TxtM1\_2.Text;

LLogin1\_2.Text = SubTopic.Login3\_2;

LT1\_2.Text = SubTopic.Tema3\_2;

LTime1\_2.Text = SubTopic.Data3\_2;

TxtM1\_2.Text = SubTopic.Mess3\_2;

}

if (SubTopic.idS4 == 1)

{

SubTopic.Tema4\_2 = Topics.Tema4;

SubTopic.Login4\_2 = Topics.Login;

SubTopic.Data4\_2 = Convert.ToString(DateTime.Now);

SubTopic.Mess4\_2 = TxtM1\_2.Text;

LLogin1\_2.Text = SubTopic.Login4\_2;

LT1\_2.Text = SubTopic.Tema4\_2;

LTime1\_2.Text = SubTopic.Data4\_2;

TxtM1\_2.Text = SubTopic.Mess4\_2;

}

if (SubTopic.idS5 == 1)

{

SubTopic.Tema5\_2 = Topics.Tema5;

SubTopic.Login5\_2 = Topics.Login;

SubTopic.Data5\_2 = Convert.ToString(DateTime.Now);

SubTopic.Mess5\_2 = TxtM1\_2.Text;

LLogin1\_2.Text = SubTopic.Login5\_2;

LT1\_2.Text = SubTopic.Tema5\_2;

LTime1\_2.Text = SubTopic.Data5\_2;

TxtM1\_2.Text = SubTopic.Mess5\_2;

}

}

private void BtnSub1\_3\_Click(object sender, EventArgs e)

{

if (SubTopic.idS1 == 1)

{

SubTopic.Tema1\_3 = Topics.Tema1;

SubTopic.Login1\_3 = Topics.Login;

SubTopic.Data1\_3 = Convert.ToString(DateTime.Now);

SubTopic.Mess1\_3 = TxtM1\_3.Text;

LLogin1\_3.Text = SubTopic.Login1\_3;

LT1\_3.Text = SubTopic.Tema1\_3;

LTime1\_3.Text = SubTopic.Data1\_3;

TxtM1\_3.Text = SubTopic.Mess1\_3;

}

if (SubTopic.idS2 == 1)

{

SubTopic.Tema2\_3 = Topics.Tema2;

SubTopic.Login2\_3 = Topics.Login;

SubTopic.Data2\_3 = Convert.ToString(DateTime.Now);

SubTopic.Mess2\_3 = TxtM1\_3.Text;

LLogin1\_3.Text = SubTopic.Login2\_3;

LT1\_3.Text = SubTopic.Tema2\_3;

LTime1\_3.Text = SubTopic.Data2\_3;

TxtM1\_3.Text = SubTopic.Mess2\_3;

}

if (SubTopic.idS3 == 1)

{

SubTopic.Tema3\_3 = Topics.Tema3;

SubTopic.Login3\_3 = Topics.Login;

SubTopic.Data3\_3 = Convert.ToString(DateTime.Now);

SubTopic.Mess3\_3 = TxtM1\_3.Text;

LLogin1\_3.Text = SubTopic.Login3\_3;

LT1\_3.Text = SubTopic.Tema3\_3;

LTime1\_3.Text = SubTopic.Data3\_3;

TxtM1\_3.Text = SubTopic.Mess3\_3;

}

if (SubTopic.idS4 == 1)

{

SubTopic.Tema4\_3 = Topics.Tema4;

SubTopic.Login4\_3 = Topics.Login;

SubTopic.Data4\_3 = Convert.ToString(DateTime.Now);

SubTopic.Mess4\_3 = TxtM1\_3.Text;

LLogin1\_3.Text = SubTopic.Login4\_3;

LT1\_3.Text = SubTopic.Tema4\_3;

LTime1\_3.Text = SubTopic.Data4\_3;

TxtM1\_3.Text = SubTopic.Mess4\_3;

}

if (SubTopic.idS5 == 1)

{

SubTopic.Tema5\_3 = Topics.Tema5;

SubTopic.Login5\_3 = Topics.Login;

SubTopic.Data5\_3 = Convert.ToString(DateTime.Now);

SubTopic.Mess5\_3 = TxtM1\_3.Text;

LLogin1\_3.Text = SubTopic.Login5\_3;

LT1\_3.Text = SubTopic.Tema5\_3;

LTime1\_3.Text = SubTopic.Data5\_3;

TxtM1\_3.Text = SubTopic.Mess5\_3;

}

}

private void BtnSub1\_4\_Click(object sender, EventArgs e)

{

if (SubTopic.idS1 == 1)

{

SubTopic.Tema1\_4 = Topics.Tema1;

SubTopic.Login1\_4 = Topics.Login;

SubTopic.Data1\_4 = Convert.ToString(DateTime.Now);

SubTopic.Mess1\_4 = TxtM1\_4.Text;

LLogin1\_4.Text = SubTopic.Login1\_4;

LT1\_4.Text = SubTopic.Tema1\_4;

LTime1\_4.Text = SubTopic.Data1\_4;

TxtM1\_4.Text = SubTopic.Mess1\_4;

}

if (SubTopic.idS2 == 1)

{

SubTopic.Tema2\_4 = Topics.Tema2;

SubTopic.Login2\_4 = Topics.Login;

SubTopic.Data2\_4 = Convert.ToString(DateTime.Now);

SubTopic.Mess2\_4 = TxtM1\_4.Text;

LLogin1\_4.Text = SubTopic.Login2\_4;

LT1\_4.Text = SubTopic.Tema2\_4;

LTime1\_4.Text = SubTopic.Data2\_4;

TxtM1\_4.Text = SubTopic.Mess2\_4;

}

if (SubTopic.idS3 == 1)

{

SubTopic.Tema3\_4 = Topics.Tema3;

SubTopic.Login3\_4 = Topics.Login;

SubTopic.Data3\_4 = Convert.ToString(DateTime.Now);

SubTopic.Mess3\_4 = TxtM1\_4.Text;

LLogin1\_4.Text = SubTopic.Login3\_4;

LT1\_4.Text = SubTopic.Tema3\_4;

LTime1\_4.Text = SubTopic.Data3\_4;

TxtM1\_4.Text = SubTopic.Mess3\_4;

}

if (SubTopic.idS4 == 1)

{

SubTopic.Tema4\_4 = Topics.Tema4;

SubTopic.Login4\_4 = Topics.Login;

SubTopic.Data4\_4 = Convert.ToString(DateTime.Now);

SubTopic.Mess4\_4 = TxtM1\_4.Text;

LLogin1\_4.Text = SubTopic.Login4\_4;

LT1\_4.Text = SubTopic.Tema4\_4;

LTime1\_4.Text = SubTopic.Data4\_4;

TxtM1\_4.Text = SubTopic.Mess4\_4;

}

if (SubTopic.idS5 == 1)

{

SubTopic.Tema5\_4 = Topics.Tema5;

SubTopic.Login5\_4 = Topics.Login;

SubTopic.Data5\_4 = Convert.ToString(DateTime.Now);

SubTopic.Mess5\_4 = TxtM1\_4.Text;

LLogin1\_4.Text = SubTopic.Login5\_4;

LT1\_4.Text = SubTopic.Tema5\_4;

LTime1\_4.Text = SubTopic.Data5\_4;

TxtM1\_4.Text = SubTopic.Mess5\_4;

}

}

private void BtnSub1\_5\_Click(object sender, EventArgs e)

{

if (SubTopic.idS1 == 1)

{

SubTopic.Tema1\_5 = Topics.Tema1;

SubTopic.Login1\_5 = Topics.Login;

SubTopic.Data1\_5 = Convert.ToString(DateTime.Now);

SubTopic.Mess1\_5 = TxtM1\_5.Text;

LLogin1\_5.Text = SubTopic.Login1\_5;

LT1\_5.Text = SubTopic.Tema1\_5;

LTime1\_5.Text = SubTopic.Data1\_5;

TxtM1\_5.Text = SubTopic.Mess1\_5;

}

if (SubTopic.idS2 == 1)

{

SubTopic.Tema2\_5 = Topics.Tema2;

SubTopic.Login2\_5 = Topics.Login;

SubTopic.Data2\_5 = Convert.ToString(DateTime.Now);

SubTopic.Mess2\_5 = TxtM1\_5.Text;

LLogin1\_5.Text = SubTopic.Login2\_5;

LT1\_5.Text = SubTopic.Tema2\_5;

LTime1\_5.Text = SubTopic.Data2\_5;

TxtM1\_5.Text = SubTopic.Mess2\_5;

}

if (SubTopic.idS3 == 1)

{

SubTopic.Tema3\_5 = Topics.Tema3;

SubTopic.Login3\_5 = Topics.Login;

SubTopic.Data3\_5 = Convert.ToString(DateTime.Now);

SubTopic.Mess3\_5 = TxtM1\_5.Text;

LLogin1\_5.Text = SubTopic.Login3\_5;

LT1\_5.Text = SubTopic.Tema3\_5;

LTime1\_5.Text = SubTopic.Data3\_5;

TxtM1\_5.Text = SubTopic.Mess3\_5;

}

if (SubTopic.idS4 == 1)

{

SubTopic.Tema4\_5 = Topics.Tema4;

SubTopic.Login4\_5 = Topics.Login;

SubTopic.Data4\_5 = Convert.ToString(DateTime.Now);

SubTopic.Mess4\_5 = TxtM1\_5.Text;

LLogin1\_5.Text = SubTopic.Login4\_5;

LT1\_5.Text = SubTopic.Tema4\_5;

LTime1\_5.Text = SubTopic.Data4\_5;

TxtM1\_5.Text = SubTopic.Mess4\_5;

}

if (SubTopic.idS5 == 1)

{

SubTopic.Tema5\_5 = Topics.Tema5;

SubTopic.Login5\_5 = Topics.Login;

SubTopic.Data5\_5 = Convert.ToString(DateTime.Now);

SubTopic.Mess5\_5 = TxtM1\_5.Text;

LLogin1\_5.Text = SubTopic.Login5\_5;

LT1\_5.Text = SubTopic.Tema5\_5;

LTime1\_5.Text = SubTopic.Data5\_5;

TxtM1\_5.Text = SubTopic.Mess5\_5;

}

}

Point lastPoint;

private void LHead\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void LHead\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

}

}

Листинг 5 – код для окна добавления тем

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TMP\_Kurs

{

public partial class AddTemaForm : Form

{

public AddTemaForm()

{

InitializeComponent();

}

private void BtnAT\_Click(object sender, EventArgs e)

{

if (Topics.id1 == 1)

{

Topics.Tema1 = TxtAddTem.Text;

Topics.Mess1 = TxtAddM.Text;

Topics.Data1 = Convert.ToString(DateTime.Now);

Topics.Login1 = Topics.Login;

this.Hide();

MainForm mainF = new MainForm();

mainF.Show();

}

if (Topics.id2 == 1)

{

Topics.Tema2 = TxtAddTem.Text;

Topics.Mess2 = TxtAddM.Text;

Topics.Data2 = Convert.ToString(DateTime.Now);

Topics.Login2 = Topics.Login;

this.Hide();

MainForm mainF = new MainForm();

mainF.Show();

}

if (Topics.id3 == 1)

{

Topics.Tema3 = TxtAddTem.Text;

Topics.Mess3 = TxtAddM.Text;

Topics.Data3 = Convert.ToString(DateTime.Now);

Topics.Login3 = Topics.Login;

this.Hide();

MainForm mainF = new MainForm();

mainF.Show();

}

if (Topics.id4 == 1)

{

Topics.Tema4 = TxtAddTem.Text;

Topics.Mess4 = TxtAddM.Text;

Topics.Data4 = Convert.ToString(DateTime.Now);

Topics.Login4 = Topics.Login;

this.Hide();

MainForm mainF = new MainForm();

mainF.Show();

}

if (Topics.id5 == 1)

{

Topics.Tema5 = TxtAddTem.Text;

Topics.Mess5 = TxtAddM.Text;

Topics.Data5 = Convert.ToString(DateTime.Now);

Topics.Login5 = Topics.Login;

this.Hide();

MainForm mainF = new MainForm();

mainF.Show();

}

}

private void CloseButton\_Click(object sender, EventArgs e)

{

// if (Topics.fl == false)

// {

this.Hide();

MainForm mainForm = new MainForm();

mainForm.Show();

//}

//else {

// this.Hide();

// SubtopicForm subtopicForm = new SubtopicForm();

// subtopicForm.Show();

//}

}

Point lastPoint;

private void AddTemaForm\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void AddTemaForm\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

}

}