НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

РАЗРАБОТКА И ИСПОЛЬЗОВАНИЕ ЭЛЕКТРОННОГО ОБРАЗОВАТЕЛЬНОГО РЕСУРСА НА УРОКАХ ИНФОРМАТИКИ ПРИ РЕШЕНИИ ЗАДАЧ НА ХРАНЕНИЕ И ПЕРЕДАЧУ ИНФОРМАЦИИ

Сагайдак Игорь Владимирович

Научный руководитель: канд. физ.-мат. наук, доцент Роман Геннадьевич Письменный

Факультет математики, информатики, биологии и технологии, Федеральное государственное бюджетное образовательное учреждение высшего образования «Кубанский государственный университет» филиал в г. Славянске-на-Кубани

г. Славянск-на-Кубани

Российская Федерация

Аннотация

Курсовая работа посвящена изучению проблемы использования электронных образовательных ресурсов на уроках информатики при решении учащимися текстовых задач. Актуальность данной темы обусловлена противоречием между необходимостью использования на занятиях электронных образовательных ресурсов для обеспечения возможности отработки учащимися навыков решения текстовых задач с учетом требований ОГЭ, с одной стороны, и ограниченностью однотипных задач в открытых источниках, с другой стороны.

Разрешение данного противоречия определило цели и задачи, а также методологический аппарат исследования.

Объект исследования: электронные образовательные ресурсы в курсе информатики основной школы.

Предмет исследования: использование электронных образовательных ресурсов при решении текстовых задач на хранение и передачу информации на уроках информатики основной школы.

Цель исследования: разработать электронный образовательный ресурс для решения текстовых задач на хранение и передачу информации на уроках информатики основной школы.

Для достижения поставленной цели необходимо решить следующие задачи:

- изучить теоретические аспекты использования электронных образовательных ресурсов на уроках информатики;
- рассмотреть текстовые задачи в курсе информатики основной школы;
- разработать авторский электронный образовательный ресурс для решения текстовых задач на хранение и передачу информации;
- описать методические особенности использования авторского электронного образовательного ресурса при решении текстовых задач на

хранение и передачу информации по информатике.

В результате проведенного исследования был разработан электронный образовательный ресурс для генерации текстовых задач по информатике и теоретически обоснованы его образовательные возможности.

Ключевые слова: электронный образовательный ресурс, текстовые задачи, методика преподавания информатики, разработка, Python, подготовка к ОГЭ, подготовка к ЕГЭ.

ВВЕДЕНИЕ

Ведущей тенденцией современности является все более глубокое проникновение цифровизации и информатизации в различные области жизни нынешнего общества. Понятия «цифровая экономика», «цифровая экосистема», «цифровая среда», «цифровая трансформация», «цифровая грамотность» прочно вошли в оборот.

На этом фоне немаловажным понятием, закрепившимся в образовании является «цифровая образовательная среда», предполагающим необходимость включения в этот процесс подготовку подрастающего поколения к возможностям информационного общества. Согласно современным исследованиям и нормативным документам, основу цифровой образовательной среды должны составлять электронные образовательные ресурсы.

Изучению образовательных возможностей электронных образовательных ресурсов посвящены работы Е. В. Данильчук, Е. В. Дорониной, Н. Ю. Куликовой, С. И. Марковой, А. В. Осина, И. Н. Смирновой, Г. А. Федоровой и др. По мнению большинства исследователей, электронные образовательные ресурсы, выполняя вспомогательную роль в рамках изучения большинства предметов, выступают как средство обучения.

Однако при преподавании школьного курса информатики, электронные образовательные ресурсы выступают как основа формирования информационной культуры учащихся. В этой связи важным аспектом использования электронных образовательных ресурсов является возможность их использования для решения текстовых задач по информатике.

Вопросами изучения сущности и особенностей текстовых задач занимались С. С. Белинский, Г. Х. Воистинова [13], З. И. Исаева [14], Х. Г. Хайитова [12] и др.

В настоящее время в сети Интернет существует огромное количество

готовых ресурсов в рамках коллекций созданных при поддержке различных федеральных программ. Однако предлагаемые там материалы не могут считаться универсальными и быть модифицированы самим педагогом применительно к решаемой учебной ситуации, что несколько снижает эффективность их применения.

В этой связи актуален вопрос самостоятельной разработке педагогом собственных ЭОР. Интересные подходы в этом направлении представлены в работах Н. Ю. Куликовой, С. Ю. Сердюковой [8], Е. Л. Склейнова, Н. А. Тарасова.

Указанные аспекты проблемы использования образовательных возможностей электронных образовательных ресурсов при изучении школьного курса информатики обусловили выбор темы исследования: «Разработка и использование электронного образовательного ресурса на уроках информатики при решении задач на хранение и передачу информации».

Объект исследования: электронные образовательные ресурсы в курсе информатики основной школы.

Предмет исследования: использование электронных образовательных ресурсов при решении текстовых задач на хранение и передачу информации на уроках информатики основной школы.

Цель исследования: разработать электронный образовательный ресурс для решения текстовых задач на хранение и передачу информации на уроках информатики основной школы.

Для достижения поставленной цели необходимо решить следующие задачи:

– изучить теоретические аспекты использования электронных образовательных ресурсов на уроках информатики;

– рассмотреть текстовые задачи в курсе информатики основной школы;

– разработать авторский электронный образовательный ресурс для решения текстовых задач на хранение и передачу информации;

– описать методические особенности использования авторского электронного образовательного ресурса при решении текстовых задач на хранение и передачу информации по информатике.

Методы исследования:

– методы анализа, сравнения и обобщения нормативной, методической и научной литературы по проблеме исследования;

– методы проектирования при разработке программ.

Курсовая работа состоит из введения, двух разделов, заключения и списка использованных источников. В первом разделе представлены теоретические аспекты использования электронных образовательных ресурсов на уроках информатики: проведен анализ образовательной роли электронных образовательных ресурсов, рассмотрены текстовые задачи в курсе информатики основной школы. Во втором - представлены методические особенности использования авторского электронного образовательного ресурса при решении текстовых задач по информатике: разработан авторский электронный образовательный ресурс для решения текстовых задач и описаны методические особенности использования авторского электронного образовательного ресурса при решении текстовых задач на хранение и передачу информации по информатике.

1. Теоретические аспекты использования электронных образовательных ресурсов на уроках информатики

1.1 Образовательная роль электронных образовательных ресурсов

Ведущей тенденцией современности является все более глубокое проникновение цифровизации и информатизации в различные области жизни нынешнего общества. Социальные вызовы определяют и необходимость включения в этот процесс подготовку подрастающего поколения к возможностям информационного общества.

На государственном уровне регламентируется необходимость дополнения традиционного процесса обучения использованием электронных образовательных ресурсов, как необходимой составляющей формирования информационной культуры и информационной компетентности как педагогов, так и учащихся. В этой связи вопрос о сущности и особенностях разработки и использования электронных образовательных ресурсов (ЭОР) является одним из актуальных.

Понятие «электронный образовательный ресурс» определяется в ГОСТ Р 52653-2006, в статье 12 подраздел 3.5 как образовательный ресурс, представленный в электронно-цифровой форме и включающий в себя структуру, предметное содержание и метаданные о них. Данное определение является очень обобщенным, в связи с чем, различные исследователи этого феномена стараются более детально представить его сущность с целью понимания возможностей его методического использования.

В работах А. В. Осина, Е. В. Дорониной и С. И. Марковой, ЭОР рассматривается как средство обучения. При этом А.В. Осин описывает ЭОР с технологической точки зрения, как «совокупность средств программного, информационного, технического и организационного обеспечения, электронных изданий, размещаемая на машиночитаемых носителях и ли в сети» [1].

С. И. Маркова рассматривает ЭОР с методической позиции как

«комплексное средство обучения, обеспечивающее различные виды учебной деятельности и позволяющее осуществить индивидуально-деятельностный подход к процессу целенаправленного формирования профессиональных компетенций в соответствующей предметной области» [1].

Инструментальный подход к определению сущности понятия позволяет понять его структурную организацию и возможности использования.

Однако в исследовании Е. В. Якушиной ЭОР рассматривается как некоторое «образовательное содержание, облеченное в электронную форму, для воспроизведения которого используются электронные устройства» [2]. Такой подход к определению понятия позволяет автору выделить ряд требований, реализация которых обеспечит эффективность в работе с ЭОР: обеспечение всех компонентов образовательного процесса, интерактивность и обеспечение высокого уровня самостоятельной деятельности учащихся, возможность удаленного полноценного обучения.

Вопросам использования ЭОР в практической педагогической деятельности учителей на различных уроках посвящено достаточно статей современных исследователей, которые рассматривают этот образовательный ресурс с различных позиций. Так, И. Н. Смирнова [3] анализирует возможности использования электронных образовательных ресурсов и с точки зрения формирования на их основе информационной культуры учащихся на уроках. В том же контексте с позиции анализа образовательных возможностей формирования у школьников информационной культуры и системы УУД на уроках информатики, изучаются ЭОР в материалах статьи М. В. Кузнецовой и И. У. Толмачевой [4]. Анализу роли ЭОР в формировании исследовательских навыков на уроках не только информатики, но и физики, и математики посвящена статья О. В. Корпуновой и М. А. Гавриловой [5]. О необходимости методической компетентности в области создания и использования ЭОР в своих статьях заявляют Г. А. Федорова [6], Е. В. Данильчук и Н. Ю. Куликова [7].

Как верно на наш взгляд, отмечено в материалах работы Е. В. Данильчук и Н. Ю. Куликовой «учителя информатики имеют существенный потенциал в самостоятельном использовании различных образовательных онлайн-платформ» [8, с. 10]. Этот факт объясняется более высоким уровнем информационной компетенции специалистов, который закладывается в период профессиональной подготовки в вузе, о чем в своем исследовании говорит Г. А. Федорова, отмечая необходимость и значимость непрерывной методической подготовки бакалавров направления Педагогическое образование профиля «Информатика» к разработке и применению ЭОР.

В связи с тем, что «современное общество характеризуется большими объемами информации, с которой человек должен работать» [3, с. 150], актуальным является вопрос создания и использования ЭОР, которые соответствовали ряду требований: научность содержания, подчиненность формы подачи информации и используемых выразительных средств учебным целям, образовательная технологичность, структурирование содержания по принципу создания образовательной среды, обеспечивающей индивидуальную траекторию обучения каждому пользователю [2, с. 134].

На сегодняшний день в сети Интернет существует огромное количество готовых ресурсов в рамках коллекций созданных при поддержке различных федеральных программ. Все они обеспечивают высокий уровень требований как в части содержания, так и в части качества продуктов. Однако предлагаемые там материалы не могут быть адаптированы для решения каких-то локальных образовательных задач и не могут быть модифицированы самим педагогом применительно к решаемой учебной ситуации, а должно использоваться в готовом неизменном виде, что несколько снижает эффективность их применения.

В этой связи актуален вопрос самостоятельной разработке педагогом собственных ЭОР. Интересные подходы в этом направлении представлены в работах Н. Ю. Куликовой, С. Ю. Сердюковой, Е. Л. Склейнова [8], Н. А. Тарасова [9].

Как отмечается в работе Н. Ю. Куликовой, С. Ю. Сердюковой, Е. Л. Склейнова, разработка авторского ЭОР может вестись в двух направлениях: в поиске новых форм разработки ЭОР, либо в разработке новых интернет-сервисов.

Авторским коллективом проведен анализ сетевых сервисов и ресурсов Интернета для разработки и совместного использования мультимедийного контента, как базы для создания ЭОР (сервисы для создания интерактивных лент времени, диаграмм, постеров, стенгазет; сервисы для создания и совместного использования виртуальных интерактивных досок; коллекции интерактивных упражнений и шаблонов, при помощи которых могут создаваться интерактивные ресурсы) [8, с.99]. Однако, как отмечается в исследовании, данные ресурсы обладают достаточно ограниченным спектром возможностей использования, что не позволяет говорить об их универсальности.

Расширение образовательного потенциала ЭОР возможно, с точки зрения Н. А. Тарасова, при условии внедрения в образовательный процесс ЭОР с открытым исходным кодом. Под «открытым ПО» автор понимает модель, обеспечивающую возможность свободно распространять копии программы с исходным кодом, и дающие возможность изменять и ли использовать ее части в новых открытых разработках [9, с. 127]. В своем исследовании Н. А. Тарасов на основе сравнительного анализа ЭОР с открытым кодом, выделяет три группы ЭОР:

- ЭОР, разработанные на основе систем с открытым исходным кодом (созданные на основе Wordpress, Joomla, Drupal и пр.);

- ЭОР, разработанные полностью самостоятельно, в которых не используются какие-либо системы, кроме инструментов для программирования;

- ЭОР, созданные на основе закрытых систем.

При этом автор отдает предпочтение первому типу ЭОР, поскольку считает его самым оптимальным по соотношению трудозатрат, времени,

потраченного на разработку и конечного результата, обеспечивающего максимальную гибкость в отношении внедрения в процесс обучения.

1.2. Текстовые задачи в курсе информатики основной школы

Изучение информатики в основной школе начинается непосредственно в 5 классе. Изучение курса начинается с введения фундаментального понятия информации, описания характеристик и представления информации. Далее идет введение не менее важных понятий - языка, как способа представления информации и алфавита, на которых основано кодирование информации. Первые текстовые задачи в курсе информатики появляются как раз при изучении кодирования информации. Например, в учебнике Босовой 5 класса параграфа 7 Кодирование информации в секции Вопросы и задания дается следующая формулировка задачи на кодирование:

«Мальчик заменил каждую букву своего имени её номером в алфавите. Получилось 18 21 19 13 1 15. Как зовут мальчика?» [10].

Для решения задачи требуется найти соотношение между номеров буквы в алфавите и самой буквой и подставить вместо цифр нужные буквы. Задача не представляет какой-либо сложности, но закладывает основу для дальнейшего изучения темы.

В научно-методической литературе понятие текстовой задачи определяется как «описание некоторой ситуации (ситуаций) на естественном языке с требованием дать количественную характеристику какого-либо компонента этой ситуации, установить наличие или отсутствие некоторого отношения между ее компонентами или определить вид этого отношения» [11, с. 60-61].

Как отмечается в исследовании Х. Г. Хайитовой, в структуре текстовой задачи выделяется три части:

– условие задачи – это вся информация об известных и неизвестных количественных величинах, характеризующих исследуемую ситуацию, и

количественных соотношениях между ними;

– требования задачи – поиск количественных отношений в состоянии задачи;

– оператор задачи – это набор действий, выполняемых в отношении количественного отношения условия выполнения требования задачи [12, с. 1].

Описывая третий компонент текстовой задачи, Г. Х. Воистинова и Д. Х. Рахматуллина [13, с. 2], со ссылкой на исследование Л. М. Фридмана и Е. Н. Турецкого выделяют несколько видов действий, необходимых для решения текстовой задачи:

– знакомство с условием задачи;

– выполнение сжатой записи условия задачи;

– формирование плана решения задачи;

– выбор модели задачи (рисунок, чертеж, таблица);

– запись решения, исходя из выбранного плана;

– формирование ответа задачи;

– проверка решения.

Рассматривая структурный состав текстовых задач как совокупность условия, обосновывая (теоретического или практического), решения и заключения, З. И. Исаева выделяет четыре типа текстовых задач:

стандартные – все компоненты известны, что позволяет учителю оценить как учащиеся поняли материал;

обучающие – неизвестен один из четырех компонентов, что позволяет учителю создавать огромное количество вариаций;

поисковые – неизвестны два компонента, что позволяет активизировать мыслительную деятельность учащихся и усложнить задание;

проблемные – где неизвестны три компонента [14, с. 83].

Курс информатики основной школы содержит достаточно большое количество разнообразных текстовых задач. Помимо приведенной выше

задачи на кодирование информации для 5-го класса в старших классах рассматриваются и более сложные их варианты. Кроме того, существуют также и другие типы текстовых задач, в том числе:

- задачи на поиск информационного веса алфавита ($N=2^i$);
- задачи на поиск информационного объема сообщения ($I=K^i$);
- задачи на файлы и файловую систему;
- задачи на глубину цвета ($N=2^i$);
- задачи на основы алгебры логики (логические задачи);
- задачи на основные алгоритмические конструкции.

Однако учитывая специфику государственной итоговой аттестации, в ОГЭ не могут быть включены все типы описанных выше задач. Задачи, которые лягут в основу нашего электронного образовательного ресурса целесообразно выбирать из заданий, представленных в ОГЭ этого кода. Кроме того, этот тип задач должен просто поддаваться алгоритмизации на одном из языков программирования. В противном случае, время, затраченное на разработку сервиса, будет излишне высоким и рациональнее составлять такие задачи самостоятельно.

Отдельно рассмотрим текстовые задачи, связанные с хранением и передачей информации. Ярким примером задачи на хранение информации является задача следующего вида:

Какой минимальный объём памяти (в Кбайт) нужно зарезервировать, чтобы можно было сохранить любое растровое изображение размером 128x128 пикселей при условии, что в изображении могут использоваться 256 различных цветов? В ответе запишите только целое число, единицу измерения писать не нужно.

Данный тип задач решается с помощью описанной выше формулы ($N=2^i$). Кроме того легко поддается шаблонизации. Шаблон этого типа задач можно представить следующим образом:

Какой минимальный объём памяти (в Кбайт) нужно зарезервировать,

чтобы можно было сохранить любое растровое изображение размером <Количество пикселей>x<Количество пикселей> пикселей при условии, что в изображении могут использоваться <Количество цветов> различных цветов? В ответе запишите только целое число, единицу измерения писать не нужно.

Далее рассмотрим задачу на передачу информации. В качестве примере рассмотрим задачу именно на передачу звуковой информации, шаблон которой представлен в следующем виде:

Файл размером <Размер> Кбайт передаётся через некоторое соединение со скоростью <Скорость X> бит в секунду. Определите размер файла (в Кбайт), который можно передать за то же время через другое соединение со скоростью <Скорость Y> бит в секунду. В ответе укажите одно число — размер файла в Кбайт. Единицы измерения писать не нужно.

Не трудно заметить, что описанные задачи легко ложатся в концепцию генерации задач.

В ОГЭ представлены 12 заданий тестовой части (номера 1–12) и 3 задания (номера 13–15) развернутой части. Среди них, текстовыми задачами являются номера:

1. Количественные параметры информационных объектов;

5. Простой линейный алгоритм для формального исполнителя;

7. Информационно-коммуникационные технологии.

Рассмотрим каждый из номеров отдельно. Задание 1 представлено тремя типами задач. Задачи первого типа требуют от ученика по кодировке и разнице размера текста (в байтах) определить вычеркнутое слово.

Типичным примером такого вида задач, будет задача соответствующая следующему шаблону:

В одной из кодировок <Название кодировки> каждый символ кодируется <Количество бит> битами. <Имя ученика> написал(а) текст (в нём нет лишних пробелов):

«<Слово 1>, <Слово 2>, … <Слово N>, - <Фраза>».

Ученик вычеркнул из списка одно слово. Заодно он вычеркнул ставшие лишними запятые и пробелы — два пробела не должны идти подряд.

При этом размер нового предложения в данной кодировке оказался на <Количество байт> байт меньше, чем размер исходного предложения. Напишите в ответе вычеркнутое слово.

Задачи второго типа проверяют умение высчитывать информационный объем текста и имеют следующий шаблон:

Статья, набранная на компьютере, содержит <X> страниц, на каждой странице <Y> строки, в каждой строке <Z> символов. В одном из представлений <Название кодировки> каждый символ кодируется <K> битами. Определите информационный объём статьи в Кбайтах в этом варианте представления <Название кодировки>.

И, наконец, задания третьего типа требуют от ученика посчитать количество символов в предложении и, с учетом заданной кодировки, вычислить информационный объем. Шаблон данного задания самый простой:

В одной из кодировок <Название кодировки> каждый символ кодируется <Количество бит> битами. Определите размер в байтах следующего предложения в данной кодировке: <Предложение из нескольких слов>

Поскольку решение задания сводится к простым арифметическим действиям и все три типа имеют свой конкретный шаблон - реализовать автоматическую генерацию таких задач не составит больших проблем.

Задание 5 связано с анализом работы программы исполнителя.

Приведем пример такого типа задания:

У исполнителя Альфа две команды, которым присвоены номера:

1. прибавь 1;

2. умножь на b

(b - неизвестное натуральное число; b>=2).

Выполняя первую из них, Альфа увеличивает число на экране на 1, а

выполняя вторую, умножает это число на b. Программа для исполнителя Альфа - это последовательность номеров команд. Известно, что программа 11211 переводит число 6 в число 82. Определите значение b. [15]

Решить эту задачу можно и с помощью языка программирования. Например, решением этой задачи будет следующий код на языке Python:

```
def find_b() –> int:
  for b in range(2, 100):
    x = 6 + 1 + 1
    if (x * b + 1 + 1) == 82:
      return b
```

Для этой задачи так же возможно построить шаблон, однако построить правильный алгоритм генерации таких заданий - нетривиальная задача. Например, если взять это же условие, но заменить 82 на 84 или на 32 - задача не будет иметь целых решений, а значит брать совершенно случайные числа - нельзя. В итоге алгоритм составления таких задач возможно, но нерационально долго.

Смысл задачи 7 состоит в построении абсолютного пути к файлу на удаленном сервере и дальнейшее представление этого адреса в качестве буквенной последовательности. Пример задания с платформы решу ОГЭ:

Доступ к файлу slon.txt, находящемуся на сервере circ.org, осуществляется по протоколу http. Фрагменты адреса файла закодированы буквами от А до Ж. Запишите последовательность этих букв, кодирующую адрес указанного файла в сети Интернет~[16].

А).txt

Б)://

В)http

Г)circ

Д)/

Е).org

Ж) slon

Данный тип задачи можно шаблонизировать - существует конечный набор протоколов и структура абсолютного адреса конкретна и заранее известна. Этот тип заданий мы так же включим в доступные для генерации при написании нашего электронного образовательного ресурса.

Таким образом, хотя в курсе информатике присутствует много видов текстовых задач, не все из них есть в ОГЭ (в виду его специфики), кроме того, далеко не для каждой задачи можно легко алгоритмизировать их случайную генерацию. На первой итерации нашего ЭОР будет доступна возможность создания четырех видов задач - Задачи на хранение информации, задачи на передачу информации, а также Задания 1 (всех трех типов) и Задания 7.

2. Методические особенности использования авторского электронного образовательного ресурса при решении текстовых задач по информатике

2.1 Разработка электронного образовательного ресурса для генерации текстовых задач на хранение и передачу информации

Создание любого сервиса должно начинаться с выбора используемых технологий и продумывания архитектуры будущего приложения.

Глобально, система должна иметь пользовательский интерфейс доступный в сети, а также серверной приложение, которое будет отвечать за расчеты и генерацию самих задач.

Реализовать это позволит деление приложения на так называемые «backend» и «frontend» составляющие, где «backend» - это серверная часть сервиса, а «frontend» - браузерная.

Для разработки электронного образовательного ресурса будут использованы следующие технологии:

- язык программирования Python (для написания серверной части приложения);

- фреймворк Django (для backend web-приложения);

- язык программирования JavaScript (для написания логики пользовательской части);

- язык гипертекстовой разметки HTML и CSS (для оформления визуальной части);

- фреймворк Vue3 (для frontend web-приложения);

- библиотека axios (для отправки http запросов).

Конечной целью приложения является генерация набора из N задач одного из четырех доступных типов (Задачи на хранение и передачу информации, кодировку, построение абсолютного пути).

Учитывая все вышесказанное, наша система на вход должна получать число N - количество необходимых задач и значение типа перечисления

(Enum) - определяющий конкретный вид запрашиваемой задачи.

Тип перечисление определим следующим образом

```
# apps/cs_gen/choices.py
from django.db.models import IntegerChoices

class SupportedExerciseType(IntegerChoices):
    ENCODING_TASK = (0, "Задача на кодировку")
    PATH_TASK = (1, "Задача на построение абсолютного пути")
```

Единой точкой входа в систему генерации задач является класс ExerciseTypeFactory, предоставляющий интерфейс к созданию нужного генератора заданий, которые в свою очередь «строят» необходимый подтип задач.

```
# apps/cs_gen/services/exercise_factory.py
class ExerciseTypeFactory:

    def __init__(
                self,
                selected_type: SupportedExerciseType,
                amount: int) -> None:
        self._type = selected_type
        self._amount = amount


    def create_exercise_generator(self) -> AbstractExerciseGenerator:
        exercise_gen_class = self._get_exercise_gen_class()
        exercise_gen = exercise_gen_class(self._amount)

        return exercise_gen
```

```
def _get_exercise_gen_class(self):

    match self._type:

        case SupportedExerciseType.ENCODING_TASK:
            return EncodingExerciseTaskGenerator

        case SupportedExerciseType.PATH_TASK:
            return PathExerciseTaskGenerator

        case SupportedExerciseType.FILE_STORAGE_TASK:
            return FileStorageExerciseTaskGenerator

        case SupportedExerciseType.FILE_TRANSFER_TASK:
            return FileTransferTaskGenerator

        case _:
            raise UnsupportedTypeError(self._type)
```

Здесь, выбор класса-генератора происходит в методе _get_exercise_gen_class(), конструкцию match-case проверяющей соответствие типу SupportedExerciseType.

На основе полученного класса создается объект генератора, которому в конструктор передается количество необходимых для генерации задач.

Особое внимание стоит уделить возвращаемому типу - метод create_exercise_generator() возвращает AbstractExerciseGenerator - абстрактный класс генератора.

Рассмотрим его подробнее.

```
# apps/cs_gen/services/exercise_generators/base_generator.py
from abc import ABC, abstractmethod

from apps.cs_gen.services.tasks.base_task import AbstractTask


class AbstractExerciseGenerator(ABC):

    @abstractmethod
    def __init__(self, amount: int) -> None:
        super().__init__()


    @abstractmethod
    def generate(self) -> list[AbstractTask]:
        pass
```

Данные абстрактный класс определяет интерфейс, который должны реализовать все его наследники. В нем определен один абстрактный конструктор, принимающий amount типа int (целое число) - количество задач и абстрактный метод generate, возвращающий список классов абстрактных задач.

Использование абстрактных классов, равно как и фабрик при проектировании архитектуры описано в книге Роберта Мартина «Чистая архитектура».

Они составляют основу «Принципа инверсии зависимостей», в котором говорится о том, что компоненты системы, должны зависеть от абстракций (в нашем случае от абстрактных классов), а не от конкретных реализаций [17].

При таком подходе система получается более строгой и стабильной, компоненты становятся переиспользуемыми, а расширение функционала

имеет минимальное влияние на остальную часть системы.

Далее рассмотрим конкретные реализации класса AbstractExerciseGenerator. Поскольку на данном этапе мы предполагаем генерацию только двух типов задач, всего было реализовано четыре класса генератора, по одному на каждый тип. Во-первых генератор для задач на кодировку:

```
# apps/cs_gen/services/exercise_generators/encoding_task.py

import random
from .base_generator import AbstractExerciseGenerator

from apps.cs_gen.services.tasks.base_task import AbstractTask
from apps.cs_gen.services.tasks.builders.base_builder import (
        AbstractTaskBuilder,
        )
from apps.cs_gen.services.tasks.builders.encoding_builder import (
   WordExclusionExerciseBuilder,
   SymbolAmountExerciseBuilder,
   TextValueExerciseBuilder,

)

class EncodingExerciseTaskGenerator(AbstractExerciseGenerator):

   SUPPORTED_SUBTYPES = [
                WordExclusionExerciseBuilder,
                SymbolAmountExerciseBuilder,
                TextValueExerciseBuilder
                ]
```

```
    def __init__(self, amount: int) -> None:
        self._amount = amount
        self._tasks = []


    def generate(self) -> list[AbstractTask]:
        for i in range(self._amount):
            builder = self._make_random_subtype_builder()
            self._tasks.append(
                builder.build_task()
            )

        return self._tasks

    def _make_random_subtype_builder(self) -> AbstractTaskBuilder:
        subtype_builder = random.choice(self.SUPPORTED_SUBTYPES)
        builder = subtype_builder()

        return builder
```

Как и было указано выше, данный класс реализует оба абстрактных метода, кроме того имеет собственный внутренний метод _make_random_subtype_builder(), отвечающий за создание одного из поддерживаемых «строителей» задач.

Строитель - это паттерн (шаблон проектирования), заключающийся в создании класса, задачей которого является конструирование сложного объекта какого-либо другого класса.

В нашем случае, внутренний метод _make_random_subtype_builder() случайным образом (с помощью встроенной библиотеки random) выбирает одного из трех классов-строителей - WordExclusionExerciseBuilder,

SymbolAmountExerciseBuilder или TextValueExerciseBuilder, отвечающих за создание конкретного типа задач. Далее, создается объект класса-строителя и возвращается.

В методе generate(), вызов вышеописанного метода происходит в цикле, таким образом, каждую итерацию мы имеем нового, случайного строителя, который создаст класс-задачу и добавить в контейнер (список) _task.

При выходе из цикла, метод generate() вернет нам список созданных задач различных подтипов.

```
# apps/cs_gen/services/exercise_generators/path_task.py

import random

from .base_generator import AbstractExerciseGenerator

from apps.cs_gen.services.tasks.base_task import AbstractTask
from apps.cs_gen.services.tasks.builders.base_builder import (
        AbstractTaskBuilder,
)
from apps.cs_gen.services.tasks.builders.path_builder import (
        AbsolutePathExerciseBuilder,
)


class PathExerciseTaskGenerator(AbstractExerciseGenerator):

    SUPPORTED_SUBTYPES = [AbsolutePathExerciseBuilder,]

    def __init__(self, amount: int) -> None:
```

```
        self._amount = amount
        self._tasks = []


    def generate(self) -> list[AbstractTask]:
        for i in range(self._amount):
            builder = self._make_random_subtype_builder()
            self._tasks.append(
                builder.build_task()
            )

        return self._tasks

    def _make_random_subtype_builder(self) -> AbstractTaskBuilder:
        subtype_builder = random.choice(self.SUPPORTED_SUBTYPES)
        builder = subtype_builder()

        return builder
```

Генераторы задач на хранение и передачу информации, а также на построение абсолютного пути имеет идентичную структуру, с тем же набором методов. Единственное отличие - в этом классе определены другие подтипы классов-строителей. В данном случае реализована поддержка только одного конкретного подтипа.

Рассмотрев генераторы, можно переходить к описанию классов-строителей. По аналогии с генераторами, мы имеем один базовый абстрактный класс, у которого задан метод build_task().

```
# apps/cs_gen/services/tasks/builders/base_builder.py
from abc import ABC, abstractmethod
```

```
from apps.cs_gen.services.tasks.base_task import AbstractTask


class AbstractTaskBuilder(ABC):

    @abstractmethod
    def build_task(self) -> AbstractTask:
        pass
```

\end{python}

Описывать конкретные реализации классов-строителей целесообразно в паре, с создаваемой им классом-задачей, поскольку класс-строитель напрямую вызывает методы задачи, устанавливающие какие-то специфические для этого класса значения. Так, например, в классе-строителе WordExclusionExerciseBuilder, создающем задачу на поиск исключенного слова вызывается набор из пяти методов.

```
# apps/cs_gen/services/tasks/builders/encoding_builder.py

ENCODINGS = [enc for enc in ENCODINGS.values()]


class WordExclusionExerciseBuilder(AbstractTaskBuilder):

    TEMPLATE = (
        "В одной из кодировок {encoding} "
        "каждый символ кодируется {bits} битами. "
        "{student} написал текст (в нём нет лишних пробелов): "
```

```
            "«{words} {phrase}». "
            "Ученик вычеркнул из списка одно слово. "
            "Заодно он вычеркнул ставшие лишними запятые и пробелы?"
            "— два пробела не должны идти подряд."
            "При этом размер нового предложения в данной кодировке "
            "оказался на {bytes_diff} байт меньше, "
            "чем размер исходного предложения. "
            "Напишите в ответе вычеркнутое слово. "
        )

    def build_task(self) -> WordExclusionExercise:
        exercise = WordExclusionExercise(self.TEMPLATE)

        name = random.choice(samples.get_names())
        sample = random.choice(samples.get_we_samples())

        exercise.set_student_name(name)
        exercise.set_phrase(sample['phrase'])
        exercise.set_words(sample['words'])
        exercise.set_encoding(random.choice(ENCODINGS))
        exercise.set_answer(random.choice(sample['words']))

        return exercise
```

В каждом таком классе мы определили атрибут TEMPLATE - это базовый шаблон текстовой задачи, в который подставляются соответствующие значения. Эти значения задаются упомянутым выше набором из пяти методов, каждый из которых начинается с set_. Так, set_student_name() задает имя ученика, set_words() - определяет набор слов, среди которых надой найти исключенное, set_phrase() устанавливает фразу,

описывающие приведенные слова, set_encoding() задает кодировку, в которой записан представленный текст, а set_answer() предопределяет ответ на задачу - исключенное слово, которое было выбрано из списка установленных слов случайным образом.

Сама же задача представлена следующим классом:

```
# apps/cs_gen/services/tasks/encoding_tasks.py

class WordExclusionExercise(AbstractTask):

    def __init__(self, pattern: str) -> None:
        self._pattern = pattern
        self._description = None
        self._answer = None

    def get_description(self) -> str:
        return self._description or self._make_description()

    def _make_description(self) -> str:
        self._render_words()
        self._description = self._pattern.format(
            phrase=self._phrase,
            words=self._rendered_words,
            student=self._student,
            encoding=self._encoding['name'],
            bits=self._encoding['bits'],
            bytes_diff=self._calculate_bytes_difference()
        )
        return self._description
```

```
def _render_words(self) -> None:
    self._rendered_words = ", ".join(self._words)

def _calculate_bytes_difference(self) -> int:
    val_answer = len(self._answer) * self._encoding['bits']
    val_symbols = 2 * self._encoding['bits']
    return (val_answer + val_symbols) // 8

def get_answer(self) -> str:
    return self._answer.capitalize()

def set_answer(self, answer: str) -> None:
    self._answer = answer

def set_words(self, words: list[str]) -> None:
    self._words = words

def set_phrase(self, phrase: str) -> None:
    self._phrase = phrase

def set_encoding(self, encoding: dict[str,Union[str,int]])->None:
    self._encoding = encoding

def set_student_name(self, student: str) -> None:
    self._student = student
```

Ключевой момент в генерации и, как следствие, её решении является количество байт, на которое изменился текст после исключения слова.

Поскольку исключенное слово мы выбираем заранее - нам остается высчитать его длину с помощью стандартной функции len(). Получив длину слова, мы прибавляем к ней еще 2 символа - пробел и запятую (исходя из условия задачи). Полученную величину необходимо умножить на вес одного символа в данной кодировок и разделить на 8, чтобы привести биты к байтам. Таким образом, мы получим единственную недостающую в шаблоне величину.

Задача на определение информационного объема статьи на порядок легче в генерации, чем задача на исключение слов. Класс-строитель для нее выглядит следующим образом.

```
# apps/cs_gen/services/tasks/builders/encoding_builder.py

class TextValueExerciseBuilder(AbstractTaskBuilder):

    TEMPLATE = (
        "Статья, набранная на компьютере, "
        "содержит {page_amount} страниц, "
        "на каждой странице {string_amount} строки, "
        "в каждой строке {symbol_amount} символов. "
        "В одном из представлений {encoding} "
        "каждый символ кодируется {bits} битами. "
        "Определите информационный объём статьи "
        "в Кбайтах в этом варианте представления {encoding}."
        )

    PAGE_CHOICES = range(4, 129, 4)
    STRING_CHOICES = range(16, 129, 16)
    SYMBOL_CHOICES = range(16, 129, 16)
```

```
    def build_task(self) -> TextValueExercise:
        exercise = TextValueExercise(self.TEMPLATE)

        exercise.set_page_amount(random.choice(self.PAGE_CHOICES))
        exercise.set_string_amount(random.choice(self.STRING_CHOICES))

exercise.set_symbol_amount(random.choice(self.SYMBOL_CHOICES))
        exercise.set_encoding(random.choice(ENCODINGS))

        return exercise
```

В методе build_task мы поочередно устанавливаем количество страниц статьи, количество строк на странице, количество символов в строке, а также информацию о кодировке. Информация о кодировке представляет собой словарь (структура данных в Python, реализованная на основе хэш-таблиц). Этот словарь содержит информацию о названии кодировки и количестве бит - весе одного символа.

Единственный важный нюанс при генерации данного типа задач - произведение страниц, строк и символов должно быть кратно 1024, поскольку ответ необходимо дать в Кбайтах. Для этого мы с помощью встроенной функции range() генерируем для каждого поля соответствующие последовательности чисел, из которой в дальнейшем случайным образом возьмем только одно. Количество страниц у нас всегда будет кратно 4, количество строк и символов - 16.

Задача на определение информационного объема одного заранее заданного предложения - самая легкая из представленных. Реализованный класс-строитель так же не имеет каких-либо примечательных особенностей.

```
# apps/cs_gen/services/tasks/builders/encoding_builder.py
```

```
class SymbolAmountExerciseBuilder(AbstractTaskBuilder):

    TEMPLATE = (
        "В одной из кодировок {encoding} "
        "каждый символ кодируется {bits} битами. "
        "Определите размер в байтах следующего предложения "
        "в данной кодировке: {sentence}"
    )

    def build_task(self) -> SymbolAmountExercise:
        exercise = SymbolAmountExercise(self.TEMPLATE)

        exercise.set_encoding(random.choice(ENCODINGS))
        exercise.set_sentence(random.choice(samples.get_sa_samples()))

        return exercise
```

Здесь мы просто выбираем случайное предложение из множества доступных, а также устанавливаем случайную кодировку. Расчет ответа на задачу так же довольно прост - считаем длину предложения, умножаем на вес одного символа и делим на 8, чтобы получить ответ в байтах.

С точки зрения расчета ответа, последняя оставшаяся задача на построение абсолютного пути к файлу на порядок сложнее.

```
# apps/cs_gen/services/tasks/builders/path_builder.py

class AbsolutePathExerciseBuilder(AbstractTaskBuilder):

    TEMPLATE = (
```

```
    "Доступ к файлу {filename}{file_extension}, "
    "находящемуся на сервере {domain_name}{domain_zone}, "
    "осуществляется по протоколу {protocol}. "
    "Фрагменты адреса файла закодированы буквами от А до Ж. "
    "Запишите последовательность этих букв, "
    "кодирующую адрес указанного файла в сети Интернет. "
    "{variants}"
)

def build_task(self) -> AbsolutePathExercise:
    exercise = AbsolutePathExercise(self.TEMPLATE)

    exercise.set_protocol(random.choice(
                            samples.get_protocols()
                                ))
    exercise.set_domain_name(random.choice(
                            samples.get_domain_names()
                                ))
    exercise.set_domain_zone(random.choice(
                            samples.get_domain_zones()
                                ))
    exercise.set_filename(random.choice(
                            samples.get_filenames()
                                ))
    exercise.set_file_extension(random.choice(
                            samples.get_file_extensions()
                                ))

    return exercise
```

Класс-строитель для данного типа задач аналогичен предыдущим. Здесь, мы по порядку устанавливаем случайно выбранные из списка доступных тип протокола, хост, доменную зону, имя файла и его расширение.

Интерес вызывает класс самой задачи, поскольку необходимо сопоставить каждую часть пути с буквой русского алфавита, перемешать их, чтобы подставить в шаблон в переменную variants. При этом ответ необходимо давать ответ в виде перечисления соответствующих букв. Для этого в классе задачи были реализованы методы _make_variants_and_answer() и _make_answer()

```
# apps/cs_gen/services/tasks/path_tasks.py

def _make_variants_and_answer(self) -> str:

  ans = self._make_answer()

  ordered_ans = OrderedDict(sorted(
  ans.items(), key=lambda x: x[0]
                  ))

 for k, v in ordered_ans.items():
   ordered_ans[k] = f"{k}) {v}"

 return "\n".join(ordered_ans.values())

def _make_answer(self) -> dict:
  abc = list("АБВГДЕЖ")
  ans = {}
  ans[abc.pop(random.randint(0, len(abc)-1))] = self._protocol
```

```
        ans[abc.pop(random.randint(0, len(abc)-1))] = "://"
        ans[abc.pop(random.randint(0, len(abc)-1))] = self._domain_name

        ans[abc.pop(random.randint(0, len(abc)-1))] = self._domain_zone
        ans[abc.pop(random.randint(0, len(abc)-1))] = "/"
        ans[abc.pop(random.randint(0, len(abc)-1))] = self._filename

        ans[abc.pop(random.randint(0, len(abc)-1))] = self._file_extension
        self._answer = "".join(ans.keys())

        return ans
```

Метод _make_answer() создает словарь с правильными ответами. Буквы из списка «АБВГДЕЖ» становятся ключами словаря. Ответ получается конкатенацией ключей созданного словаря.

Метод _make_variants_and_answer() вызывает описанный выше метод и, кроме этого, сортирует словарь по ключам, чтобы варианты ответа стояли в алфавитном порядке и создает из этого строку, которая в дальнейшем попадет в переменную variants.

Классы строителей для задач на хранение и передачу информации имеют аналогичную структуру, их исходный код представлен ниже.

```
# apps/cs_gen/services/tasks/builders/info_builder.py

class FileStorageExerciseBuilder(AbstractTaskBuilder):

    TEMPLATE = (
        "Какой минимальный объём памяти (в Кбайт) "
        "нужно зарезервировать, чтобы можно было сохранить "
        "любое растровое изображение размером {width}x{height}
```

```
пикселей"
            "при условии, что в изображении могут использоваться "
            "{colors} различных цветов? В ответе запишите только "
            "целое число, единицу измерения писать не нужно."
        )

        def build_task(self) -> FileStorageExercise:
            exercise = FileStorageExercise(self.TEMPLATE)

            exercise.set_width(random.choice(range(128, 2049, 128)))
            exercise.set_height(random.choice(range(128, 2049, 128)))
            exercise.set_colors(random.choice(list(map(
                                    lambda x: 2**x, range(1, 9)))))

            return exercise

    class FileTransferExerciseBuilder(AbstractTaskBuilder):

        TEMPLATE = (
            "Файл размером {val} Кбайт передаётся через некоторое "
            "соединение со скоростью {sp_1} бит в секунду. "
            "Определите размер файла (в Кбайт), который можно "
            "передать за то же время через другое соединение "
            "со скоростью {sp_2} бит в секунду. В ответе "
            "укажите одно число — размер файла в Кбайт. "
            "Единицы измерения писать не нужно."
        )

        def build_task(self) -> FileTransferExercise:
            exercise = FileTransferExercise(self.TEMPLATE)
```

```
        exercise.set_val(random.choice(range(1, 1025)))
        exercise.set_sp_1(random.choice(list(map(
                                lambda x: 2**x, range(5, 10)))))
        exercise.set_sp_2(random.choice(list(map(
                                lambda x: 2**x, range(4, 10)))))

        return exercise
```

В задаче на хранение (класс FileStorageExerciseBuilder) мы задаем ширину и высоту в пикселях, а также количество цветов в палитре. В задаче на передачу информации (FileTransferExerciseBuilder) задается размер файла, изначальная скорость передачи и новая скорость.

Сами классы задач FileStorageExercise и FileTransferExercise спроектированы аналогично предыдущим, поэтому рассмотрим только методы расчета ответов.

```
# apps/cs_gen/services/tasks/info_tasks.py

# FileStorageExercise
def _make_answer(self) -> str:
  i = int(math.log2(self._colors))
  val = self._width * self._height * i
  return self._to_kb(val)

# FileTransferExercise
  def _make_answer(self) -> str:
  val_1 = self._val * 8 * 1024
  t_1 = val_1 // self._sp_1
  val_2 = self._sp_2 * t_1
```

```
    return self._to_kb(val_2)

def _to_kb(self, val: int) -> str:
    return val // 8 // 1024
```

В первом случае мы пользуемся формулой (N=2^i), высчитывая значение i и считаем общий вес, переводя ответ в Кбайты. Во втором - переводим вес в биты, делим на изначальную скорость, умножаем на новую и также переводим в Кбайты.

Таким образом, был нами был разработан сервис отвечающий за генерацию четырех типов задач. Разработанные компоненты системы соответствуют общепринятым нормам чистого кода и реализуют некоторые основные шаблоны проектирования архитектуры веб-сервисов. Полный исходный код проекта доступен в системе контроля версий GitHub по ссылке https://github.com/ZenBt/cs-exercise-generator

2.2 Примеры использования разработанного электронного образовательного ресурса на уроках информатики

Разработанный нами электронный образовательный ресурс имеет пользовательский web-интерфейс. При первоначальной загрузки страницы, пользователей встречает надпись с предложением выбрать тип задач (рис. 1) и выпадающий список из двух элементов (рис. 2).

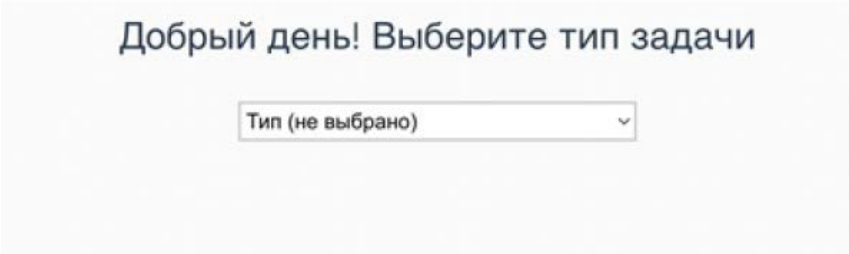

Рисунок 1 - Начальный экран

Изначально элемент списка не выбран, только после выбора конкретного типа задачи появляется следующее окно ввода (рис. 3).

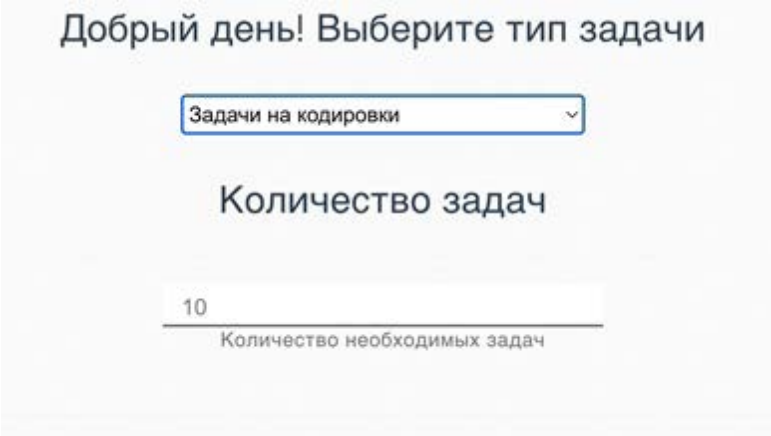

Рисунок 2 - Поле ввода количества задач

На данном, пилотном этапе развития электронного образовательного ресурса доступны четыре типа задач (рис.3) (задачи на хранение информации, задачи на передачу информации, задачи на кодировки и задачи на абсолютный путь), первый из которых имеет три подтипа - задачи на исключение слова, на вычисление информационного объема статьи и вычисление информационного объема данного предложения.

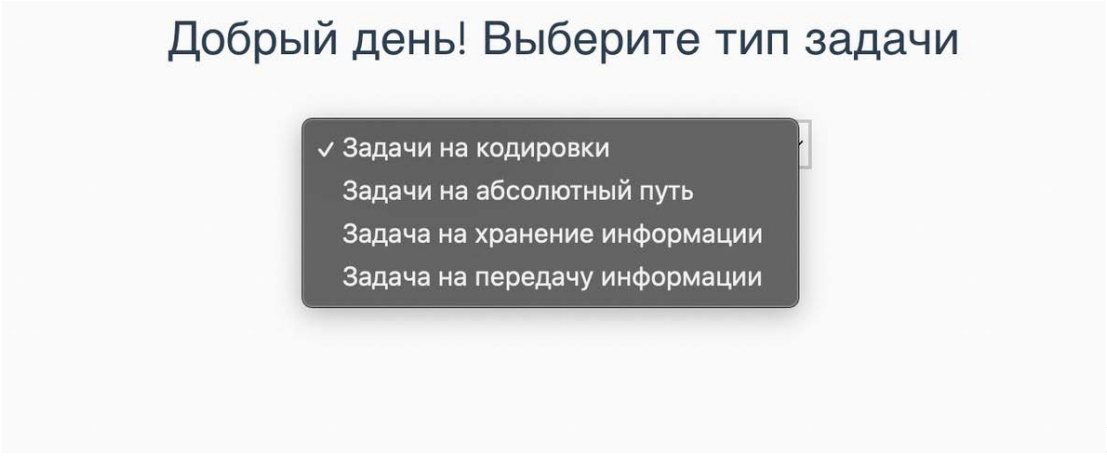

Рисунок 3 -Типы задач

Введя какое-либо целое положительное число, становится доступна кнопка для генерации задач, по нажатии на которую происходит отправка запроса на сервер, для получения запрошенных задач. Задачи отображаются непосредственно снизу вышеописанной кнопки (рис. 4). При повторном нажатии - будет сгенерирован новый набор задач, отличный от предыдущего.

Рисунок 4 - Поле ввода количества задач

Существует ограничение на количество одновременных задач. Невозможно создать за раз более 50 задач. Это ограничение введено

программно и обусловлено удобством использования - маловероятно, что понадобится генерировать более чем 50 задач, кроме того большое количество записей негативно влияет на пользовательский интерфейс. При введении отрицательного числа - кнопка сгенерировать пропадает, а при вводе числа большего, чем 50 и попытке генерации задач, появляется соответствующее уведомление о наличие ограничения (рис. 5).

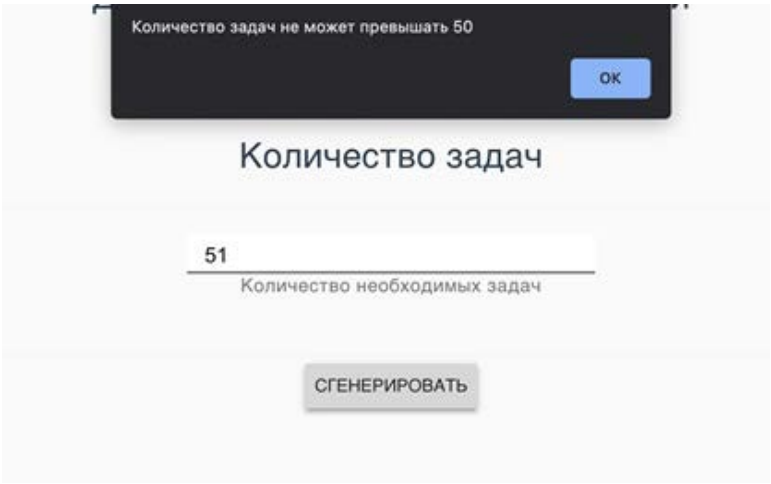

Рисунок 5 - Поле ввода количества задач

Стоит отметить, что у каждой из сгенерированных задач доступна возможность просмотра ответа. По нажатии на соответствующую кнопку, ответ будет здесь же отображен (рис.6). Возможность обратно скрыть ответ на данный момент не была реализована.

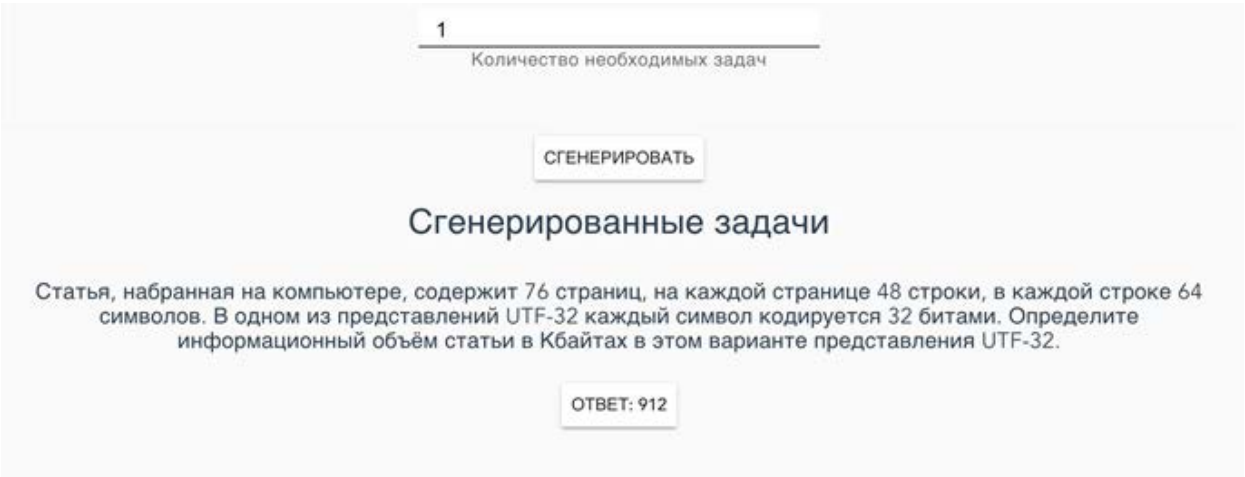

Рисунок 6 - Поле ввода количества задач

Сервис генерации задач имеет несколько сфер применения со своими целями и задачами.

Во-первых, он может быть использован учителем при подготовке к уроку, когда задачи на Решу-ЕГЭ уже все прорешены, а необходимо большее их количество. Обычно в такой ситуации учителю приходится тратить до нескольких часов на придумывание различных вариантов задач, учитывая при этом необходимость отсутствия повторов, а также специфику выбора исходных данных, ведь далеко не все варианты чисел дают в ответе целочисленный ответ. Однако использование разработанного ресурса позволяет решить эту проблему.

Во-вторых, возможно ситуация, когда учителю при проведении контрольной может понадобиться много однотипных вариантов. Специфика проблемы аналогично предыдущей, таким образом сервис позволить быстро и точно получить готовые задачи с ответами.

В-третьих, сервис может быть использован учащимися для самоподготовки. Большое количество возможных уникальных вариантов задач (более 2500 на каждый тип) гарантирует, что на контрольной в классе, если таковая будет, выданные задания будут численно отличаться. Таким образом учащиеся могут отработать алгоритм решения, на собственных, уникальных примерах задач.

В-четвертых, ресурс может пригодиться репетиторам для создания уникальных заданий для каждой группы учащихся.

Таким образом, разработанный нами электронный образовательный ресурс обладает минимально необходимым пользовательским интерфейсом и прилагающимся к нему практическим функционалам. При этом ресурс имеет вариативные сферы применения и может быть использован как преподавателями, так и самими учащимися.

ЗАКЛЮЧЕНИЕ

Таким образом, в ходе данной курсовой работы были решены все поставленные задачи:

— изучены теоретические аспекты использования электронных образовательных ресурсов на уроках информатики;

— рассмотрены текстовые задачи в курсе информатики основной школы;

— описан процесс разработки авторского ЭОР для решения текстовых задач на хранение и передачу информации;

— описаны методические особенности использования авторского электронного образовательного ресурса при решении текстовых задач на хранение и передачу информации по информатике.

Разработанный электронный образовательный ресурс имеет широкий количественный и качественный потенциал развития. В следующих версиях во-первых, следует добавить большее количество доступных для генерации задач, во-вторых, необходимо реализовать вывод решения к каждому типу задач, в-третьих, целесообразно добавить функцию решения задач на время без возможности посмотреть ответы в процессе, а также в дальнейшем сохранять историю решенных задач.

Исходя из всего вышеописанного, цель исследования можно считать достигнутой.

Практическая значимость данной курсовой работы заключается в возможности использовать разработанный электронный образовательный ресурс в педагогической практике, а также в наличии потенциала развития у созданного ресурса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Батакова, Е. Л. Интерактивные средства обучения как часть электрон-но-образовательных ресурсов / Е. Л. Батакова, Н. В. Батакова // Вестник ТГПУ. - 2016. - №1 (166). - URL: https://cyberleninka.ru/article/n/interaktivnye-sredstva-obucheniya-kak-chast-elektronno-obrazovatelnyh-resursov (дата обращения: 18.10.2022).

2. Якушина, Е. В. Электронно-образовательные ресурсы / Е. В. Якушина // Школьные технологии. - 2011. - №3. - URL: https://cyberleninka.ru/article/n/elektronno-obrazovatelnye-resursy (дата обращения: 18.10.2022).

3. Смирнова, И. Н. Формирование информационной культуры обучающихся посредством реализации электронных образовательных ресурсов на уроке / И. Н. Смирнова // Психология и педагогика служебной деятельности. - 2021. - №2. - URL: https://cyberleninka.ru/article/n/formirovanie-informatsionnoy-kultury-obuchayuschihsya-posredstvom-realizatsii-elektronnyh-obrazovatelnyh-resursov-na-uroke (дата обращения: 18.10.2022).

4. Кузнецова, М. В. Содержательный подход к разработке заданий для уроков информатики и ИКТ / М. В. Кузнецова, И. У. Толмачёва // Актуальные проблемы гуманитарных и естественных наук. - 2014. - №7-2. - URL: https://cyberleninka.ru/article/n/soderzhatelnyy-podhod-k-razrabotke-zadaniy-dlya-urokov-informatiki-i-ikt (дата обращения: 18.10.2022).

5. Корпунова, О. В. Применение электронных образовательных ресурсов на уроках физики, математики, информатики, с целью развития исследовательских навыков / О. В. Корпунова, М. А. Гаврилова // Мир науки. Педагогика и психология. - 2018. - №3. - URL: https://cyberleninka.ru/article/n/primenenie-elektronnyh-obrazovatelnyh-resursov-na-urokah-fiziki-matematiki-informatiki-s-tselyu-razvitiya-issledovatelskih-navykov (дата обращения: 18.10.2022).

6. Федорова, Г. А. Разработка и применение электронных

образовательных ресурсов в структуре методической подготовки бакалавров в педагогическом вузе / Г. А. Федорова // Вестник КГПУ им. В.~П.~Астафьева. - 2014. - №3 (29). - URL: https://cyberleninka.ru/article/n/razrabotka-i-primenenie-elektronnyh-obrazovatelnyh-resursov-v-strukture-metodicheskoy-podgotovki-bakalavrov-v-pedagogicheskom-vuze (дата обращения: 18.10.2022).

7. Данильчук, Е. В. Подготовка будущих учителей информатики к созданию и использованию виртуальных образовательных площадок в обучении школьников / Е. В. Данильчук, Н. Ю. Куликова // Известия ВГПУ. - 2020. - №10 (153). - URL: https://cyberleninka.ru/article/n/podgotovka-buduschih-uchiteley-informatiki-k-sozdaniyu-i-ispolzovaniyu-virtualnyh-obrazovatelnyh-ploschadok-v-obuchenii-shkolnikov (дата обращения: 18.10.2022).

8. Куликова, Н. Ю. Использование мультимедийных и интернет-технологий для разработки электронных образовательных ресурсов интерактивной доски при обучении информатике / Н. Ю. Куликова, С. Ю. Сердюкова, Е. Л. Склейнов // Известия ВГПУ. - 2013. - №2 (77). - URL: https://cyberleninka.ru/article/n/ispolzovanie-multimediynyh-i-internet-tehnologiy-dlya-razrabotki-elektronnyh-obrazovatelnyh-resursov-interaktivnoy-doski-pri (дата обращения: 18.10.2022).

9. Тарасов, Н. А. Использование электронных образовательных ресурсов с открытым исходным кодом в обучении / Н. А. Тарасов // МНКО. - 2017. - №2 (63). - URL: https://cyberleninka.ru/article/n/ispolzovanie-elektronnyh-obrazovatelnyh-resursov-s-otkrytym-ishodnym-kodom-v-obuchenii (дата обращения: 18.10.2022).

10. Информатика. Программа для основной школы: 5-6 классы. 7-9 классы / Л. Л. Босова, А. Ю. Босова. - 3-е изд. - М.: БИНОМ. Лаборатория знаний, 2015. - 88 с.: ил. - (Программы и планирование).

11. Белинский, С. С. Об определении понятия текстовая задача по математике / С. С. Белинский // Вестник магистратуры. - 2013. - №12-4 (27). -

URL: https://cyberleninka.ru/article/n/ob-opredelenii-ponyatiya-tekstovaya-zadacha-po-matematike (дата обращения: 21.12.2022).

12. Хайитова, Х. Г. Текстовые задачи и методы их решения / Х. Г. Хайитова // Проблемы педагогики. - 2021. - №6 (57). - URL: https://\\cyberleninka.ru/article/n/tekstovye-zadachi-i-metody-ih-resheniya (дата обращения: 21.12.2022).

13. Воистинова, Г. Х. Приемы обучения решению текстовых задач / Г. Х. Воистинова, Д. Х. Рахматуллина // StudNet. - 2021. - №1. - URL: https://cyberleninka.ru/article/n/priemy-obucheniya-resheniyu-tekstovyh-zadach (дата обращения: 21.12.2022).

14. Исаева, З. И. Методика обучения решению текстовых задач с помощью составления уравнений / З. И. Исаева // Проблемы современного педагогического образования. 2021. - №73-2. - URL: https://cyberleninka.ru/article/n/metodika-obucheniya-resheniyu-tekstovyh-zadach-s-pomoschyu-sostavleniya-uravneniy (дата обращения: 21.12.2022).

15. Образовательный портал для подготовки к экзаменам СДАМ ГИА. / Гущин Д.~Д., 2022. - Режим доступа: https://inf-oge.sdamgia.ru/problem?id =10383

16. Образовательный портал для подготовки к экзаменам СДАМ ГИА. / Гущин Д. Д., 2022. - Режим доступа: https://inf-oge.sdamgia.ru/problem?id=17

17. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. - СПб.: Питер, 2022. - 352 с.: ил. - (Серия «Библиотека программиста»).