**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ДОПУСТИТЬ К ЗАЩИТЕ**

**Профессор с возложенными обязанностями заведующего Кафедрой информационных систем в искусстве и гуманитарных науках**

**\_\_\_\_\_\_\_\_\_\_\_(Борисов Н.В.)**

**“\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**Направление 09.03.03 «Прикладная информатика»**

**Уровень Бакалавриат**

**Основная образовательная программа**

**«Прикладная информатика в области искусств и гуманитарных наук»**

**На тему**

 **«*Реализация автоматической проверки ответов в неоднородной базе автономных учебных заданий»***

 **Студента** *Козловой Елены Сергеевны*

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

*(подпись студента)*

**Руководитель:** *канд. техн. наук, доцент , Посов Илья Александрович*

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

*(подпись руководителя***)**

**Консультант:** докт. пед. наук, профессор *каф. ВМ-2, СПбГЭТУ «ЛЭТИ», Поздняков Сергей Николаевич*

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

*(подпись рецензента)*

**Санкт-Петербург**

**2018**

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| ВВЕДЕНИЕ………………………………………………………………….. | 4 |
| 1.Учебные объекты и системы электронного обучения…………………... | 7 |
| 2.Формат представления задач в системах электронного обучения……... | 10 |
| 2.1.SCORM………………………………………………………... | 10 |
| 2.2.IMS QTI………………………………………………………... | 13 |
| 2.3.Standalone tasks………………………………………………... | 14 |
| 3.Используемые технологии………………………………………………... | 17 |
| 4.Создание базы задач………………………………………………………. | 19 |
| 4.1.Поиск источников…………………………………………….. | 19 |
| 4.2.Сбор и анализ информации о задачах из выбранных источников………………………………………………………… |  22 |
| 4.3.Приведение полученных данных к формату автономных задач……………………………………………………………….. |  32 |
| 5.Автоматическая проверка ответов……………………………………….. | 35 |
| 5.1.Поиск подходящих задач……………………………………... | 35 |
| 5.2.Проектирование и реализация модулей автоматической проверки…………………………………………………………… |  36 |
| 6.Web-приложение………………………………………………………….. | 40 |
| 6.1.Создание макета………………………………………………. | 40 |
| 6.2.Проектирование web-приложения…………………………... | 41 |
| 6.3.Реализация web-приложения………………………………… | 41 |
| ЗАКЛЮЧЕНИЕ……………………………………………………………... | 44 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……………………….. | 45 |
| ПРИЛОЖЕНИЕ……………………………………………………………... | 47 |

ВВЕДЕНИЕ

В век компьютерных технологий обучение вышло на качественно новый уровень: у людей появилась возможность постоянно учиться чему-то новому, повышать квалификацию и получать официальное образование без территориальной привязки к обучающей организации. Знания, которые раньше можно было получить только лично находясь в стенах учебного заведения, в наше время становятся доступнее и ближе благодаря системам e-Learning (электронного обучения). Системы электронного обучения используются повсеместно, во многих учебных заведениях и фирмах, предусматривающих обучение сотрудников. С помощью систем электронного обучения можно сдавать экзамены, проходить учебные курсы, следить за своей успеваемостью и многое другое.

Учебный контент для систем электронного обучения создаётся преподавателями. И как бы прекрасна ни была идея электронного обучения, существующие системы e-Learning не всегда удовлетворяют нуждам преподавателя. Иногда случается так, что функционал одной системы не позволяет грамотно организовать обучение, а перенести задачи из одной системы в другую невозможно из-за разного подхода к реализации систем электронного обучения. В идеале, преподаватель имеет единую базу заданий и набор программных инструментов для дистанционной работы учеников с задачами, которые используются в разных ситуациях. Каждая программа поддерживает только подмножество заданий. Но подмножества могут пересекаться, и это значит, что некоторые задания можно использовать для разных целей с помощью разных инструментов.

Достигнуть подобного можно, если рассматривать задачу как объект в смысле объектно-ориентированного программирования. Одна задача реализует несколько интерфейсов, например, она может сама предоставить свое условие или выставить оценку на ответ пользователя. Многие действия, которые обычно совершаются при автоматизации обучения, становятся ответственностью самих задач, а не системы электронного обучения. Благодаря такому подходу можно будет минимизировать зависимость базы учебного контента преподавателя от систем электронного обучения.

Называем задачи такого формата автономными учебными заданиями (standalone tasks).

В рамках выпускной квалификационной работы была поставлена цель протестировать концепцию автономных заданий, собрав базу из задач формата для разработки и реализации автоматической проверки ответов внутри собранных автономных заданий.

Для достижения цели поставлены следующие задачи:

* Проанализировать возможности существующих систем электронного обучения;
* Найти существующие базы учебных заданий;
* Проанализировать информацию, содержащуюся в учебных заданиях;
* Проанализировать действий, совершаемые над учебными заданиями;
* Собрать учебные задания из разных источников и оформить их в виде автономных заданий;
* Проанализировать возможность реализации автоматической проверки ответов в собранных заданиях;
* Спроектировать систему автоматической проверки ответов;
* Реализовать автоматическую проверку заданий внутри автономных заданий;
* Реализовать web-приложение для демонстрации автоматической проверки ответов в автономных заданиях.

Результатом выпускной квалификационной работы будет являться база задач в формате автономных заданий, для некоторых из которых будет реализована возможность автоматической проверки ответа, возможности которой можно будет протестировать с помощью web-приложения. Базу смогут использовать преподаватели, чья предметная область совпадёт с тематикой задач, собранных в базе, для внедрения в процесс обучения.

1. Учебные объекты и системы электронного обучения

Под электронным обучением понимаются все формы обучения, где основной формой реализации процесса обучения выступают информационные и телекоммуникационные системы, прежде всего Интернет и мультимедиа. Такие формы обучения направлены на построение у учащегося системы знаний с учётом его индивидуального опыта, практики и подготовки.

Концептуальная модель системы электронного обучения представлена на рисунке 1.

Рисунок 1 – Концептуальная модель системы электронного обучения

В основе модели лежит база данных, или база знаний, содержащая образовательный контент (электронные образовательные ресурсы), или так называемые learning objects.

Learning Objects (или учебные объекты) – совокупность элементов учебных материалов, практических заданий и критериев оценки, объединённых на основе единой цели обучения.

Каждый элементарный объект обучения может включать в себя учебный текстовый или мультимедийный материал; глоссарий, понятия которого расшифровываются в данном тексте; элементы обсуждения (чат, форум, доска для рисования); элементы практических занятий; набор контрольных вопросов и тестов; метаданные объекта; инструкции для обработки информационного содержания объекта.

Основная идея использования learning object: открытость, возможность повторного использования и интероперабельность. К ключевым характеристикам learning objects можно отнести:

* Быстрое усвоение материала, т.к. в learning objects материал дан небольшими кусками;
* Самодостаточность learning object – каждый объект может быть взят независимо;
* Многоразовость – один learning object может использоваться в нескольких контекстах для нескольких целей;
* Learning objects можно сгруппировать в более крупные коллекции материалов.

Существует ряд сценариев использования учебных заданий:

* Задание можно отобразить учащемуся для решения:
	+ на экране компьютера;
	+ на листе бумаги;
* Отобразить учащемуся интерфейс для ввода ответа;
* Отобразить ответ на задание и комментарии к решению;
* Подсказать, как решать задачу;
* Автоматически проверить введенный учащимся ответ.

Реализация этих возможности требует специализированное учебное программное обеспечение (LMS).

Learning Management System (или LMS, система управления обучением) – программное обеспечение, управляющее процессом обучения, learning objects, отвечающее за анализ результатов обучения, планирование и протоколирование результатов обучающегося. Включает в себя функцию регистрации и авторизации обучающихся.

Runtime Service (RTS, сервис времени выполнения) – программное обеспечение, отвечающее за выполнение и доставку learning objects. Может включать в себя такие функции, как распределение ресурсов, диспетчеризация процессов, управление вводом/выводом и обработка данных. В некоторых системах RTS может быть включён в LMS.

Создание и использование технологических систем в образовании предполагает наличие системы стандартов и соглашений, адекватных условиям их применения. Среда обучения для таких систем формируется стандартами на интерфейсы, форматы учебных заданий, протоколы обмена информацией с целью обеспечения мобильности, стабильности, эффективности, способностью взаимодействовать и функционировать с другими системами без каких-либо ограничений доступа и реализации. Далее подробнее рассмотрим самые популярные из существующих форматов представления учебных заданий.

1. Форматы представления заданий в системах электронного обучения
	1. SCORM

SCORM – это сборник стандартов для систем электронного обучения. Используемые в SCORM стандарты определяют структуру учебных материалов и интерфейс среды выполнения. Основан на стандарте XML.

Все требования и стандарты в SCORM разделены на три книги: Content Aggregation Model (CAM), Runtime Environment (RTE) и Sequencing and Navigation (SN).

Книга CAM описывает типы учебного материала, способы описания его объектов и стандарты его упаковки для успешного обмена учебным материалом между системами обучения.

Книга RTE описывает требования, предъявляемые к системе электронного обучения с целью обеспечения использования учебного контента в различных системах e-Learning. В том числе:

* Определяет процесс запуска учебного контента;
* Определяет методы взаимодействия между учебным контентом и системой электронного обучения;
* Определяет модель данных, использующихся для передачи информации о работе слушателя с учебным контентом.

Книга SCORM SN определяет структуру дистанционного курса и процесс прохождения курса, основанный на результатах обучения и стратегии обучения, заложенный курс, разработчиком дистанционного курса.

Книга SCORM SN описывает, каким образом должны быть обработаны события, связанные с навигацией по дистанционному курсу, чтобы обеспечить предоставление слушателю необходимого ему учебного контента.

Блоки учебного материала в SCORM-пакете могут быть двух типов:

* Assets (ресурс) – элемент, не взаимодействующий с сервером LMS. Сюда могут относится текст, изображение, звук, видео и пр.
* Sharable Content Object (SCO, разделяемый объект содержимого) – объект, взаимодействующий с LMS. Он сообщает о ходе и результате обучения, получает и передаёт дополнительные данные. Как минимум такой объект сообщает о запуске и завершении сессии. SCO использует SCORM Runtime Environment для взаимодействия с LMS.

Схему организации контента в стандарте SCORM вы можете увидеть на рис. 2.



Рисунок 2 – Схема организации контента в стандарте SCORM

После разработки учебного контента, последний помещается в пакет – zip-архив (рис. 3). Данный пакет создаётся с целью передачи учебного контента между различными системами электронного обучения. Пакет контента состоит из следующих основных документов:

* XML-документ, описывающий структуру и ресурсы учебного контента: imsmanifest.xml;
* Физические файлы учебного контента.

Манифест пакета включает:

* Метаданные (свойства компонентов учебного материала);
* Организацию учебного материала (в каком порядке расположены компоненты);
* Ресурсы (ссылки на файлы, содержащиеся в пакете);
* Под-манифест (xml-файл может содержать под-манифесты, описывающие отдельные части пакета. Может иметь смысл, если пакет очень большой и имеет сложную структуру, чтобы не перегружать один файл большим объёмом данных).



Рисунок 3 – Концептуальная схема упакованного курса

Сделать задачу в формате SCORM сложно из-за большого объёма необходимых для этого знаний и данных. Чтобы курс заработал в системе электронного обучения, важно правильно упорядочить файлы, написать код их взаимодействия. Кроме того, задания в формате SCORM представляют собой «черный ящик»: пока ученик работает, система управления обучением не знает, что происходит внутри этого ящика. Она может только получить в конце результат участия. SCORM файлы вообще содержат в себе всё подряд, это может быть целый интерактивный учебник.

* 1. IMS QTI

IMS QTI (Instructional Management System, Question and Test Interoperability) – системы организации обучения, унифицированные вопросы и тесты.

Основные направления разработки спецификаций IMS – метаданные, упаковка содержания, совместимость вопросов и тестов, а также управление содержанием.

QTI – один из разделов стандарта IMS, стандартизирующий способы описания тестовых заданий и результатов тестирования. Он состоит из двух основных частей: Assessment, Section, Item (вопросы и их организация в тесте) и QTI Results Reporting (формат записи результатов).

Данные о тестах в формате IMS QTI описываются на основе языка XML, корневым элементом которого является элемент questestinterop. Он представляет собой своеобразный контейнер для элементов: assessment (тест), section (тема, секция) и item (тестовое задание). Корневой элемент содержит один или несколько элементов assessment (тест), внутри тестов содержится одна или более тем и уже темы содержат задания теста.

Элемент assessment (тест) является контейнером для одного теста согласно спецификации QTI. Он должен содержать, по крайней мере, один элемент section и может включать метаданные, цели теста, его продолжительность, информацию для тестируемого после прохождения этого теста и т.д. Элемент section создает логическую структуру теста путем рассредоточения тестовых заданий (ТЗ) по секциям (темам). Как правило, сходные ТЗ группируются и описываются внутри контейнера section.

В структуре элемента item можно выделить пять частей:

* Цели тестирования — материалы для описания целей.
* Материалы для определения контекста вопроса (метаданные).
* Часть представления ТЗ — содержит формулировку, вариант(ы) ответа.
* Часть обработки ТЗ — содержит инструкции по обработке результата (анализ и подсчет баллов).
* Часть обратной связи — материал для представления испытуемому в виде обратной связи — инструкции, рекомендации, или просто результат ответа на ТЗ.

Каждая из указанных частей также имеет развитую иерархическую структуру, что позволяет хранить информацию о тестах любых форм и конструкций.

Стандарт IMS QTI позволяет разделить процесс создания и использования тестовых заданий, так, как это реализовано в AICC и SCORM для учебных материалов. Создание качественных тестовых заданий требует значительных временных затрат.

Как мы можем видеть, QTI полностью описывает устройство заданий или тестов из нескольких заданий. Задание имеет условие, один из нескольких предопределенных типов ответов, описание обратной связи для учащегося, т.е. как реагировать на разные варианты правильных и неправильных ответов. Описывает, как вычислять количество полученных баллов.

Задачи в формате QTI имеют слишком большое описание стандарта. Кроме того, ограничен набор вариантов для взаимодействия с задачей.

* 1. Standalone tasks

Standalone tasks, или автономные задания, можно сказать, являются промежуточным вариантом SCORM и IMS QTI и, несомненно, более облегчённым. Это такой специальный формат, в котором описываются и реализуются все действия, которые можно совершать над конкретным заданием. Такие задания могут самостоятельно форматировать условие, ответ, проверять правильность ответа ученика, генерировать варианты условий и т.п. Системе электронного обучения требуется только знать возможности заданий и уметь обратиться к заданию, чтобы оно выполнило необходимые действия. Это необходимо, чтобы сделать задания максимально переносимыми между системами электронного обучения и при этом потенциально неограниченными в возможностях.

Автономное задание представляет собой папку-архив (рис. 4), содержащую файл «exercise.json» с информацией о задаче, сопутствующие задаче материалы (например, изображения), и модули, реализующие функционал данной конкретной задачи.



Рисунок 4 – Представление задачи в формате standalone task

Файл «exercise.json» содержит поля с информацией о задаче. Например, поле «task» содержит условие, поле «answer» ответ и так далее. Разные задачи содержат разное количество полей: где-то будет только условие, где-то условие, подсказка, решение и ответ. Количество полей значения не играет.

 Содержимое полей записывается в виде блока html-кода.

Модули, реализующие функционал автономных заданий, написаны на языке php.

Файл «modules.php» служит связующим звеном между системой электронного обучения и задачей: системе электронного обучения необходимо лишь обратиться к этому файлу, все дальнейшие действия задание будет выполнять само.

Файл «modules.php» вызывает другие модули автономного задания, которые могут, например, показать на экране информацию о задаче, сгенерировать условие или автоматически проверить правильность введённого пользователем ответа.

Автоматическая проверка ответа – это особый пример модуля автономной задачи. Каждое автономное задание решает, как оно будет проверять ответ: сравнивая с правильным или выполняя более сложные вычисления. Системе электронного обучения для этого не нужно будет выполнять лишних действий: как уже упоминалось ранее, ей нужно только обратиться к задаче.

Несомненным преимуществом автономных задач является независимость от систем электронного обучения. Задача может существовать «сама по себе»: вы можете просто открыть её в браузере. По этой же причине системе электронного обучения будет легко поддерживать данный формат заданий.

Полноразмерная реализация концепции формата автономных учебных заданий и их распространение смогут сделать знания и данные более доступными.

1. Используемые технологии

Практическая часть данной дипломной работы проводилась в два этапа.

На первом этапе создавалась база задач. Для этого информация о задачах из различных источников собиралась с использованием языка программирования Java. Выбор данного языка продиктован следующими его преимуществами:

* Java - объектно-ориентированный язык.

Это позволяет создавать модульные программы, исходный код которых может использоваться многократно.

* Язык Java крайне популярен.

В связи с этим для Java разработано множество библиотек, расширяющих специальные возможности языка. О библиотеках, которые были использованы в дипломной работе расскажем далее.

* Язык Java открыт и бесплатен.

Большая часть использованных для сбора базы задач ресурсов представляет собой web-страницы. Для выгрузки информации о задачах была использована Java-библиотека для разбора html-страниц JSoup. Данная библиотека позволяет извлечь необходимые данные, используя DOM, CSS и методы в стиле jQuery. Библиотека поддерживает спецификации HTML5 и позволяет разбирать страницы, как это делают современные браузеры. JSoup удобен в использовании, имеет обширный функционал и легок к пониманию: именно поэтому свой выбор мы остановили на нём.

База создавалась из задач в формате standalone exercises. Для этого необходимо было всю собранную информацию о каждой задаче записать в JSON-файл. Такие файлы создавались с использованием библиотеки Jackson, позволяющей оперировать файлами формата JSON. Выбор именно этой библиотеки обоснован необходимостью работы с большим количеством файлов, а Jackson считается самой высокопроизводительной библиотекой. Кроме того, Jackson многофункционален и прост в использовании.

Вторым этапом практической части была разработка модулей для задач из базы.

Модули были реализованы на языке PHP для расширения возможностей переноса между платформами систем электронного обучения, основанных на web-приложениях. Для сбора информации о возможности автоматической проверки так же был использован язык Java.

Для демонстрации работоспособности собранной базы автономных заданий было реализовано web-приложение. Приложение реализовано на HTML c использованием фреймворка Boostrap. Использование данного фреймворка обусловлено следующими его преимуществами:

* Bootstrap предлагает достаточное количество шаблонов и готовых решений для вёрстки макета страницы, что значительно сокращает время, необходимое для создания макета.
* Страницы, созданные с использованием данного фреймворка, поддерживают кросс-браузерность и адаптивность.
* Популярность Bootstrap. В связи с этим существует огромное количество обучающих материалов для освоения фреймворка.

Функционал web-приложения так же реализован с помощью PHP и Javasript.

Средой разработки были выбраны IntellijIDEA для работы на Java и PhpStorm для работы с PHP и HTML от компании JetBrains. Продукты JetBrains популярны среди разработчиков в связи с широким набором встроенных инструментов для рефакторинга, дизайном, ориентирующимся на продуктивность работы программистов, совместимостью со многими свободными инструментами разработчиков (CVS, Apache Ant, Maven и др.) и поддержкой многих языков программирования.

1. Создание базы задач
	1. Поиск источников

Для реализации неоднородной базы автономных учебных заданий нам необходимо собрать множество задач из открытых баз учебных заданий. Нужны самые разные виды заданий из самых разных баз, чтобы понять, можно ли собрать задачи в единую базу, у учебных объектов которой будет реализован функционал автономных заданий. Для этого необходимо сохранить как можно больше исходных данных о задаче.

Поиск баз проводился в Google по запросам

* База задач
* Задачи для школьников
* База олимпиадных заданий

и другим подобным. В связи с тем, что в функционале автономных задач необходимо реализовать автоматическую проверку ответов, предпочтения отдавались задачам по математическим дисциплинам, в которых может быть указан чёткий ответ: математика, информатика, физика и прочие. Одним из критериев выбора базы была возможность распознать и выгрузить данные задачи, в связи с этим не рассматривались сборники задач, например, данные в PDF. Ограничений по количеству задач на сайте не было.

Из предложенного поисковиком списка, были выбраны следующие базы.

* + 1. dist-olimpiada.krasnogorka.edusite.ru

Сборник олимпиадных задач по информатике. Задачи имеются с решением и без решения, разделены по темам. Страницы с условиями и ответами разделены друг от друга. На одной странице предлагаются список условий к задачам, на другой список решений этих задач (идентификация условия и ответа происходит по номеру задачи). Решение имеется не для всех задач. Для некоторых подразумеваются лишь подсказки, некоторые не решены совсем.

Задач немного, однако основная сложность состоит в соотнесении условия задачи с ответом к ней.

* + 1. problems.ru

Сайт problems.ru представляет из себя базу задач по математке, алгебре, геометрии, информатике, математическому анализу и т.п., которые отсортированы по темам, сложности, источникам, из которых были взяты эти задачи, и классам обучающихся. Задачи отображаются на web-странице в виде текстовой информации, разделённой на категории: условие, ответ, в некоторых случаях имеются подсказка и решение. В тексте заданий встречаются изображения, относящиеся к задаче.

В базе имеется колоссальное количество задач на разные темы.

* + 1. bebras.ru

Сборник задач по информатике, предлагавшихся для решения школьникам, участвовавшим в международном конкурсе по информатике «Бобёр». Задачи сразу собраны в архив файлов типа json.

* + 1. fipi.ru

Сайт федерального государственного бюджетного научного учреждения «Федеральный институт педагогических измерений» содержит открытую базу заданий ЕГЭ для подготовки школьников к единому государственному экзамену. Задачи в базе разделены по предметам, внутри каждого предмета – по темам. Задания не имеют решений, ответов или любой другой явной дополнительной информации, лишь условия.

Для нашей базы используем лишь задачи по информатике

* + 1. IDZ-2

В отличие от предыдущих источников, систему индивидуальных домашних заданий (ИДЗ) IDZ-2 не найти в сети интернет. Эту программу используют преподаватели Санкт-Петербургского государственного электротехнического университета «ЛЭТИ» имени В. И. Ульянова для формирования контрольных и домашних работ для студентов.

Программа генерирует варианты условий и ответы к ним, может группировать вопросы по темам для создания проверочных работ, и выдаёт их пользователю в виде pdf-файла для печати.

Программа содержит колоссальное количество данных, которые хотелось бы сохранить и использовать, возможно, немного иным способом, чем делает это IDZ-2. Поэтому с разрешения автора было решено пополнить базу автономных заданий задачами из IDZ-2.

* 1. Сбор и анализ информации о задачах из выбранных источников
		1. dist-olimpiada.krasnogorka.edusite.ru

Из всех выбранных баз задач, данный сайт содержит наименьшее количество задач, поэтому было решено начать именно с него.

dist-olimpiada.krasnogorka.edusite.ru – это web-сайт, на страницах которого представлены списки задач (рис. 5).



Рисунок 5 – Представление задач на странице dist-olimpiada.krasnogorka.edusite.ru

Из них, как мы видим, явно можно выделить два свойства каждой задачи: её номер и условие к ней. На отдельной странице, таким же списком, представлены ответы к задачам. Таким образом, выделяется третье свойство задачи: её ответ.

В процессе анализа страницы было установлено, что номер каждой новой задачи содержится в параграфе <p class=”zd”>, а условие самой задачи начинается со следующей строчки в новом параграфе. В самом условии могли находится изображения, которые так же необходимо было скачивать. В конце параграфов с условием могли содержаться изображение, маленькая пиктограмма, и ссылка на страницу с решением данной задачи.

Разбор задач проходил следующим образом:

* В отдельную папку с ресурсами сохранялись все страницы с задачами;
* С каждой страницы был взят блок, в котором содержались сами задачи и разбирался на строки;
* В цикле перебирались все строки взятого блока подряд. Происходил поиск параграфа <p class=”zd”>. Если такой параграф был найден, из него брался текст – номер задачи, а все параграфы после него, до следующего параграфа <p class=”zd”>, записывались в строку, содержащую условие задачи;
* Когда в всё условие было записано (т.е. был найден следующий параграф <p class=”zd”>), из строки условия отсекались, если имелись, ссылка на страницу с решением данной задачи и соответствующая ей пиктограмма;
* Далее строка с условием проверялась на наличие в ней элементов <img>: если таковые имелись, они скачивались, а в тексте условия ссылка на изображение заменялась на новую ссылку, ведущую к скачанному изображению в директории с задачей;
* В конце концов, полученные данные сохранялись в файл exercise.json в папку, соответствующую номеру задачи.

Далее разбирались страницы, содержащие в себе решения к задачам (рис. 6).



Рисунок 6 – Страница сайта dist-olimpiada.krasnogorka.edusite.ru, содержащая решения задач

Принцип построения страниц с условиями и с решениями на сайте dist-olimpiada.krasnogorka.edusite.ru одинаковый: блок, содержащий в себе необходимые нам данные, поделён на параграфы; параграф с номером задачи имеет тот же класс. Поэтому алгоритм разбора этих страниц в начале не отличается от предыдущего. Но после того, как мы нашли и записали номер задачи и текст решения, нам нужно найти папку, название которой совпадает с номером текущей задачи и перезаписать содержащийся в ней файл “exercise.json” следующим образом:

* Средствами библеотеки Jackson создаётся новый JSON-объект из файла “exercise.json”.
* Все имеющиеся в нём данные сохраняются, к ним добавляется информация о решении.
* В файл “exercise.json” соответствующей задачи записываются обновлённые данные.

Подробнее о записи результата в файл JSON и скачивании картинок будет рассказано в следующих параграфах.

* + 1. problems.ru

problems.ru – web-сайт, содержащий огромную базу задач из математических дисциплин. Каждая задача представлена на отдельной странице (рис. 7), её уникальный номер содержится не только в тексте на самой странице, но и в её адресе, что упрощает поиск и сохранение задач.



Рисунок 7 – Представление задачи на сайте problems.ru

В самом банальном случае задача из базы problems.ru имеет номер, условие, тему, сложность, классы учащихся, которые могут её решить и список источников, из которых она была взята. Разные задачи имеют разные наборы свойств и на деле полный их список таков:

* Номер задачи
* Тема
* Сложность
* Классы
* Условие
* Решение
* Подсказка
* Ответ
* Источники и прецеденты использования

Для удобства разбора и сохранения задач был написан отдельный Java-класс Exercise, содержащий все эти поля и умеющий сохранить свои данные в json-файл. При этом, если поле данного класса осталось пустым, оно не записывается в json-файл совсем.

Структура представления задачи на сайте выглядит следующим образом. В контейнере “div.componentboxheader” содержится номер задачи. За ним следует контейнер “div.componentboxcontents”, содержащий всю информацию о задаче. Название тем, к которым относится текущая задача, содержатся среди объектов “td.problemdetailssubjecttablecell”. На всей странице содержится единственный элемент “td.problemdetailsdifficulty”, текст которого содержит информацию о сложности задачи и классах учеников, которые могут её решить.

Свойство задачи (условие, подсказка, решение, ответ или источники и прецеденты использования) является заголовком <h3>, после которого записываются параграфы, содержащие текст. В тексте могут содержатся изображения, относящиеся к задаче.

Одна задача из базы problems.ru разбирается следующим образом:

* Текстовое значение контейнера “div.componentboxheader” записываем как номер текущей задачи;
* Из списка объектов найденных по тегу “td.problemdetailssubjecttablecell” находим название тем, зная, что данные теги содержат либо строку с названием, либо строки, список значений которых нам известен – их можно смело не брать в расчёт, проверяя текстовое содержимое очередного тега из полученного списка на соответствие этим самым известным нам значениям. Таким образом, полученный список тем добавляем к текущей задаче.
* Далее берём текстовую переменную тега “td.problemdetailsdifficulty”. Перебираем каждое слово элементом Scanner, находя соответствие со строками «Сложность:» и «Классы:» и записывая следующий за найденным элемент в соответствующее поле текущей задачи.
* Далее берём весь контейнер <div.componentboxcintents> и представляем его в виде строки, содержащей html-код. Строку разделяем на массив строк по регулярному выражению “<h3>”. Таким образом, имеем массив строк, первый элемент которого не будет содержать ничего необходимого нам, а каждый следующий элемент будет содержать значения какого-либо свойства задачи, которое нам нужно записать.
* Создаём HashMap, в который будем сохранять пары «ключ: значение», где ключом будет название найденного свойства задачи, а значением, соответственно, его текст.
* К каждому элементу массива (кроме первого) дописываем в начале “<h3>” (это необходимо, чтобы дальше без проблем парсить строку, содержащую html-код).
* В цикле проверяем каждую строку на содержание в ней элементов <img>. Если такие имеются, скачиваем их, а ссылку в строке изменяем на ссылку на скачанное в папку с текущей задачей изображение. В этой же строке находим элемент <h3>, записываем его текстовое значение в созданный HashMap как ключ, и присваиваем паре по этому ключу значение всей строки, с которой сейчас работаем.
* После цикла необходимо добавляем в класс нашей задачи недостающие данные. Делаем это, присваивая полям класса значения пары HashMap с соответствующим полю ключом, например:

exercise.setTask(tagData.get("Условие"));

* В заключении, сохраняем задачу в файл exercise.json в папке, соответствующей номеру задачи.

На сайте problems.ru содержится огромное количество задач. Каждая задача имеет свой номер, который явно отображён в адресе страницы: «http://problems.ru/view\_problem\_details\_new.php?id=N», где N – номер задачи. Таким образом, способ перебора задач значительно упрощается. Достаточно найти наименьший и наибольший номер задачи, а затем запустить цикл, который бы перебирал задачи одну за другой.

Однако, не все номера задач существуют. Чтобы программа не выдавала ошибку, когда находится номер, к которому нет задачи, пропускаем страницу, не обрабатывая её. Узнать об ошибке можем по “div.componentboxheader”: в случае, когда такой задачи не существует, значение строки этого контейнера будет «Ошибка».

* + 1. bebras.ru

База олимпиадных задач bebras.ru представлена набором JSON-файлов (рис. 8).



Рисунок 8 – Представление задач в базе bebras.ru

Задачи в данной базе имеют следующий набор свойств:

* id – номер задачи
* p – данные задачи
	+ title – название задачи
	+ statement – условие
	+ question – вопрос
	+ answers – предлагаемый список ответов
	+ explanation – пояснения

Значения полей-свойств записаны в виде html-кода.

Каждая новая строка файла – новая задача. В данных задач могут содержатся изображения, которые так же необходимо скачать и сохранить вместе с файлом, содержащим задачу.

Алгоритм сохранения задачи:

* Построчно читаем файл, содержащий базу задач;
* Присваиваем задаче порядковый номер (в порядке разбора – какой по счёту рассматриваем задачу, такой номер и присваиваем);
* Каждую строку представляем в виде объекта JsonNode, что даёт возможность получить доступ к полям JSON-файла и их значениям;
* Из полей statement, question, answers и explanation скачиваем изображения и сохраняем их в папку текущей задачи;
* И, в итоге, сохраняем текст самой задачи в файл exercise.json в папку, соответствующую номеру текущей задачи.

Всего в базе bebras.ru было 402 задачи. Разобрать и сохранить по отдельности удалось не все, т.к. в некоторых из них не было возможности скачать изображения: они были удалены или недоступны по своему адресу.

* + 1. fipi.ru

На официальном сайте Федерального института педагогических измерений (fipi.ru) в открытом доступе для подготовки школьников к единому государственному экзамену находится банк заданий ЕГЭ. Задания собраны по всем предметам, по которым можно сдать экзамен. Соответственно, задачи классифицируются по предметам, внутри каждого предмета – по темам. Формат представления задач на странице вы можете увидеть на рис. 9.



Рисунок 9 – Представление задач в базе fipi.ru

На каждой странице содержится 10 заданий, помещённых в контейнеры <td class=”cell\_1”>. Задачи в открытом банке не имеют ответов, только условия. Но для каждой задачи аналитически можно выделить следующие свойства:

* Номер задачи;
* Предмет;
* Тема;
* Условие.

Именно такие поля будем создавать для форматирования задачи в JSON-файл.

Алгоритм разбора задач с сайта fipi.ru:

* Скачиваем страницы, на которых имеются задачи;
* Выбираем с каждой страницы список контейнеров <td class=”cell\_1”>.
* В тексте каждого контейнера находим изображения и скачиваем их.
* Задаче присваиваем номер в порядке перебора, присваиваем значения полям «предмет» и «тема», в поле «условие» сохраняем значение рассматриваемого контейнера <td class=”cell\_1”>.
* Полученную задачу сохраняем в отдельную папку, название которой соответствует номеру задачи.

Открытый банк заданий ЕГЭ содержит огромное количество задач: десятки тысяч. В данной учебно-практической работе извлекались задачи лишь по информатике, их было 1600.

* + 1. IDZ-2

Задачи из этого задачника — генерируются, поэтому они все время разные, и в чистом виде условие и ответы у них найти не получится. Если запустить «.exe» файл с задачей, получим файлы «data.tmp» и «answer.tmp» — это часть условия и ответа. То, что меняется при генерации. Та часть текста, которая при генерации не меняется, хранится в отдельных файлах, «\*.prm» — с условием, «\*.sln» — с ответом. «\*» - потому что для каждой задачи эти файлы свои.

Все описанные выше файлы часто содержат инструкции LaTeX. Превращает эти инструкции в окончательную версию проверочных работ с разными вариантами как раз программа IDZ-2. Нам же нужно получать итоговые pdf-файлы с вариантами задач без запуска этой программы.

Для начала соберём информацию о задачах, имеющихся в системе. Информация о задачах содержится в XML-файле. Одна задача – один узел с именем «ROW». Собираем и сохраняем все имеющиеся о задаче данные, но особенно нас интересуют связанные с данной задачей файлы с расширениями «.prm» и «.sln». Так же важной для каждой задачи информацией является «ID» файла, запуск которого может сгенерировать её условие. Алгоритм сохранения задачи из строки XML в файл JSON в приложении 1.

Далее необходимо соотнести сохранённые ранее в задачи «ID» файлов, генерирующих условие, с информацией об этих файлах, содержащейся в другом XML-документе. Эту информацию необходимо собрать вместе, в одной задаче, чтобы в дальнейшем у нас получилось правильно сгенерировать её условие. Алгоритм сбора информации о задаче и информации о запускающем файле в приложении 2.

После этого, используя всю информацию, которую мы знаем о задаче, мы можем генерировать различные варианты задач и их ответов (приложение 3).

* 1. Форматирование полученных данных в формат standalone tasks.

По основной концепции данного формата, задача является папкой-архивом, в которой хранятся дополнительные данные (в нашем случае, это изображения) и файл с информацией о задаче в формате JSON. Таким образом, всю собранную информацию о задачах необходимо было записать в json-файлы, которые затем помещались в папку-архив, куда помимо того сохранялись сопутствующие изображения.

Эти действия были общими для всех задач, в связи с чем алгоритмы для них были вынесены статичными методами в отдельный класс Saver, который отвечал за скачивание изображений и сохранение данных.

Статичный метод downloadImages(Document document, String pathToImg, String url) находит в документе document все элементы <img>. Далее метод рассматривает каждое из изображений в цикле перебора коллекции, полученной в результате выполнения поиска изображений в документе: document.select("img"). Значение аттрибута src изображения передаётся другому методу вместе со строками pathToImg и url: getImage(String src, int imgNumber, String pathToImg, String url). Число imgNumber – порядковый номер изображения в документе. Метод getImage() ищет изображение по ссылке src, скачивает его в папку, путь к которой указан в pathToImg, называя его pathToImg + imgNumber + “.png”. Т.к. src чаще всего – это относительный путь к изображению, а не полная ссылка на него, проверяем, начинается ли src со строки “http://”, и, если нет, добавляем к src в начале url – эту строку мы определяем отдельно в каждой базе. Например, для базы задач problems.ru эта строка была: “http://problems.ru/”. В результате удачного выполнения метод getImage() возвращает полный путь до изображения на нашем компьютере. Методу downloadImage() остаётся поменять значения атрибута src текущего изображения.

Когда закончится цикл перебора найденных в документе элементов <img>, метод вернёт изменённый документ в виде строки.

Лаконичный метод saveJsonFile(String path, Object o) записывает Java-объект в файл, адрес которого path + “/exercise.json” в виде JSON-файла. Это всё выполняется средствами пакета Jackson:

public static void saveJsonFile(String exerciseNumber, Object o) throws IOException {

 ObjectMapper mapper = new ObjectMapper();
 File file = new File(exerciseNumber);
 file.mkdirs();

 mapper.writeValue(new File(exerciseNumber + "/exercise.json"), o);
}

По итогу проведённой работы по сбору базы автономных учебных заданий, нам удалось собрать 24 896 задач, некоторые из них со сторонними ресурсами в виде изображений. Из них:

* Задачи dist-olimp: 63;
* Задачи bebras: 395;
* Задачи fipi: 1 935;
* Задачи problems.ru: 22 101;
* Задачи IDZ-2: 402.
1. Автоматическая проверка ответов
	1. Поиск подходящих задач

Для реализации модулей автоматической проверки ответов необходимо из задач, в информации о которых имеется поле «answer», содержимое которого является ответом к соответствующей задаче, выделить те, к которым можно применить автоматическую проверку. Для автоматической проверки могут подойти такие задачи, в которых необходимо выбрать ответ(-ты) из представленных вариантов, ввести строку, которую можно сравнить с образцом, задачи, к которым нет явного ответа, но есть алгоритм решения, с результатом выполнения которого можно сравнить ответ пользователя и т.п.

Был проведен анализ существующих в собранной базе задач видов ответов, в результате которого мы выделили задачи, на которых можно реализовать автоматическую проверку, и сгруппировали по общим признакам. Были получены следующие группы:

* Задачи с ответом в виде числа.

Ответы в таких задачах необходимо сравнивать со строкой, которую ввёл пользователь.

* Задачи с выбором варианта ответа.

Ответы в таких задачах представлены в виде фраз «может/не может», «существует/не существует» и т.д.

* 1. Проектирование и реализация модулей автоматической проверки

Модуль автоматической проверки должен выводить на экран форму ввода ответа и кнопку типа «Submit», при нажатии на которую происходит сравнение введённых пользователем данных с правильным ответом. В зависимости от результата проверки, пользователь должен получить обратную связь, которая информировала бы его о правильности или неправильности введённого им ответа.

Создавалось два модуля, по одному для каждого вида задач: «answerInput.php» и «answerChoice.php» для задач с ответом в виде числа и для задач с выбором варианта ответа соответственно.

Задачам, в зависимости от их принадлежности к указанным выше группам, будем добавлять поле «answerType», чтобы в дальнейшем алгоритмически добавлять в архивы с задачами необходимые модули автоматической проверки ответа.

Кроме того, к задачам, в которых возможна автоматическая проверка ответа, необходимо было добавить поле «answerCheck», с содержимым которого будет сравниваться ответ, который даст пользователь. Формировалось поле «answerCheck» на основе данных, содержащихся в поле «answer».

Строка поля «answer» представляет собой HTML-код, содержащий в себе ответ к задаче.Необходимо было выделить лишь ту часть ответа, которая требуется для автоматической проверки, и записать её в специальное поле соответствующей задачи. Для этого воспользовались регулярными выражениями.

Для задач с ответом в виде числа искали последовательность цифр от 0 до 9 любой длины, которая начиналось бы либо с начала строки, либо с начала параграфа <p>. За числом может следовать уточняющее слово, например «33 обезьяны». Заканчивается последовательность точкой, пробелом и закрывающим тегом параграфа </p> или переводом строки /n:

Pattern pattern = Pattern.*compile*("(<p>|^)[0-9]+[ а-яА-Я]\*.( \\\n| </p>)");

Далее подходящие под регулярное выражение ответы форматировались до вида целого числа: убирались HTML-теги и текст, лишние точки и пробелы, после чего ответы в новом формате записывались в новое поле задачи «answerCheck». Здесь же определяли для задачи поле «answerType»:

if (m3.find()) {
 String newAnswer = m3.group();
 newAnswer = newAnswer
 .replaceAll("<p>", "")
 .replaceAll("</p>", "")
 .replaceAll("\\.", "")
 .replaceAll(" ", "")
 .replaceAll("[а-яА-Я]+", "");
 newExercise.put("answerCheck", newAnswer);
 newExercise.put("answerType", "answerInput");

}

Сложнее было с выделением из текста ответов для второй группы задач. Здесь встречались такие слова, как:

* Может/не может;
* Да/нет;
* Можно/нельзя;
* Существует/не существует;
* И др. вариации данных выражений.

Для удобства создания модуля автоматической проверки ответов для данного типа заданий решили, что будем искать по отдельности положительные и отрицательные ответы, которые начинаются с «Да», «Нет», «Существ…», «Мож…», «Мог…», «Нельзя», «Не мож..», «Не мог…», «Не существ…» следующими регулярными выражениями:

Pattern rightAns = Pattern.*compile*("(<p>|^)(Да[.,]|Мо[жг]|Существ)");
Pattern falseAns = Pattern.*compile*("(<p>|^)(Нет[.,]|Нельзя|Не\\ (мо[жг]|существ))");

Далее добавляем к задаче поле «answersVariants» со значением строки из двух вариантов ответа: «[Да, Нет]». Полю «answerCheck» присваиваем «1», если ответ положительный, и «2», если отрицательный, а полю «answerType» - «answerChoice». Покажем на примере алгоритма для положительных ответов:

if (m1.find()) { //для положительных ответов
 String newAnswer = "1";
 ArrayList<String> answersVariants = new ArrayList<>();
 answersVariants.add("Да");
 answersVariants.add("Нет");
 newExercise.put("answerCheck", newAnswer);
 newExercise.put("answersVariants", answersVariants.toString());
 newExercise.put("answerType", "answerChoice");
}

После применения алгоритма выделения данных для автоматической проверки из информации об ответах в задачах всей базы, мы получили 1278 задач, для которых можно использовать модули автоматической проверки.

Далее реализуем модули:

* Модуль создаёт и выводит форму ввода ответа на экран, в т.ч. кнопку «Submit»;
* Для передачи данных между сервером и клиентом форма использует метод «POST»;
* Модуль «answerInput.php» после нажатия кнопки «Submit» получает введённые пользователем данные, после чего сравнивает их со значением поля «answerCheck»;
* Для вариантов ответа в модуле «answerChoice.php» используем радиокнопки, количество которых зависит от размера массива в поле «answersVariants»;
* Модуль «answerInput.php» после нажатия кнопки «Submit» получает «value» той радиокнопки, которую выбрал пользователь, после чего сравнивает его со значением поля «answerCheck»;
* После сравнения полученных от пользователя данных с правильным ответом, модуль либо предлагает пользователю попробовать ещё раз и вновь выводит форму, если ответ был неверным, либо поздравляет с правильным решением.

Реализованные модули автоматической проверки вынесены в приложения 4 и 5.

1. Web-приложение
	1. Создание макета

Макет разрабатывался в программе Adobe Photoshop CC (рис. 10).



Рисунок 10 – Макет web-приложения.

Так как разработка web-приложения не была основной целью данной выпускной квалификационной работы, макет выполнен с расчётом максимально показать функциональность автономных учебных заданий, конкретно – работоспособность функции автоматической проверки ответа. Но при построении макета всё равно учитывалась необходимость сделать приложение не отталкивающим внешне и интуитивно понятным.

Верхняя область страницы занята баннером c названием web-приложения. Под баннером располагается рабочая область. В левой части рабочей области находится навигационная панель, где можно найти интересующую задачу. Предусмотрена панель поиска. Правая часть рабочей области позволяет просматривать информацию о задаче (например, условие задачи, решение, если таковое имеется и так далее). Здесь же, для тех задач, для которых это возможно, отображается форма ввода ответа для автоматической проверки. После проверки, информация о её результате выводится в том же месте.

* 1. Проектирование web-приложения.

Так как автономные задания по сути своей не требуют огромного функционала от системы электронного обучения, главное, что должна уметь последняя – обратиться к заданию. Для этого необходимо реализовать поиск по базе задач.

В базе задачи сгруппированы по источникам, из которых они были собраны, поэтому самый очевидный способ поиска задач – поиск по каталогам: пользователь обращается к конкретному каталогу, и на навигационной панели среди задач этого каталога выбирает необходимую.

При обращении пользователя к конкретному заданию в навигационной панели, система электронного обучения должна обратиться к модулю, который отвечает за отображение задания на странице. Модуль находится в папке-архиве задачи, поэтому всё, что требуется знать системе электронного обучения – к какому заданию обратился пользователь. Дальнейшие действия возьмёт на себя автономное задание.

* 1. Реализация web-приложения.

Как уже говорилось ранее, web-приложение не отвечает за отображение на странице тела задачи: все действия исполняет само автономное задание. В связи с этим реализация web-приложения уместилась в 77 строк (приложение 1).

В области навигационной панели реализована функция поиска по каталогу. Главная сложность этапа реализации перехода по каталогу: невозможность сразу сказать, является ли вложенная в текущий каталог папка архивом автономной задачи или очередным каталогом. Поэтому, когда пользователь нажимает на строку навигационной панели, методом «GET» передаётся информация о том, какую папку на этот раз выбрал пользователь. Далее проверяется, содержит ли данная папка файл «exercise.json», то есть, является ли эта папка уже архивом автономной задачи, либо это всё ещё директория. Если пользователь выбрал директорию, на панели навигации отобразятся папки, которые выбранная директория содержит.

Если же пользователь обратился к автономной задаче, web-приложение включает в тело кода главный модуль «modules.php» данной задачи. Этот модуль подключает к странице все остальные модули, связанные с задачей. Список таких модулей прописан в поле «modules» файла «exercise.json» задачи, что позволяет без проблем подключить их к странице как php-файлы.

modules.php

<?php
$file = file\_get\_contents($\_SESSION['id'] . "/exercise.json");

$dataJson = json\_decode($file);
$arrayModules = $dataJson->modules;

$\_SESSION['file'] = $dataJson;

foreach ($arrayModules as $module) {
 $modulePath = $\_SESSION['id'] . "/" . $module;
 echo "<br>";
 include $modulePath;
 echo "<br>";
}
?>

Некоторые модули к задачам мы так же реализовали в рамках данной дипломной работы, потому как они однотипны. Например, модуль, отвечающий за отображение условия задачи, мало чем будет отличаться от модуля, отвечающего за отображение решения:

task.php

<?php
 $task = $\_SESSION['file']->task;
 echo $task;
?>

solution.php

<?php
 $solution = $\_SESSION['file']->solution;
 echo $solution;
?>

Реализованное web-приложение практически не отличается от созданного макета. Но мы решили отказаться от идеи реализации панели поиска задач по вводимым символам в рамках данной выпускной квалификационной работы. Все задачи, собранные нами ранее в базу автономных заданий, были загружены в созданное web-приложение и готовы к использованию.

Готовое приложение доступно для просмотра по адресу http://standalone.multivariant.ru/index.php.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы были выполнены все поставленные в начале выпускной квалификационной работы задачи:

* Проанализированы возможности существующих систем электронного обучения;
* Собрана база автономных учебных заданий из задач в найденных открытых базах учебных заданий, содержащая в себе 24 494 задачи;
* В результате анализа возможности реализации автоматической проверки ответов для собранной базы заданий, для 1278 автономных заданий были спроектированы и реализованы модули автоматической проверки ответов;
* Реализовано web-приложение, демонстрирующее возможность автоматической проверки ответов в автономных заданиях. Кроме того, приложение демонстрирует преимущества формата автономных задач для электронного обучения.

Практические результаты работы (web-приложение и база автономных заданий) пригодны для дальнейшего развития и использования преподавателями в учебных целях.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Электронное обучение [Электронный ресурс] / ru.wikipedia.org – URL: [https://ru.wikipedia.org/wiki/Электронное\_обучение](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) *(дата обращения 17.12.2017)*
2. Макаров Р. В. Анализ стандартов для электронного обучения / Р.В. Макаров, научный руководитель Н.Ф. Телешева, 2016 г.
3. Силкина Н. С., Соколинский Л. Б. Модели и стандарты электронного обучения / Н.С. Силкина, Л.Б. Соколинский – журнал Вестник Южно-Уральского государственного университета, серия: Вычислительная математика и информатика, 2014 г.
4. Стандарт SCORM и его применение [Электронный ресурс] / cccp.ifmo.ru - URL: <http://cccp.ifmo.ru/scorm/index.html> *(дата обращения 18.12.2017)*
5. Углубленное изучение QTI [Электронный ресурс] / ibm.com – URL: <https://www.ibm.com/developerworks/ru/library/x-qti/index.html> *(дата обращения 18.12.2017)*
6. IMS QTI – Интероперабельность вопросов и тестов [Электронный ресурс] / metadata.ru – URL: <http://www.metadata.ru/content/view/37/52/> *(дата обращения 18.12.2017)*
7. jsoup: Java HTML Parser [Электронный ресурс] / jsoup.org - URL: <https://jsoup.org/> *(дата обращения 12.10.2017)*
8. Введение в JSON [Электронный ресурс] / json.org – URL: <https://www.json.org/json-ru.html> *(дата обращения 31.10.2017)*
9. Jackson Tutorial [Электронный ресурс] / tutorialspoint.com – URL: <https://www.tutorialspoint.com/jackson/> (дата обращения 31.10.2017)
10. Задачи [Электронный ресурс] / problems.ru – URL: <http://problems.ru/> *(дата обращения 01.11.2017)*
11. Олимпиадные задачи по информатике с решением [Электронный ресурс] / dist-olimpiada.krasnogorka.edusite.ru – URL: <http://dist-olimpiada.krasnogorka.edusite.ru/p4aa1.html> *(дата обращения 12.10.2017)*
12. Международный конкурс по информатике «Бобёр 2017» [Электронный ресурс] / bebras.ru – URL: <http://bebras.ru/bebras17/domain_contests> *(дата обращения 29.11.2017)*
13. Открытый банк заданий ЕГЭ [Электронный ресурс] / fipi.ru – URL: <http://85.142.162.119/os11/xmodules/qprint/index.php?theme_guid=521f264f9341e31197f4001fc68344c9&proj_guid=B9ACA5BBB2E19E434CD6BEC25284C67F> *(дата обращения 11.12.2017)*

ПРИЛОЖЕНИЕ

1

public class XMLReader {
 private static final String FILENAME = "resource/IDZ/IDZ/DB/MATHTASK.XML";
 public static void main(String[] args) {
 try {
 // Строим объектную модель исходного XML файла
 final File xmlFile = new File(System.getProperty("user.dir")
 + File.separator + FILENAME);
 DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
 DocumentBuilder db = dbf.newDocumentBuilder();
 Document doc = db.parse(xmlFile);
 // Нормализация (необязательно)
 doc.getDocumentElement().normalize();
 System.out.println("Корневой элемент: "
 + doc.getDocumentElement().getNodeName());
 System.out.println("============================");
 // Получаем все узлы с именем "ROW"
 NodeList nodeList = doc.getElementsByTagName("ROW");
 String[] attributes = new String[]{"ID\_Task", "Name", "Desc", "ID\_Topic", "ID\_Prg", "Params", "ID\_Source", "FNTemplateCond", "FNTemplateAnswer", "FNMacros", "GUID"};
 for (int i = 0; i < nodeList.getLength(); i++) {
 // Выводим информацию по каждому из найденных элементов
 Node node = nodeList.item(i);
 System.out.println();
 System.out.println("Текущий элемент: " + node.getNodeName());

 ObjectMapper mapper = new ObjectMapper();
 ObjectNode ex = mapper.createObjectNode();
 if (Node.ELEMENT\_NODE == node.getNodeType()) {
 Element element = (Element) node;
 for (String attribute: attributes) {
 System.out.println(attribute + ": "
 + element.getAttribute(attribute));
 ex.put(attribute, element.getAttribute(attribute));
 }
 Saver.saveJsonFile("IDZ/" + element.getAttribute("ID\_Task"), ex);
 }
 }
 } catch (ParserConfigurationException | SAXException
 | IOException ex) {
 Logger.getLogger(XMLReader.class.getName())
 .log(Level.SEVERE, null, ex);
 }
 }
}

2

public class replacePrg {
 private static final String FILENAME = "resource/IDZ/IDZ/DB/MATHPRG.XML";
 public static void main(String[] args) throws IOException {
 ObjectMapper mapper = new ObjectMapper();
 File jsonExercise = new File("IDZ/");
 for (File entry: jsonExercise.listFiles()) {
 ObjectNode newExercise = mapper.createObjectNode();
 BufferedReader bf = new BufferedReader(new InputStreamReader(new FileInputStream(entry + "/exercise.json")));
 //создаем строку с задачей
 JsonNode jn = mapper.readTree(getText(bf));
 String ID\_Task = "";
 for (Iterator<Map.Entry<String, JsonNode>> it = jn.getFields(); it.hasNext(); ) { //перебираем все поля
 Map.Entry<String, JsonNode> node = it.next();
 if (Objects.equals(node.getKey(), "ID\_Task")) {
 ID\_Task = node.getValue().asText();
 }
 if (Objects.equals(node.getKey(), "ID\_Prg")) {
 String ID\_Prg = node.getValue().toString();
 System.out.println(ID\_Prg);
 //узнаём ID того exe, который относится к этой задаче, и переносим все данные об exe в наш json
 BufferedReader bf2 = new BufferedReader(new InputStreamReader(new FileInputStream("resource/exe/" + ID\_Prg.replaceAll("\\\"", "") + ".json")));
 JsonNode jn2 = mapper.readTree(getText(bf2));
 for (Iterator<Map.Entry<String, JsonNode>> it2 = jn2.getFields(); it2.hasNext(); ) { //перебираем все поля
 Map.Entry<String, JsonNode> exeNode = it2.next()
 if (Objects.equals(exeNode.getKey(), "Name")) {
 newExercise.put("Prg\_Name", exeNode.getValue());
 }
 if (Objects.equals(exeNode.getKey(), "Desc")) {
 newExercise.put("Prg\_Desc", exeNode.getValue());
 }
 if (Objects.equals(exeNode.getKey(), "ExeName")) {
 newExercise.put("ExeName", exeNode.getValue());
 }
 }
 }
 newExercise.put(node.getKey(), node.getValue());
 }
 Saver.saveJsonFile("IDZ/" + ID\_Task, newExercise);
 }
 }
 public static String getText(BufferedReader bf) throws IOException {
 String str;
 String exercise = "";
 while ((str = bf.readLine()) != null) {
 exercise = exercise + str;
 }
 return exercise;
 }
 public static void creatorPrg() {
 try {
 // Строим объектную модель исходного XML файла
 final File xmlFile = new File(System.getProperty("user.dir")
 + File.separator + FILENAME);
 DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
 DocumentBuilder db = dbf.newDocumentBuilder();
 Document doc = db.parse(xmlFile);
 doc.getDocumentElement().normalize();
 System.out.println("Корневой элемент: "
 + doc.getDocumentElement().getNodeName());
 System.out.println("============================");
 // Получаем все узлы с именем "ROW"
 NodeList nodeList = doc.getElementsByTagName("ROW");
 for (int i = 0; i < nodeList.getLength(); i++) {
 Node node = nodeList.item(i);
 ObjectMapper mapper = new ObjectMapper();
 ObjectNode ex = mapper.createObjectNode();
 if (Node.ELEMENT\_NODE == node.getNodeType()) {
 Element element = (Element) node;
 for (String attribute: new String[]{"ID\_Prg", "Name", "Desc", "ExeName"}) {
 System.out.println(attribute + ": "
 + element.getAttribute(attribute));
 ex.put(attribute, element.getAttribute(attribute));
 }
 Saver.saveJsonFile("IDZ/exe/" + element.getAttribute("ID\_Prg"), ex);
 }
 }
 } catch (ParserConfigurationException | SAXException
 | IOException ex) {
 Logger.getLogger(XMLReader.class.getName())
 .log(Level.SEVERE, null, ex);
 }
 }
}

3

public class createPDF {
 public static void main(String[] args) throws IOException {
 String directory = "resource\\IDZ\\IDZ\\PROGS"; //директория для запуска exe
 File idzDirectory = new File("IDZ"); //директория с задачами
 int count = 5; //число создаваемых задач
 int filesNumber = idzDirectory.listFiles().length;
 int n = 0;
 for (File exercise: idzDirectory.listFiles()) { //перебираем все задачи с IDZ
 if (exercise.isDirectory()) {
 if (n < filesNumber) {
 String[] parametrs = getParametrs(exercise.getPath());
 String exerciseNumber = exercise.getName();
 for (int i = 1; i <= count; i++) { //создаём count pdf-файлов
 if (!parametrs[0].endsWith(".exe")) {
 parametrs[0] = parametrs[0] + ".exe";
 }
 String command = parametrs[0] + " " + i + " " + parametrs[1];
 System.out.println(command);
 int processStatus = createProcess(directory + "\\" + command, directory);
 if (processStatus == 0) {
 try (
 BufferedReader dataBR = new BufferedReader(new InputStreamReader(new FileInputStream(directory + "\\data.tmp")));
 BufferedReader answerBR = new BufferedReader(new InputStreamReader(new FileInputStream(directory + "\\answer.tmp")));
 BufferedWriter answerWriter = new BufferedWriter(new FileWriter("vprod/result/answer.tex"));
 BufferedWriter dataWriter = new BufferedWriter(new FileWriter("vprod/result/data.tex"))
 ) {
 String data = dataBR.readLine();
 String answer = answerBR.readLine();
 System.out.println(data + " ||| " + answer);

 String answerFile = parametrs[2];
 String conditionFile = parametrs[3];
 answerWriter.write(answerFile + ";" + answer);
 dataWriter.write(conditionFile + ";" + data);
 } catch (Exception e) {
 return;
 }
 //для создания pdf
 createProcess("pdflatex idz2.tex", "vprod/");
 Files.move(new File("vprod/idz2.pdf").toPath(), Paths.get("IDZ/" + exerciseNumber + "/" + i + ".pdf"), StandardCopyOption.REPLACE\_EXISTING);
 }
 }
 n++;
 }
 }
 }
 }
 public static String[] getParametrs(String path) throws IOException {
 ObjectMapper mapper = new ObjectMapper();
 BufferedReader bf = new BufferedReader(new InputStreamReader(new FileInputStream(path + "/exercise.json")));
 String str;
 String exercise = "";
 while ((str = bf.readLine()) != null) {
 exercise = exercise + str;
 }
 JsonNode jn = mapper.readTree(exercise);
 String exeName = "";
 String params = "";
 String answerFile = "";
 String conditionFile = "";
 for (Iterator<Map.Entry<String, JsonNode>> it = jn.getFields(); it.hasNext(); ) { //перебираем все поля
 Map.Entry<String, JsonNode> node = it.next();
 if (Objects.equals(node.getKey(), "ExeName")) {
 exeName = node.getValue().asText();
 }
 if (Objects.equals(node.getKey(), "Params")) {
 params = node.getValue().asText();
 }
 if (Objects.equals(node.getKey(), "FNTemplateAnswer")) {
 answerFile = node.getValue().asText();
 }
 if (Objects.equals(node.getKey(), "FNTemplateCond")) {
 conditionFile = node.getValue().asText();
 }
 }
 return new String[]{exeName, params, answerFile, conditionFile};
 }
 public static int createProcess(String command, String directory) throws IOException {
 Process p = Runtime.getRuntime().exec(command, null, new File(directory));
 try {
 boolean stopped = p.waitFor(10, TimeUnit.SECONDS);
 if (!stopped) {
 System.out.println( command + " не остановился");
 p.destroyForcibly();
 showProcessOutput(p);
 return 1;
 } else if (p.exitValue() != 0) {
 System.out.println(command + " завершился с кодом " + p.exitValue());
 showProcessOutput(p);
 }
 return 0;
 } catch (InterruptedException e) {
 System.out.println("Процесс выполнения exe файла прервался");
 return 1;
 }
 }
 private static void showProcessOutput(Process p) {
 InputStream er = p.getErrorStream();
 InputStream out = p.getInputStream();
 try {
 Files.copy(er, Paths.get("error.log"), StandardCopyOption.REPLACE\_EXISTING);
 Files.copy(out, Paths.get("output.log"), StandardCopyOption.REPLACE\_EXISTING);
 } catch (IOException e) {
 System.out.println("failed to flush streams " + e);
 }
 }
}

4

**answerInput.php**

$form = "
<form method='POST' class='form-inline'>
 <input name='answer' type='text' class='input-small'>
 <button type='submit' class='btn'>Отправить</button>
</form>
";
if (empty($\_POST['answer'])) {
 echo $form;
} else {
 $rigthAnswer = $\_SESSION['file']->answerCheck;
 $userAnswer = $\_POST['answer'];
 if ($userAnswer == $rigthAnswer) {
 echo "Поздравляю, вы ввели правильный ответ!";
 } else {
 echo "Вы ошиблись, попробуйте ещё раз. <br>";
 echo $form;
 }
}

5

**answerChoice.php**

$answersVariants = $\_SESSION['file']->answersVariants;
$answersVariants = str\_replace("[", "", $answersVariants);
$answersVariants = str\_replace("]", "", $answersVariants);
$arrayAnswers = explode(", ", $answersVariants);
$form = "
<form method='POST'>
";
for ($i = 1; $i <= count($arrayAnswers); $i++) {
 $form .= "
 <div class='radio'>
 <label>
 <input type='radio' name='answer' value='". $i ."'>" .
 $arrayAnswers[$i-1]
 ."</label>
 </div>
 ";
}
$form .= "
 <button type='submit' class='btn'>Отправить</button>
</form>
";
if (empty($\_POST['answer'])) {
 echo $form;
} else {
 $rigthAnswer = $\_SESSION['file']->answerCheck;
 $userAnswer = $\_POST['answer'];

 if ($userAnswer == $rigthAnswer) {
 echo "Поздравляю, вы ввели правильный ответ!";
 } else {
 echo "Вы ошиблись, попробуйте ещё раз. <br>";
 echo $form;
 }
}

6

<?php
session\_start();

if ($\_GET['id'] != null) {
 if (file\_exists($\_GET['id'] . "/exercise.json")) {
 $\_SESSION['id'] = $\_GET['id'];
 $\_SESSION['modules'] = $\_GET['id'] . "/modules.php";
 } else {
 $\_SESSION['directory'] = $\_GET['id'] . "/";
 }
} else {
 echo "hello";
 $\_SESSION['directory'] = "exercisesBase/";
}
?>

<html lang="en">
 <head>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <title>eLearning</title>

 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>

 <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap.min.css">
 <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap-theme.min.css">
 <link rel="stylesheet" href="main.css">

 <script src="//netdna.bootstrapcdn.com/bootstrap/3.3.2/js/bootstrap.min.js"></script>

 <script>
 function clickFunction(number) {
 location.href = "http://localhost//eLearning/index.php?id=" + number;
 }
 </script>
 </head>
 <body>
 <header class="row">
 <div id="h1" class="col-md-10 col-md-offset-1">
 Standalone eLearning system
 </div>
 <br>
 <div id="h2" class="col-md-10 col-md-offset-1">
 иные стандарты обучения
 </div>
 </header>

 <div class="row" id="center">
 <div id="exercisePanel" class="col-md-4">
 <ol class="breadcrumb">
 <li class="active" function onclick="clickFunction('exercisesBase')">exercisesBase</li>
 </ol>
 <?php
 $dir = $\_SESSION['directory'];
 if (is\_dir($dir)) {
 if ($dh = opendir($dir)) {
 while (($file = readdir($dh)) !== false) {
 if (($file != ".")&&($file != "..")) {
 echo "<div class='exercise' onclick='clickFunction(\"" . $dir . $file . "\")'>" . $file . "</div>";
 }
 }
 closedir($dh);
 }
 }
 ?>
 </div>
 <div id="mainField" class="col-md-8">
 <?php
 if ($\_SESSION['modules'] != null) {
 include ($\_SESSION['modules']);
 }
 ?>
 </div>
 </div>
 </body>
</html>