Распределенная информационная система RADOMS. Клиентская часть

Дудкин Е.В., студент кафедры информатки СПбГУ,
evgene.dudkin@gmail.com,
Щавелев Е.М., студент кафедры информатки СПбГУ,
egor.schavelev@gmail.com,
Кузнецов Е.В., студент кафедры информатки СПбГУ,
evgeney.kuznetsov@gmail.com,
Зотов М.А., студент кафедры системного программирования СПбГУ,
zotov1994@mail.ru.
Суворова А.В., лаборатория теоретических и междисциплинарных проблем информатики, СПИИРАН, suvalv@mail.ru
Тулупьев А.Л., кафедра информатики СПбГУ, лаборатория теоретических и междисциплинарных проблем информатики, СПИИРАН, alexander.tulupyev@gmail.com

Аннотация

RADOMS (Research and Development Outcomes Management System) — разрабатываемая распределенная информационная система с клиент-серверной структурой для управления сведениями о результатах интеллектуальной деятельности.

В работе рассмотрена реализация некоторых компонентов настоящей системы, таких как сервис по управлению видом таблиц, валидация форм, сохранение пользовательских настроек в локальное хранилище, а также возможность импорта данных в систему из пользовательских файлов и экспорт данных в виде отчётов.

Введение

Российские научные и научно-педагогические работники, исследователи и иные творческие деятели — всем им нужно предъявлять сведения о результатах работы в самых разных видах, структурах, срезах и форматах. Таким образом возникает необходимость в информационной системе, которая бы позволяла управлять этими сведениями.

Целью настоящей работы является автоматизация синтеза типовых отчётов об интеллектуальной деятельности, которые часто используются при составлении заявок для получения грантов или подтвер-

ждения успешного освоения грантов; для текущего контроля или изучения формальных показателей интеллектуальной деятельности.

Результатом настоящей работы является система RADOMS[6], которая позволяет клиенту добавлять информацию о своих публикациях, создавать необходимую выборку публикаций и экспортировать ее в различных форматах, таких как: xlsx, docx, csv и др.

Постановка задачи

Для успешной работы системы, необходимо было решить следующие задачи клиентской части:

- создать веб-страницу, позволяющую просматривать добавленные в систему данные (в виде таблицы), а также предоставляющую возможность добавлять новые записи и редактировать старые;
- реализовать сервис по управлению видом таблиц;
- реализовать универсальный компонент клиентской части, показывающий состояние элемента во время ожидания ответа от сервера, иными словами, отображение процесса загрузки данных с сервера (англ. «loadbar»);
- реализовать сохранение пользовательских настроек в локальное хранилище;
- реализовать компонент клиентской части, отвечающий за конкатенацию полей записи из таблицы публикаций;
- реализовать импорт данных из пользовательских файлов;
- реализовать синтез различного вида отчётов на основе пользовательских данных;
- обеспечить валидацию форм;
- обеспечить адаптивный дизайн.

Технологии

Для решения поставленных задач использовались технологии: AngularJS 1.5.X[1], Materializecss[4], HTML5[3], SCSS[7].

Реализация сервиса по управлению видом таблиц

В системе RADOMS список публикаций отображается в виде таблицы. Этот список может содержать такие поля, как категория, авторы, название, год, общее количество страниц, дата опубликования, город и другие. Из-за большого количества полей таблица становится объемной и трудной для понимая. Поэтому необходимо предоставить пользователю возможность выбирать те поля, которые нужно отображать в таблице.

Указанная цель была достигнута с помощью реализации сервиса по управлению отображаемых колонок.

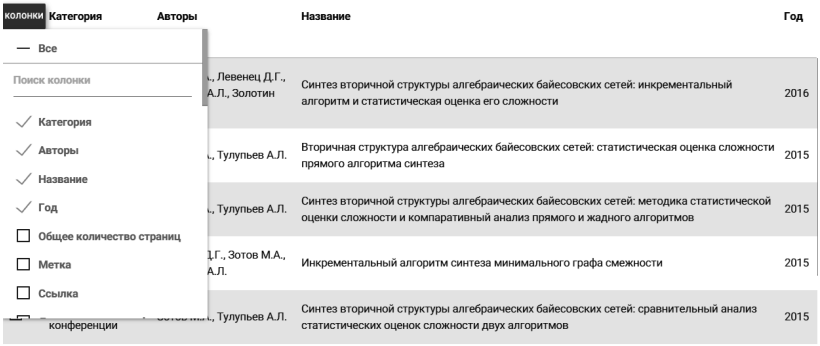
После наведения на кнопку «колонки» появляется список атрибутов (Рис. 1 а)). Именно здесь клиент может выбирать те атрибуты, которые будут отображаться на экране.

После выбора нужных атрибутов, внешний вид таблицы меняется (Рис. 1 б)).

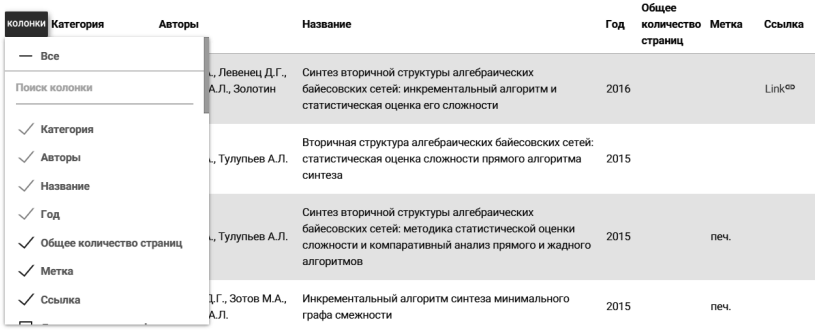
Валидация форм

От типа публикации зависят поля, обязательные для заполнения пользователем. Так, для статьи в журнале необходимыми полями будут: авторы, название публикации, год первая страница и название самого журнала. Для монографии, например, обязательными полями для заполнения будут только название, авторы и год.

Стояла задача предотвратить возможное добавление публикаций в систему, в которых пропущены какие-либо обязательные для заполнения поля. Данная задача была успешно решена.



а)



б)

Рис. 1: Зависимость вида таблиц от выбранных колонок

Таким образом, если поля, которые обязательно нужно заполнить, пусты, то у клиента не будет возможности добавить публикацию в систему. Так, если заполнены все необходимые поля кроме названия публикации, то кнопка добавления будет недоступна (Рис. 2).

Адаптивный дизайн

Система RADOMS должна обспечивать клиенту удобное использование не только с десктопных, но и с мобильных и планшетных устройств. Но если учитывать особенности таких устройств, а именно размеры экрана, то необходимо сделать так, чтобы внешний вид системы менялся относительно экрана, на котором просматривается

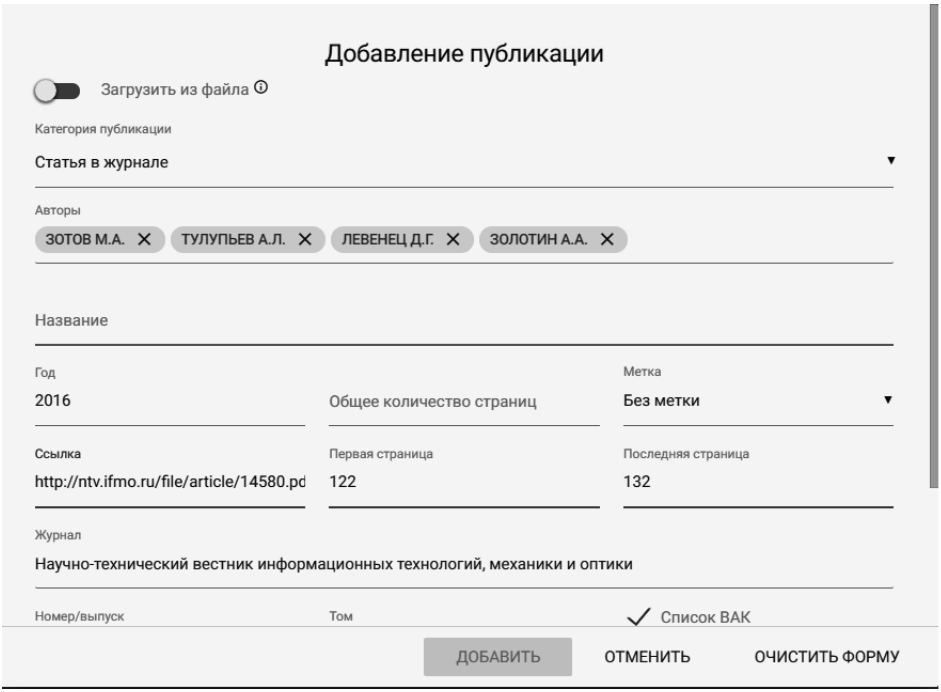
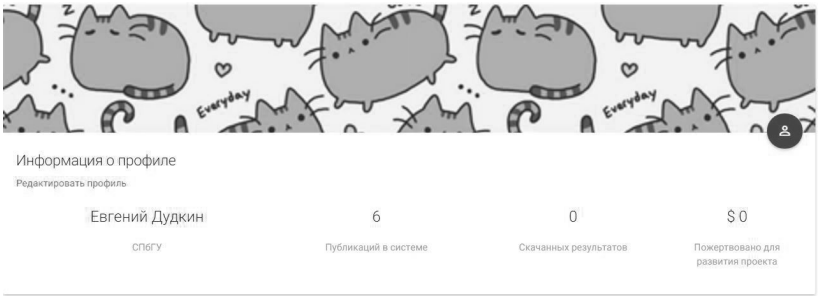


Рис. 2: Окно добавления публикации

приложение. Таким образом, обеспечение адаптивного дизайна — еще одна задача, которая стояла в рамках настоящей работы.

Данная задача была решена с помощью фреймворка Materializecss[4]. Этот фреймворк облегчает верстку и разметку страниц. С помощью «текучей адаптивной сетки», состоящей из 12 колонок, можно задавать размеры элементов. Причем эти размеры могут меняться относительно размера экрана.

Например, на странице «Профиль» есть карточка, в которой отображается информация о пользователе. На рисунке 3 а) показан внешний вид карточки просматриваемой с компьютера, а на рисунке 3 б) – с мобильного устройства.



а)

б)

Рис. 3: Адаптивная карточка пользователя

Реализация универсального компонента клиентской части, показывающего состояние элемента во время ожидания ответа от сервера

RADOMS — это инфомационная система с клиент-серверной структурой, что подразумевает под собой то, что вся информация о пользователе находится на сервере.

Здесь появляется одна из проблем клиент-серверной архитектуры, а именно — необходимость ожидания загрузки информации с серве-

ра. Пользователю необходимо дать понять, что в данный момент осуществляется связь и загрузка данных с сервера, иначе он может решить что система зависла и/или работает неправильно. Отсюда и возникает задача создания компонента под названием лоадбар, показывающего состояние выполнения запроса к серверу поверх элемента, к которому такой компонент присоединён.

Такой компонент должен органично вписываться в дизайн настоящего проекта, обладать свойством кроссбраузерности, т.е. одинаково отображаться в разных браузерах; также должен показывать «состояние» элемента, к которому он прикреплен (точнее, состояние выполнения запроса к серверу); также необходимо, чтобы разрабатываемый компонент был и оставался центрированным по ширине и высоте относительно размеров элемента, к которому он прикреплён. Другими словами, если пользователь изменил размер окна браузера, что, конечно, повлечёт изменение размера элемента, к которому прикреплён разрабатываемый компонент, последний должен переместиться в центр элемента всякий раз, когда изменятся размеры первого.

Компонент был успешно реализован. Теперь, например, при добавлении публикации, после нажатия кнопки «добавить», появляется полоса загрузки (Рис. 4). В данном случае форма добавления публикации имеет ширину 500 пикселей, и лоадбар получает форму полосы, но, если ширину формы сделать менее 300 пикселей, то лоадбар примет форму окружности (Рис. 5)

Данная задача была реализована в виде директивы AngularJS. К элементу необходимо добавить атрибут «with-progress» и переменную, хранящую в себе статус запроса к серверу. Пока переменная принимает значение true или строго больше нуля, элемент будет недоступным для работы и затемнится, а поверх него появится полоса загрузки.

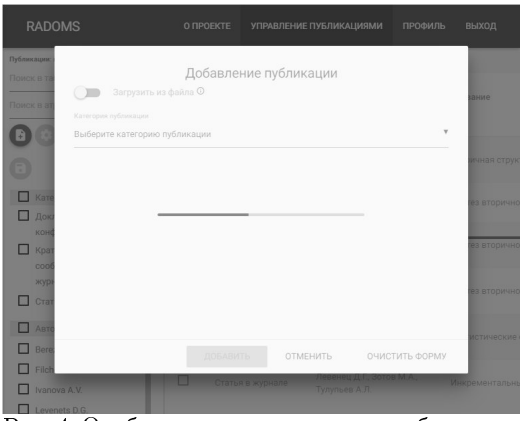


Рис. 4: Отображение загрузки при добавлении публикации на большом экране

Рис. 5: Отображение загрузки при добавлении публикации на небольшом экране

Сохранение пользовательских настроек в локальное хранилище

В ходе работы с таблицей загруженных публикаций, пользователь определяет какие колонки необходимо показывать и/или скрывать с помощью выпадающего списка всевозможных атрибутов над таблицей (Рис. 6).

Для пользователя будет ожидаемо, если в следующий раз, когда он посетит страницу с таблицей загруженных публикаций, предыдущая выборка отображаемых колонок сохранится.

Реализация такого поведения и являлось задачей, которая была решена с помощью сохранения набора отображаемых колонок в локальное хранилище. Для этого выбранные пользователем колонки сохраняются в localstorage сразу после внесения каких-либо изменений, а при загрузке страницы «Управления публикациями», из локального хранилища выгружается информация о выбранных ранее колонках.

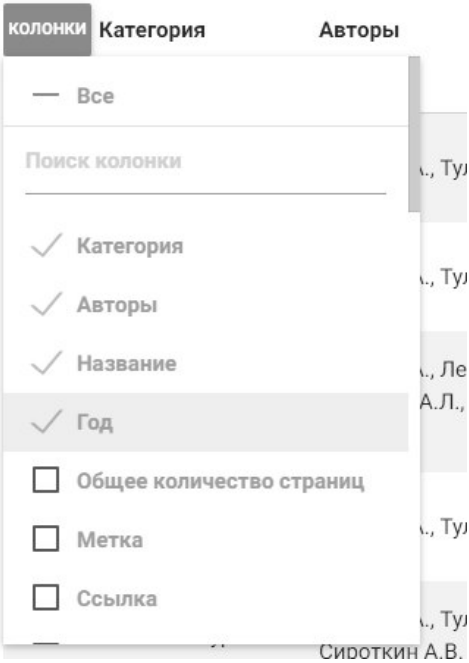


Рис. 6: Выпадающий список, содержащий названия колонок

Реализация компонента клиентской части, отвечающего за конкатенацию полей записи из таблицы публикаций

При работе с таблицей загруженных публикаций одной из главных функций является возможность экспорта выбранных записей таблицы. Каждая запись в таблице есть публикация, одним из атрибутов которой является категория. В свою очередь, каждая категория публикации однозначно определяет набор остальных атрибутов и выходной формат. Так, например, для категорий публикации «монография», «тезис» и «доклад на конференции» определен следующий выходной формат (Рис. 7).

Отсюда возникает задача правильной конкатенации полей записи при экспортировании данных из таблицы. Указанная задача была решена с помощью создания сервиса, в котором реализованы методы, принимающие на вход массив выбранных пользователем публикаций.

4	монография	[Автор И.О., Автор И.О., ...] [Название монографии]. [город]: [издательство], [год]. [количество страниц] с. <URL: [ссылка]>. <([Дополнительная информация])> *Пример: Тулупьева Т.В., Пащенко А.Е., Тулупьев А.Л., Красносельских Т.В., Казакова О.С. Модели ВИЧ-социально-значимого поведения в контексте психологической защиты и других адаптивных стилей.*

а)

9	тезис	[Автор И.О., Автор И.О., ...] [Название тезиса или доклада] // [Название сборника]. <Т. [том].> <([место конференции], [даты конференции]).> [город]: [издательство], [год]. С. [первая страница]<–[последняя страница]>. <URL: [ссылка]>. <([Дополнительная информация])> *Пример: Пащенко А.Е., Суворова А.В. Программный комплекс для экспертного оценивания интенсивности поведения респондента в условиях дефицита информации // Интегрированные модели в ИИ. Научно-практическая конференция студентов, аспирантов, молодых ученых и специалистов (Коломна, 26–27 мая 2009 г.). Научные доклады. В 2-х т. Т. 2. М.: Физматлит, 2009. С. 220–241.*
10	доклад на конференции	

б)

Рис. 7: Пример выходного формата для определенных категорий

Более детально, суть работы каждого метода можно описать следующим образом.

1. toPlainBibliography — метод, который на вход принимает массив публикаций, а на выход отдает массив из строк, конкатенированных по правилам описанным для каждой категории[1]. Входной массив публикаций необходимо особым образом отсортировать по количеству авторов: сначала должны располагаться публикации с одним автором в лексикографическом порядке, далее все остальные публикации так же в лексикографическом порядке.

2. toPlainBibliographyToNumeric — выходные данные метода аналогичны выходным данным из предыдущего метода; различие заключается лишь в том, что в начале каждой вставлено число, равное индексу данной строки в результирующем массиве.

3. toPlainClusterBibliography — на выходе структура представляет собой словарь (map), у которого ключом является либо год, либо категория (в зависимости от входящих параметров), а значением — список публикаций, соответствующих данному ключу (простыми словами, метод группирует публикации по году или ка-

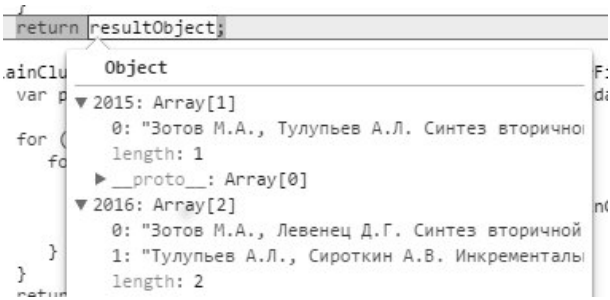
[1] Правила конкатенации описаны в техническом задании проекта

тегории). Внутри метода также для каждой записи проводится соответствующая конкатенация, и для каждого ключа осуществляется сортировка аналогично пункту 1 (Рис. 8).

4. toPlainClusterBibliographyNumeric — выходной формат аналогичен пункту 3, с дополнением в том, что в начале каждой строки вставлено число, равное индексу данной строки в массиве, принадлежащим соответствующему ключу.

✓	Статья в журнале	Зотов М.А., Левенец Д.Г.	Синтез вторичной структуры алгебраических байесовских сетей: инкрементальный алгоритм и статистическая оценка его сложности	2016
✓	Доклад на конференции	Зотов М.А., Тулупьев А.Л.	Синтез вторичной структуры алгебраических байесовских сетей: сравнительный анализ статистических оценок сложности двух алгоритмов	2015
✓	Монография	Тулупьев А.Л., Сироткин А.В.	Инкрементальный алгоритм синтеза минимального графа смежности	2016

а)



б)

Рис. 8: Пример работы метода toPlainClusterBibliography

Выходной результат методов в дальнейшем используется при реализации экспортирования данных в форматы Excel и Word.

Импорт и экспорт данных

Важной функцией сервиса RADOMS является возможность импорта данных в систему из пользовательских файлов и, наоборот, экспорт данных на устройство пользователя. Для решения этих задач был создан парсер, позволяющий представлять полученные из пользовательских файлов данные в JavaScript-объект.

Импорт

Принцип работы заключается в следующем. Парсер получает на вход файл формата csv/xls/xlsx и представляет полученные из него данные в виде массива JavaScript-объектов, причём каждый объект создаётся на основе одной строки из файла, кроме первой — элементы же первой строки используются в качестве ключей создаваемых объектов.

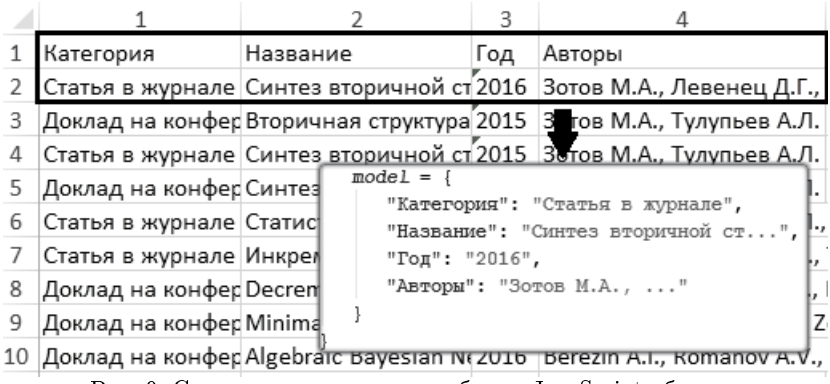


Рис. 9: Соответствие записи в таблице JavaScript-объекту

Для реализации парсинга файлов формата xls и xlsx были использованы библиотеки SheetJS/js-xls[8] и SheetJS/js-xlsx[9] соответственно. Данные библиотеки обладают идентичным функционалом, отличаясь только форматом входных данных, и их методов полностью хватает для решения поставленной задачи. В частности, библиотеки предоставляют следующие методы:

- метод read, позволяющий представить полученные после чтения данные в удобный для обработки формат;

- метод sheet_to_json, производящий парсинг данных в массив JavaScript-объектов.

Для реализации парсинга csv файлов была использована библиотека Papa Parse[5], которая для этих целей содержит специальный метод — Papa.parse.

Экспорт данных в виде отчёта

Одним из пунктов технического задания была реализация генератора docx файлов, содержащих отчёты, составленные на основе пользовательских данных. Под пользовательскими данными понимается массив, содержащий объекты, причём каждый объект является представлением одной публикации, то есть содержит такие поля, как «Авторы», «Название», «Год публикации» и прочее. На данном этапе разработки требовалось реализовать возможность генерации двух типов отчётов.

- *В виде списка литературы.*

СПИСОК

опубликованных работ

2006

1. Суворова А.В. Изменения СДО, произошедшие за пять лет // Компьютерные инструменты в образовании. №6. 2006. С. 27–32.

2009

1. Пащенко А.Е., Тулупьев А.Л., Суворова А.В., Тулупьева Т.В. Сравнение параметров угрозообразующего поведения в разных группах на основе неполных и неточных данных // Труды СПИИРАН. 2009. Вып. 9. СПб.: Наука, 2009. С. 252–261.
2. Суворова А.В., Пащенко А.Е., Тулупьева Т.В. Тулупьев А.Л. Построение доверительных интервалов оценок интенсивности рискованного поведения на основе неравенства

- *В виде таблицы,* содержащей данные по публикациям для конкретных авторов.

СПИСОК

научных работ
А. В. Суворова

1	2	3	4	5	6
№ п/п	Наименование работы, ее вид	Форма работы	Выходные данные	Объем, стр.	Соавторы
1	2	3	4	5	6
1	Изменения СДО, произошедшие за пять лет (статья)	печ.	Компьютерные инструменты в образовании. №6. 2006.		
2	Косвенные оценки и сравнение параметров угрообразующего поведения в разных группах по неполным и неточным данным (статья)	печ.	Международная конференция по мягким вычислениям и измерениям. Сборник докладов. 2009. Т. 2. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2009	5/1	Тулупьев А.Л., Тулупьева Т.В., Пащенко А.Е.

Поскольку предполагалось, что отчёты должны иметь единообразный вид, было решено реализовать их генерацию на основе шаблонов с заполняемыми полями. Для этих целей была выбрана библиотека DocxGenJS[2], позволяющая генерировать файлы на основе заготовок и затем скачивать их на устройство пользователя. Рассмотрим реализацию данного метода на примере отчёта, представляющего из себя список литературы.

1. Шаг первый — создание шаблона. Шаблон представляет из себя docx файл, содержащий определённые теги (Рис. 10). Теги располагаются таким образом и имеют такой стиль, какой должен будет иметь расположенный на их месте текст в конечном файле.

СПИСОК

опубликованных работ

{-w:p list}**{year}**

1. {-w:p article}{authors} {publication}{/article}{/list}

Рис. 10: Пример шаблона

2. Шаг второй — чтение и парсинг файла методами библиотеки DocxGenJS.

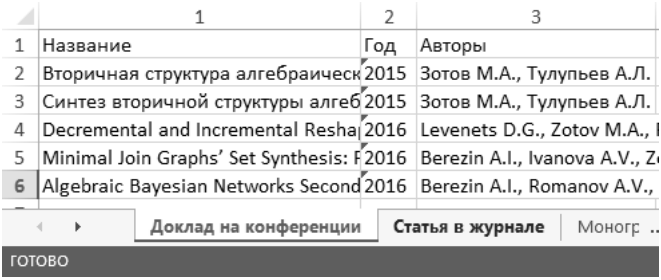
3. Шаг третий — представление данных, которые требуется ввести в отчёт, в виде объекта, ключи которого совпадают с названиями тегов в файле-шаблоне. В случае, если в файл нужно записать несколько однотипных строк (например, список — строки таблицы), данные представляются в виде массива объектов, каждый из которых соответствует одной строке.

4. Шаг четвёртый — генерация нового файла на основе полученного на третьем шаге представления данных. Поскольку в системе шаблоны пришлось бы физически расположить на сервере, и при каждом запросе на генерацию отчёта, пришлось бы заново их читать и парсить, а результат выполнения таких действий каждый раз был бы одинаковым, было принято решение «запомнить» результаты выполнения первых двух шагов. В результате получилось несколько base64 строк, каждая из которых соответствует определённому шаблону, на основе которых уже можно производить генерацию отчётов. Для хранения этих строк был использован сервис Constant, поставляемый библиотекой AngularJS.

Экспорт данных в виде таблицы

Также одним из уже реализованных вариантов экспорта данных является возможность создания xlsx книги, каждый лист которой содержит данные по публикациям определённой категории, а каждая строка в листе содержит данные по одной конкретной публикации.

Для решения этой задачи была использована библиотека SheetJS/js-xlsx, позволяющая генерировать xlsx файлы. На вход функции подаются пользовательские данные. Далее происходит заполнение таблицы на их основе: в первую строку записываются все ключи из объектов пользовательских данных, а затем каждая строчка заполняется сведениями о каждой конкретной публикации. При этом сама xlsx книга делится на несколько листов, каждый из которых содержит только данные о публикациях определённой категории (пример — Рис. 11).



	1	2	3
1	Название	Год	Авторы
2	Вторичная структура алгебраическ	2015	Зотов М.А., Тулупьев А.Л.
3	Синтез вторичной структуры алгеб	2015	Зотов М.А., Тулупьев А.Л.
4	Decremental and Incremental Resha	2016	Levenets D.G., Zotov M.A.,
5	Minimal Join Graphs' Set Synthesis: F	2016	Berezin A.I., Ivanova A.V., Z
6	Algebraic Bayesian Networks Second	2016	Berezin A.I., Romanov A.V.,

Рис. 11: Таблица публикаций, разбитая по категориям

Заключение

Таким образом, были решены все поставленные задачи, а именно: была создана веб-страница, позволяющая просматривать добавленные в систему данные (в виде таблицы), а также предоставляющая возможность добавлять новые записи и редактировать старые, был реализован сервис по управлению видом таблиц, был реализован универсальный компонент клиентской части, показывающий состояние элемента во время ожидания ответа от сервера, иными словами, отображение процесса загрузки данных с сервера (англ. «loadbar»), было реализовано сохранение пользовательских настроек в локальное хранилище, был реализован компонент клиентской части, отвечающий за конкатенацию полей записи из таблицы публикаций, был реализован импорт данных из пользовательских файлов. Также была обеспечена валидация форм и обеспечен адаптивный дизайн.

Все частные цели, а вследствие и общая цель достигнута. Сервис обладает всеми заявленными особенностями и функционалом, опубликован на сайте http://radoms.ru и находится в стадии тестирования.

Литература

[1] AngularJS. — URL: https://docs.angularjs.org/guide/services (дата обращения: 12.06.2016).

[2] DocxGenJS. — URL: http://github.com/open-xml-templating/docxtemplater/ (дата обращения: 12.06.2016).

[3] HTML5. — URL: http://www.w3.org/TR/html5 (дата обращения: 12.06.2016).

[4] Materialize. — URL: http://materializecss.com/cards.html (дата обращения: 12.06.2016).

[5] Papa Parse. — URL: http://papaparse.com/ (дата обращения: 12.06.2016).

[6] RADOMS. — URL: http://radoms.ru (дата обращения: 12.06.2016).

[7] Sass. — URL: http://sass-lang.com/ (дата обращения: 12.06.2016).

[8] SheetJS/js-xls. — URL: http://github.com/SheetJS/js-xls/ (дата обращения: 12.06.2016).

[9] SheetJS/js-xlsx. — URL: http://github.com/SheetJS/js-xlsx/ (дата обращения: 12.06.2016).